NAIST-IS-DT9961028

Doctor's Thesis

Partial Language Analysis using Support Vector Learning

HIROYASU YAMADA

March 15, 2002

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

Doctor'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
DOCTOR of ENGINEERING

HIROYASU YAMADA

Thesis committee: Yuji Matsumoto, Professor
Shunsuke Uemura, Professor
Kiyohiro Shikano, Professor

Partial Language Analysis using Support Vector Learning *

HIROYASU YAMADA

Abstract

The difficulty of analyzing full language often hampers the problem to develop practical natural language processing (NLP) systems. In this thesis, we focus on two fundamental *partial* language analysis, 1) Japanese named entity extraction and 2) Partial parsing in English. Named entity extraction is the task for extracting information such as proper nouns and numerical expressions from a document and classifying these expressions into some categories such as person, location, organization, and date. Partial parsing takes a sentence as an input and interprets it as a parsed sub-tree which does not include ambiguities. Both techniques are useful for not only a wide range of applications such as machine translation and information retrieval field but also preprocessing of full language analysis. We present corpus-based method for named entity extraction and partial parsing using a machine learning technique, Support Vector Machines(SVMs) and especially show how SVMs can work well in partial language analysis.

Keywords:

language analysis, named entity extraction, natural language parsing, information retrieval, support vector learning

*Doctor's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9961028, March 15, 2002.

Acknowledgements

I would like to thank Professor Yuji Matsumoto, my supervisor, for giving me the opportunity to study here in NAIST. Since I have been studying in NAIST, he has always helped me to overcome many problems, giving many valuable comments, his guidance and encouragement throughout this research. His interest in NL research, especially parsing, also helped make my research a lot more fun. I also wish to express my thanks to Professor Shunsuke Uemura, Professor Kiyohiro Shikano for their agreement to become my referees.

Thanks are also due to the members of Matsumoto lab., especially to the member of the SVM-Group. Their wide knowledge, technical comments, suggestions, and encouragement provided me with support to complete my thesis.

Finally, I wish to thank my parents and brothers for their unfailing assistance, support, constant encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the explosive growth of the Internet, a large number of electronic documents have been available. The keyword-match based information retrieval provides fast access to the relevant documents. For retrieving the more relevant information, it is necessary to analyze the documents by using natural language processing.

However, there exists many forms of ambiguity in natural language, thus it is difficult to analyze completely the documents. Partial language analysis is one of the most important techniques for solving this problem and is to analyze partial syntactic structure of a sentence such as noun phrase and verb phrase. In many natural language processing applications, partial analysis of language is often utilized rather than full language processing. For instance, in information retrieval, it needs to retrieve documents or texts with information content that is relevant to users' requirement at high speed. Partial language analysis such as noun phrase analysis is helpful to such a retrieval. Furthermore, partial language analysis is useful for not only these applications but also a preprocessing module for full language analysis.

In this thesis, we define "partial language analysis" as follows :

- **Text Chunking**:
  Text chunking is to divide text into syntactically related non-overlapping groups of words. Noun phrase chunking and named entity extraction are well known techniques of text chunking. Part-of-speech tagging is also regarded as text chunking.

- **Bracketing**:

Bracketing represents the result of text chunking as nested brackets which shows syntactic structure. Noun phrase bracketing, partial and full parsing techniques are categorized into the bracketing.

In this thesis, we focus on two problems of partial language analysis : **Japanse named entity extraction** and **partial parsing in English**.

Named entity extraction is a kind of the proper noun phrase analysis and is the task for extracting proper nouns, numerical expressions, etc. from a document and classifying these expressions into some categories such as person, location, organization and date. Named entity extraction is an 'intermediate task', i.e., it is not an end in itself, but rather is necessary at one level or another to accomplish most natural language processing tasks and is expected to improve the quality of information retrieval(IR). Especially, in question answering systems, named entity extraction techniques are necessary to answer users' queries such as 'when', 'where', and 'who' [1, 34].

Partial parsing is one of the fundamental techniques in natural language processing, and takes a sentence as an input and interprets it as a parse sub-tree which does not include ambiguities. This technique is useful for not only a wide range of applications information retrieval field and machine translation but also a preprocessing of full language parsing.

Over the last few years, a number of large scale machine-readable text corpora including strategies for evaluation have become available. Therefore, partial language analysis including named entity extraction and partial parsing has been studied based on statistics or stochastic models of language. The statistical approach acquires automatically rules for language analysis using statistic information from a large number of documents. It has overcome the problems of conventional rule-based approaches which rely on linguistic and domain specific knowledge. Statistical models are now widely used as robust mechanisms. However, the approach still has a problem: it is not clear how to handle rare usages.

Recent approaches to machine learning, especially the results of supervised learning approaches, show the effectiveness of their techniques and other tasks in natural language processing with high accuracy [33, 43, 27, 2, 44, 32, 15].

In this thesis, we focus on a supervised learning algorithm **Support Vector Machines** (SVMs) which are developed by Vapnik et al [41, 42]. SVMs have been found to have a good performance for many natural language processing tasks such as chunk-

ing [36, 37], part-of-speech tagging [38], dependency structure analysis[19], word sense disambiguation[12] and natural language processing applications such as text categorization [39, 40, 45, 10, 11].

In this thesis, we propose a method for two partial language analysis: Japanese named entity extraction and partial parsing in English using support vector learning, and show the effectiveness of our method throughout the experiments based on a large corpus.

The rest of this thesis is organized as follows. Firstly, in Chapter 2, we report on some studies related to the named entity extraction and parsing with some statistics and machine learning approaches. In Chapter 3, we explain an overview of the SVMs and its theoretical advantage for partial language analysis. In Chapter 4, we propose a method for Japanese named entity extraction using SVMs. In Chapter 5, we propose a partial parser which constructs a parse binary tree with deterministic bottom-up algorithm Our parser regards each parsing process as a classification task which classifies the context into the action for constructing a parse tree. We apply SVMs to the task.

Finally, Chapter 6 summarizes the thesis and ends with conclusions and future work.

Chapter 2

Related Work

2.1 Named Entity Extraction

Named entity extraction is one of the noun phrase chunking. The task is to extract a phrase like proper nouns and classify some category such as “PERSON”, “LOCATION” and “DATE”. Most of the previous studies of named entity extraction have applied chunking techniques to the task. We first explain some named entity representation models.

2.1.1 Representation of Named Entity

Ramshow et al. proposed a representation of chunk called ’Inside/Out’ for text chunking[21]. Tjong Kim Sang et al. proposed four models for representation of a chunk, i.e., IOB1, IOB2, IOE1, and IOE2 [6]. The IOB2 models in his study is equal to ’Inside/Out’ model in Ramshow’s study. In this thesis, we call the Inside/Out model as the IOB2 model. Sekine[35] and Uchimoto[18] used Start/End model as the representation of named entity, especially for Japanese. Furthermore, Uchimoto et al. add three tags to the Start/End model: PRE, POST and MID. These 6 models are summarized as follows:

- IOB1: If the first word inside a named entity ’X’ immediately follows another named entity, the word is assigned to ’B-X’ tag, otherwise to ’I-X’ tag.
- IOB2: The first word inside a named entity ’X’ is always assigned to ’B-X’ tag.

- IOE1: If the final word inside a named entity 'X' immediately precedes another named entity, the word is assigned to 'E-X' tag, otherwise to 'I-X' tag.
- IOE2: The final word inside a named entity 'X' is always assigned to 'E-X' tag.
- S/E(Start/End): If a word itself is a named entity 'X' then the word is assigned to 'S-X' tag. The first or final word inside a named entity 'X' is always assigned to 'B-X', 'E-X' tag respectively. Neither a first nor final word inside a named entity 'X' is assigned to 'I-X' tag .
- S/E+(Uchimoto's model): this model is equal to the S/E model except for non-NE tag. If the non-NE word precedes named entity 'X', the word is assigned to 'PRE-X' tag. If the non-NE word is just one word between different named entities, the word is assigned to 'MID-X' (X is the former named entity category). If the non-NE word follows named entity 'X', the word is assigned to 'POST-X'.

In all of 6 models, if a word is not named entity, then the word is assigned to 'O' tag. We call each tag **NE-tag** in this thesis.

Figure 2.1 illustrates the representation of named entity using 6 models(IOB1, IOB2, IOE1, IOE2, S/E and S/E+) for a sample sentence "エリツィン大統領は四日, 日米両国 ".

	PERSON			DATE		
	エリツィン	大統領	は	四	日	,
IOB1	I-PERSON	O	O	I-DATE	I-DATE	O
IOB2	B-PERSON	O	O	B-DATE	I-DATE	O
IOE1	I-PERSON	O	O	I-DATE	I-DATE	O
IOE2	E-PERSON	O	O	I-DATE	E-DATE	O
S/E	S-PERSON	O	O	B-DATE	E-DATE	O
S/E+	S-PERSON	POST-PERSON	PRE-DATE	B-DATE	E-DATE	MID-DATE

	LOCATION	LOCATION		…
	日	米	両国	…
	I-LOCATION	B-LOCATION	O	
	B-LOCATION	B-LOCATION	O	
	I-LOCATION	E-LOCATION	O	
	E-LOCATION	E-LOCATION	O	
	S-LOCATION	S-LOCATION	O	
	S-LOCATION	S-LOCATION	POST-LOCATION	

Figure 2.1. Representation of named entity for 6 models

2.1.2 Named Entity Extraction using Machine Learning

Decision Tree

Sekine et al.[35] proposed a method for Japanese named entity recognition using decision tree learning. Their method recognizes a named entity in a sentence by classifying each word in the sentence into a NE-tag with Start/End model. They constructed a decision tree classifier using following three types of features.

- Part-of-speech tagged by JUMAN [20]
  They define the set of categories based on the major and minor part-of-speech tags.

- Character type of words
  Character type of words, such as *Kanji, Hiragana, Katakana, Alphabet, number,* or *Symbol,* etc. and some combinations of these character types.

- Special Dictionaries
  They use a dictionary which a word belongs to some types of named entity. The dictionary created is based on dictionary entries of JUMAN, found on Web documents or, from human knowledge.

They reported their experiments using a small set of data from two different domain: One is the Accident Report Domain which consists of a training set of 103 articles and a test set of 11 articles containing 2,368 named entities in all. The results for the data set was 85 at F-value. The another one is the Executive Succession Domain for demonstrating the portability of their method. The results is 84 at F-value for the data set.

Nobata proposed a method for Japanese named entity extracion using decision tree learing[47]. Their method is similar to Sekine's study, and the result attained 70.3 of F-value for GNENRAL domain test data in IREX workshop[14].

Isozaki

Isozaki proposed a method using a simple rule generator and decision tree learning for Japanese Named entity extraction [9]. His method is summarized as follows:

1. Generating *recognition rules*

   In his method, each named entity for training data is converted to a recognition rule via morphological analyser. A *recognition rules* are written as

   $pattern_1, pattern_2, \cdots, pattern_n \rightarrow X, c_1, c_2$

   where n is the number of words inside the named entity category X. $Pattern_i$ can be represented as a word: character-type: a part-of-speech of i-th word. c_1 refers to the restriction of proper nouns by a suffix dictionary, and c_2 is that it does not restrict numbers by a suffix dictionary. Every c_i has either of two values, 0(no restriction) and 1(restriction).

   For example, in the pattern "大阪銀行", which consists of two words: "大阪" and "銀行", two patterns, i.e., 大阪: all-kanji:location-name, and 銀行:all-kanji:common-noun are generated from the following rules.

   ::location-name,
   :*:common-noun
   → ORGANIZATION, 0, 0

   In the above rule, '*' shows that it matches any sequence of words.

2. Refinement of recognition rules by using decision tree learning, and generating named entity candidates

   By applying each recognition rule to the untagged training data, named entity candidates for the rule are obtained. By comparing the candidates with the given answer for the training data, these candidates are classified into two types, i.e., positive and negative examples for the recognition rule. Finally, decision tree learning is applied to classify these examples correctly.

3. Arbitration of these named entity candidates

   Once the refined rules are generated, they are applied to a new document and a large number of named entity candidates are obtained. They use a kind of *left-to-right longest match method* to rank candidates. The method operates as

follows: First, compared with each starting point of named entity candidates, then selected the earliest ones. If two or more candidates start at the same point, their ending points are compared and the longest candidate is selected. The procedure is repeated until the candidate set becomes empty.

They evaluated their method by comparing it with maximum entropy system(ME). They use the standard IREX[14] training data and the formal run test data. For tokenization, they use ChaSen 2.2.1 [46]. which has about 90 part-of-speech tags and large proper noun dictionaries, i.e., 32,167 person names, 16,610 organization names, 67,296 location names, and 26,106 other proper nouns. The result of ME attained at 82.8 of F-value when length of left and right context is one word respectively. When left and right context length are two words respectively, ME attained at 82.7 of F-value. The success rate of their method, on the other hand, was 84.1 when left context is one word and as right context is two words.

2.1.3 Maximum Entropy Model

Uchimoto et al. proposed a method based on a maximum entropy (ME) model and transformation rules for Japanese named entity extraction [18]. The maximum entropy model has recently been successfully introduced to a variety of natural language application, such as part-of-speech tagging [30], parsing [31, 8] and so on.

The method estimates a probability $P(\text{NE-tag}|w)$ by using ME model, where w is a word in a sentence, then extracts named entity to search for the optimal NE-tag sequences for whole the sentence by using Viterbi algorithm.

Further, in Japaneses named entity extraction, named entity boundary is often different from a word segments by morphological analyser. In this case, the system can not extract named entity correctly. To cope with the problem, they used a transformation based method similar to Brill's one in the part-of-speech tagging [3]. Transformation rules are automatically acquired from the training data using the difference between word segmentation by morphological analyser and correct named entity boundary. Let word "在日" be the categories which is illustrated the follows.

word	part-of-speech (major tags)	part-of-speech (minor tags)	NE-tag
‘在日’	‘名詞’	‘サ変名詞’	O

if “日” in “在日” is the named entity of “LOCATION”, then ‘在日’ is translated into two words. These two words are assigned to new part-of-speech and correct NE tags as follows.

word	part-of-speech (major tags)	part-of-speech (minor tags)	NE-tag
‘在’	‘名詞’	‘普通名詞’	PRE-LOCATION
‘日’	‘名詞’	‘普通名詞’	‘S-LOCATION’

They use lexical items and part-of-speech features, and 40 NE-tags. They used the CRL named entity data, the IREX-NE dry-run training data, and the formal-run data, which consist of about 12,000 sentences in their experiment. They also sued F-measure as an evaluation measure. They reported that the method achieved at 85.75 of F-value for the test set of restricted domain, and 80.17 at F-value for general domain in IREX competition[14]. Although their results have shown satisfactory performance, they used a simple feature selection in the training data, i.e. they only use words in the data with more than two. This means that it deteriorate the performance of the training data itself.

Sassano

Sassano et al. proposed *variable length model* for Japanese named entity extraction [23, 22]. The idea behind the method is that it overcomes the disadvantage of the fixed context length model(especially, 3 context length model like Sekine and Nobata’s studies) and it incorporates richer contextual information as well as patterns of constituent morphemes within a named entity. In principle, as part of the training phase this model considers which of the preceding/subsequent words constitute one named entity together with the word at the current position in a sentence.

It also considers several words in the left/right contexts of the NE. Their method restricts the model to explicitly considering the cases of NEs of the length up to three

words and only implicitly considering those longer than three words. It also restricts the model by considering two words in both left and right contexts of the named entity.

As the learning model, they used decision list and maximum entropy model. and use a IOB2 and Start/End models in figure 2.1. They use three kinds of features :(i) a pair of word and part-of-speech tag, (ii) a pair of character types and part-of-speech, and (iii) part-of-speech. The character type consists of 6 elements: *Hiragana*, *Kanji*, *Katakana*, Numeric, *Alphabet*, *Symbol*, and their combination.

They evaluated their method using the IREX workshop's training and test data. They compared the results of the combinations of the two NE representation models: the IOB2 and Start/End models, two approaches to contextual feature design: the 3, 5 context length and the variable length models, and two learning algorithm: decision tree and Maximum entropy model. Among these combinations, the variable length model, Start/End models and ME learning outperforms other combinations and the accuracy rate of 82.8 of F-value.

2.2 Parsing in English

Jensen

Jensen et al. proposed a partial parsing 'fitted parse' [16]. The significant feature of the parsing is to produce a reasonable approximate parse that can serve as input to the remaining stages of processing, if the rules of a more conventional syntactic grammar are unable to produce a parse for an input sentence.

The fitting procedure begins after the *core grammar* has been applied in a bottom-up, parallel fashion, but has failed to produce an complete parse tree of the sentence. At this point, as a by-product of bottom-up parsing, records are available for inspection that describe the various segments of the sentence from many perspectives, according to the rules that have been applied.

The algorithm of fitting proceeds in two main stages: first, a head constituent is chosen; next, remaining constituents are fitted in. In the first stage, candidates for the head are tested preferentially as follows, from most to least desirable:
(a)Verb phrases with tense and subject;
(b)Verb phrases with tense but no subject;
(c)Segments other than verb phrase;

(d)Untensed verb phrases;

If more than one candidate is found in any category. The one preferred is the widest. If there is a tie for widest, the leftmost of those is preferred. If there is a tie for leftmost, the one with the best value for the parse metric is chosen. If there is still a tie, an arbitrary choice is made. If the head constituent covers the input sentence, the fitting process is complete, otherwise to go to the next stage.

In the next stage, the remaining constituents are added on either sided, with the following order of preference:
(i) segments other than verb phrases
(ii) untensed verb phrases ;
(iii) tensed verb phrases;

As with the choice of the head, the widest candidate is preferred at each step. The fit moves outward from the head. Both leftward to the beginning of the strnig, and rightward to the end until the entire input sentence has been fitted into the best approximate parse tree. The overall effect of the fitting process is to select the largest chunk of sentence like material within a text string and consider it to be central, with left-over chunks of text attached in some resonable manner.

2.2.1 Statistical Parsing

In this section, we report on some statistical parsing using standard data set "Penn treebank Wall Street Journal" corpus. We first describe the standard data set and evaluation measure in this corpus. Then we report on four studies using this corpus.

Penn Treebank Wall Street Journal Corpus

The Wall Street Journal portion of the Penn Treebank corpus developed by Marcus et al. for wide range of research fields in natural language processing [28, 29]. The corpus consists of over 4.5 million words in 24 sections of American English and is annotated for part-of-speech information and syntactic bracketing.

The standard data set consists of two parts: training and test set. The training set contains 39,832 sentences from section 2 to section 21 and test set consists of

2,416 sentences of section 23 in the corpus. The standard evaluation measure based on PARSEVAL [5] is follows:

$$\text{Labeled Recall (LR)} = \frac{\text{number of correctly detected constituents}}{\text{number of constituents annotated by human}}$$

$$\text{Labeled Precision} = \frac{\text{number of correctly detected constituents}}{\text{number of constituents annotated by the parser}}$$

$$\text{Cross Brackets (CB)} = \frac{\text{number of constituents crossing a annotated constituent}}{\text{number of sentences}}$$

$$\text{0 Cross Brackets (0CB)} = \text{percentage of sentences which have 0 crossing brackets.}$$

$$\text{2 or less Cross Brackets(2CB)} = \text{percentage of sentences which have less than 2 crossing brackets.}$$

Magerman

Magerman presented a statistical parser called SPATTER which is based on decision-tree learning techniques constructing a complete parse tree for every sentence [4]. A parse tree for a sentence is constructed by starting with the sentence's words as leaves of a tree structure, and labeling and extending these nodes until a single-rooted, labeled tree is constructed. This pattern recognition process is driven by the decision-tree learning.

In his method, representation of constituent consists of four elements: words, tags, labels, and extensions.

Figure 2.2 shows the representation of constituent in the method. the word features are string of the word itself(e.g, a, brown, and cow in figure 2.2). The tag feature can take on any value in the part-of-speech tag set, like 'DT', and the label feature can take on any value in the non-terminal set, like 'right'. The extension can take on any of the following five values.

- right: 'Right' is a node which is the first child of a constituent.
- left: 'Left' is a node which is the last child of a constituent.

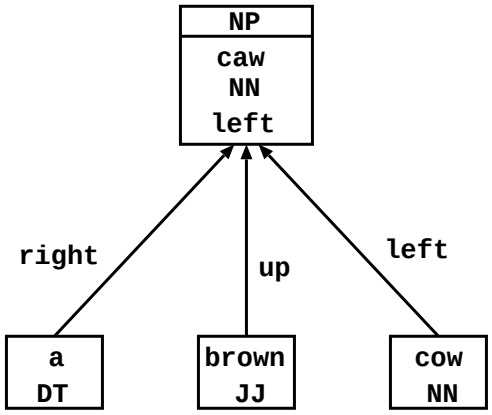

Figure 2.2. Representation of constituent and labeling of extensions in SPATTER for sample noun phrase “a brown caw”.

- up: ‘Up’ is a node except for the first and last child of a constituent.
- unary: ‘Unary’ is a node which is a child of an unary constituent.
- root: the node is the root of the tree.

SPATTER consists of three main decision tree models: a part-of-speech tagging model, a node-extension model, and a node-labeling model. These decision tree models are grown using following question with traversing the parsed sentence in training data.

- What is the X at the current node?
- What is the X at the node to the Y?
- What is the X at the node two nodes to the Y
- What is the X at the current node’s first child from the Y ?
- What is the X at the current node’s second child from the Y ?

where X is one of the word, tag, label or extension, and Y is either left or right.

The training is proceeded as follows. The training corpus is divided into two set, one is tree growing, approximately 90% and another is tree smoothing, 10%. For

each parse sentence in the tree growing corpus, the correct state sequence is traversed. Each state transition from s_i to s_{i+1} is used as a training example for the decision-tree growing process for the appropriate tree. After the decision tree are grown, they are smoothed by the tree smoothing corpus. By searching the highest probability of a parse tree, the parsing is proceeded. The probability of a parse is just the product of the probability of each of the actions made in constructing the parse, according to the decision-tree models, the parsing procedure.

They evaluated SPATTER using the Penn Treebank Wall Street Journal corpus. The result attained both 84.6% precision and recall for sentences of 40 words or less.

Collins

Collins proposed generative lexicalized models for statistical natural language parsing [26]. The his model is based on probabilistic context-free grammar. The model searches the best parse tree T_{best} that maximizes conditional probability $P(T|S)$ as follows, where S is the input Sentence and T is the parse tree. The

$$\begin{aligned} T_{best} &= argmax_T P(T|S) \\ &= \arg\max_T P(T,S)P(S) \\ &= \arg\max_T P(T,S) \end{aligned}$$

To maximize conditional probability $P(T|S)$ is equal to maximize the joint probability $P(T,S)$. A joint probabilty $P(T,S)$ is then defined by attaching probabilities to a top-down derivation to tree using probabilistic context-free grammars rules $LHS \rightarrow RHS$, where LHS and RHS stand for "left hand side" and "rigt handside" respectively. Thus a conditional probability $P(T,S)$ can be written by n applications of context-free rules $LHS_i \rightarrow RHS_i$ $(1 \leq i \leq n)$ as follow:

$$P(T,S) = \prod_i P(RHS_i|LHS_i)$$

Furthermore, each probability $P(RHS_i|LHS_i)$ can be wriiten as lexicalized constituent of RHS_i and using head word of the LHS_i

$$
\begin{aligned}
P(RHS_i|LHS_i) &= P(L_{n+1}(l_{n+1})\cdots L_1(l_1)H(h)R_1(r_1)\cdots R_{m+1}r_{m+1}|Par(h)) \\
&= P_h(H|Par(h)) \times \\
&\quad \prod_{i=1\cdots n+1} P_l(L_i(l_i)|L_1(l_1)\cdots L_{i-1}, Par(h), H) \times \\
&\quad \prod_{i=1\cdots m+1} P_r(R_j(r_j)|L_1(l_1)\cdots L_{n+1}(l_{n+1}), R_1(r_1)\cdots R_{j-1}(r_{j-1}), Par(h), H)
\end{aligned}
$$

where H is the head-child of the phrase, which inherits the head-word h from its parent Par. $L_1 \cdots L_n$ and $R_1 \ldots R_m$ are left and tight modifiers of H. Either n or m may be zero, and $n = m = 0$ for unary rules. By the two assumptions: the modifiers are generated independently of each other and distance can be incorporated into the model by increasing the amount of dependence between the modifires. The probaility can be written as follows:

$$
P_l(L_i(l_i)|L_1(l_1)\cdots L_{i-1}, Par(h), H) = P_l(L_i(l_i)|D_l(i-1), Par(h), H)
$$

$$
P_r(R_j(r_j)|L_1(l_1)\cdots L_{n+1}(l_{n+1}), R_1(r_1)\cdots R_{j-1}(r_{j-1}), Par(h), H) =
$$

$$
P_r(R_j(r_j)|D_r(i-1), Par(h), H)
$$

where $D_l(i-1)$ and $D_r(i-1)$ are functions of the surface string below the previous modifiers. The distance measure is a vector with the following two elements: (1) is the string of zero length? ; (2) does the string contain a verb ?

Finaly, the model is extended to make the complement/adjunct distinction and to have parameters corresponding directly to probability distributions over subcategorization frames for head-words. The model, as a preprocessing, adds the "-C" suffix to all non-terminals in Penn treebank training data that satisfy the following conditions:

1. The non-terminal must be:(1) an NP, SBAR, or S whose parent is an S; (2)an NP, SBAR, S, or VP whose parent is a VP; or (3) an S whose parent is an SBAR
2. the non-terminal must not have one of the followin semantics tags: ADV, VOC, BNF, DIR, EXT, LOC, MNR, TMP, CRL or PRP.

In addition, the first child following the head of a prepositional phrase is marked as a complement. The generative process is extended to include a probabilistic choice of left and right subcategorization frames as follows:

1. Choose a head H with probability $P_H(H|Par(h))$
2. Choose left and right subcat frames, LC and RC, with probabilities $P_{lc}(LC|Par(h),H)$ and $P_{rc}(RC|Par(h),H)$. Each subcat frame is a multiset specifying the complements that the head requires in its left or right modifiers.
3. Generate the left and right modifiers with probabilities $P_l(L_i(l_i)|D_l(i-1),Par(h),H,LC)$ and $P_r(R_j(r_j)|D_r(i-1),Par(h),H,RC)$ respectively. Thus the subcat requirements are added to the conditioning context. As complements are generated they are removed from the appropriate subcat multiset.

In his experiment, the accuracy of this model is over 88.1% recall and precision for the standard data set of Penn Treebank Wall Street Journal corpus.

Ratnaparkhi

Ratnaparkhi proposed a machine learning method to parse sentences with maximum entropy model [31]. His parser constructs labeled syntactic parse trees with actions similar to those of a standard *shift-reduce* parser. The actions of the parser are produced by some procedures(passes). The first pass takes an input sentence, and uses TAG to assign each word a part-of-speech tag. The second pass takes the output of the first pass and uses CHUNK to determine the 'flat' phrase chunks of the sentence, where a phrase is 'flat' if and only if it is a constituent whose children are not constituents. Starting from the left, CHUNK assigns each pair a 'chunk' tag, either Start X, Join X, or Other, where X is a constituent label. The chunk tags are then used for chunk detection, in which any consecutive sequence of words $w_m, \cdots, w_n$ ($m \geq n$) are grouped into a 'flag' chunk X if w_m has been assigned Start X and $w_{m+1}, \cdots, w_m$ have all been assigned Join X. The third pass always alternates between the use of BUILD and CHECK, and completes any remaining constituent structure. BUILD decides whether a tree will start a new constituent or join the incomplete constituent immediately to its left. BUILD always processes the leftmost tree without any Start X or Join X annotation. After, BUILD, control passes to CHECK, which finds the most recently

proposed constituent, and decides if it is complete. If CHECK decides yes, then the proposed constituent takes its place in the forest as an actual constituent, on which BUILD does its work. Otherwise, the constituent is not finished and BUILD processes the next tree in the forest, t_{n+1}.

He evaluated his method using the same data as Collins[24, 25] and Charniak[7]: Penn Treebank of the Wall Street Journal developed at the University of Pennsylvania. The maximum entropy parser was trained on sections 2 through 21 of the Wall Street Journal corpus, and tested on section 23. Table 2.1 describes the number of training events extracted from the Wall Street Journal corpus.

Table 2.1. Sizes of Training Events, Actions, and Features

Procedure	# of training events	# of actions	# of features
TAG	935,655	43	119,910
CHUNK	935,655	41	230,473
CHECK	1,097,584	2	182,474
BUILD	1,097,584	52	532,814

The result of his experiments is over 86.3% recall and precision for the standard data set of Penn Treebank Wall Street Journal corpus.

Charniak

Charniak proposes a new parser for parsing down to Penn tree-bank style parse trees [8]. The characteristic of his approach is to use a 'maximum-entropy-inspired' model for conditioning and smoothing.

The parser is based upon a probabilistic generative model. For all sentences s, and all parses π, the parser assigns a probability $p(s,\pi) = p(\pi)$, the equality holding when we restrict consideration to π whose yield is s. Then for any s the parser returns the parse π that maximizes this probability. That is, the parser implements the function:

$$argmax_{\pi} p(\pi|s) = argmax_{\pi} p(\pi)$$

They estimate the above probability $p(\pi)$ by using *maximum-entropy-inspired* model. A conditional probability $p(a \mid H)$ with a set of features $f_1, \cdots, f_j$ that connect a to the history H. is computed as follows:

$$p(a \mid H) \quad = \quad \frac{1}{Z(H)} e^{\lambda_1(a,H) f_1(a,H) + \cdots \lambda_m(a,H) f_m(a,H)} \tag{2.1}$$

Here the λ_i are weights between negative and positive infinity that indicate the relative importance of a feature. $Z(H)$ is the normalizing factor.

Maximum-entropy models have two benefits for a parser builder. First, the probability model should be easily changeable, i.e., it is enough just to change the set of features use. Second, the features used in these models need have no particular independence condition of one another. Further, they used smoothing method which can write as follows:

$$p(a \mid b_{1,n}) \quad = \quad \lambda(b_{1,n}) \hat{p}(a \mid b_{1,n}) + (1 - \lambda(b_{1,n})) p(a \mid b_{1,n-1}) \tag{2.2}$$

$\lambda(b_{1,n})$ is the interpolation factor, where $b_{1,n} = b_1, \cdots b_n$.

They created a parser based upon the maximum-entropy-inspired model, smoothed using standard deleted interpolation. They used the chart parser as a first pass to generate candidate possible parses to be evaluated in the second pass by his probabilistic model. For runs with the generative model based upon Markov grammar statistics, the first pass uses the same statistics, but conditioned only on standard PCFG information.

The result achieves 90.1% average precision/recall for sentences of length ≤ 40, and 89.5% for sentences of length ≤ 100, when they used 'standard' sections of the Wall Street Journal Treebank.

Chapter 3

Support Vector Machines

Support Vector Machines (SVMs) were developed by V. Vapnik et al. based on structural risk maximization from statistical learning theory [41, 42]. We first explain the binary classification problem and an overview of SVMs. Then, we show the theoretical advantage of SVMs for partial language analysis. Section3.3 will introduce the kernel method for dealing with non-linear classification problem. The last section shows two conventional methods for multi-class classification with SVMs.

3.1 Overview of Support Vector Machines

Binary classification problem

Suppose we are given a training set S of l examples, such as
$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_3), \cdots, (\mathbf{x}_i, y_i), \cdots, (\mathbf{x}_l, y_l)\}$
where $\mathbf{x}_i \in R^n$, $y_i \in +1, -1$. If $y_i = 0$ represents positive example, and otherwise to negative example.

Binary classification is frequently performed by using a real-valued function $f : X \subseteq \mathbf{R}^n \mapsto \mathbf{R}$ from given training examples. Let $\mathbf{x}' \in \mathbf{R}^n$ be a test example, the $\mathbf{x}'$ is assigned to positive class, if $f(\mathbf{x}') \leq 0$, and otherwise to the negative example.

SVMs is a kind of the binary classifier with a linear function $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$ in the feature space X. We explain a case in which all training examples are linear separable.

Figure 3.1 shows an overview of SVMs in linear separable case. SVMs optimize parameters $\mathbf{w}$ and 'b' based on maximum *margin* strategy. The margin is defined as

the distance between two separate hyperplanes which is equal distance from and a parallel with the optimal hyperplane. The advantage of the maximum margin strategy is not fitting for any training examples. Like hyperplane in Figure 3.1. The strategy theoretically guarantees low generalization error for unknown example even in a high dimensional feature space.

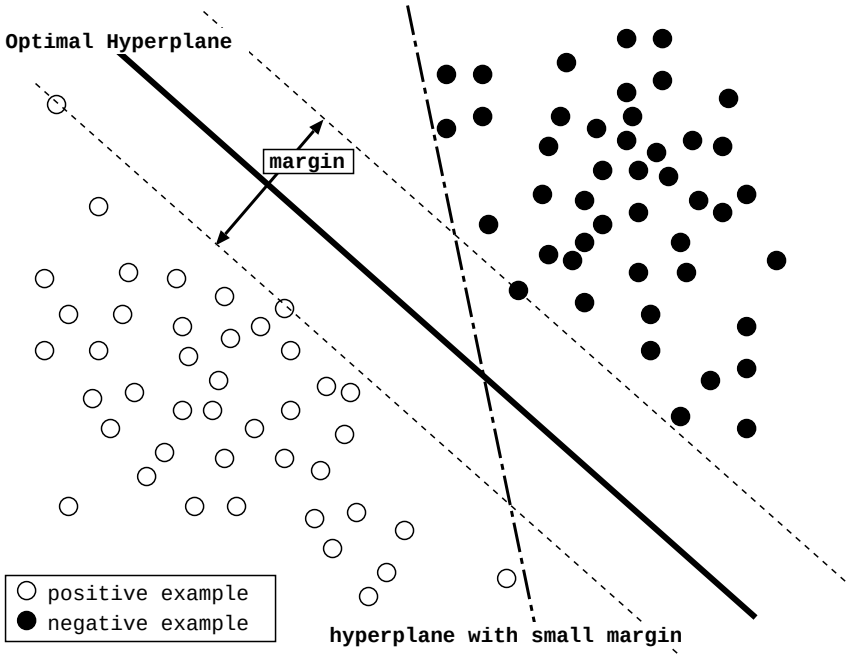

Figure 3.1. Overview of Support Vector Machines

The margin can be written as norm of $\frac{2}{\|\mathbf{w}\|}$. Thus, we can find an optimal hyperplane maximizing margin by solving the following problem:
Primal Problem

$$\begin{aligned} &\textit{minimizes} && \tfrac{1}{2}\|\mathbf{w}\|^2 \\ &\textit{subjectto.} && y_i(\mathbf{x}_i \cdot \mathbf{w} + b \geq 1), i = 1, ..., l \end{aligned}$$

The primal problem can transform into the following dual problem by introducing Lagrange multiplier α_i. The primal Lagrangian is the following formula:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right) \tag{3.1}$$

The corresponding dual is found by differentiating with respect to **w** and b,

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i = \mathbf{0} \tag{3.2}$$

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.3}$$

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{3.4}$$

Eq.(3.4) shows that the optimal **w** can be written as a linear combination of training examples. And substitute Eq.(3.4) for Eq.(3.1), we can obtain the following dual problem:

Dual Problem

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{3.5}$$

$$\sum_{i=1}^{l} y_i \alpha_i = 0, \quad \alpha_i \geq 0 \tag{3.6}$$

The dual problem is the well known quadratic problem, it is possible to solve optimal α_i. Finally, the optimal hyperplane can be expressed the following function:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \right) \tag{3.7}$$

Notice that there is a Lagrange multiplier α_i for every training examples. In the solution, these examples for which $\alpha_i > 0$ are called *support vector*, and lie on one of the separate hyperplanes in Figure 3.1. All other training examples have $\alpha_i = 0$, and line on outside a pair of separate hyperplanes. The support vectors are the critical elements of the training set.

The next section explains why the maximum margin strategy guarantees for low generalized error bound even in a high dimensional feature space.

3.2 Low Generalization Error in High Dimensional Feature Space

The following theorem proof by Vapnik et al [42].

Theorem 1 *Let H be a hypothesis space having VC dimension d. for any probability distribution D on $X \times \{-1,1\}$, with probability $1-\delta$ over l random examples S, any hypothesis $h \in H$ that makes Err_{emp} errors on the training set S has error no more than*

$$Err \leq Err_{emp} + \sqrt{\frac{h(\ln(2l/h)+1) - \ln(\delta/4)}{l}}$$

provided $d \leq l$

Decreasing the generalization error of hypothesis f depend on only the VC dimension h, if the $Error_{emp}$ and l are fixed. And VC dimension of SVMs bounds the following equation,

$$h \leq \min(D^2/\rho^2, n) + 1 \tag{3.8}$$

where D is the radius of hypersphere covering all training examples, ρ is the margin, and n is the dimension of the feature space. Eq.(3.8) denote that VC dimension of SVMs depend on only the margin if the number of dimension of the feature space is sufficiently large. Thus maximum margin strategy can decrease the low generalization error even in high dimensional feature space.

In many NLP tasks including partial language analysis, it is often use a lot of features such as word, part-of-speech tags, character type of the word, and so on. Therefore, we need to deal with high dimensional feature space. However, most previous learning alorithms were inevitable to over-fitting when they applied to the high dimensional feature space.

However, maximum margin strategy of SVMs make it possible to avoiding the over-fitting even in a high dimensional feature space.

The next section explains the kernel method for dealing with a case in which classification problem is potentially not linear separable.

3.3 Learning with Combination of Features by Kernel Method

Limitation of the linear learning machine is that it can not deal with non-linear separable problem like well-known XOR problem. Kernel function can deal with the non-linear classification by using implicitly mapping the original feature space into high dimensional space. Especially the polynomial kernel function make it possible to deal with combination of any features.

We illustrates the advantage of kernel function using a simple example. Consider the case of two dimensional feature space and polynomial kernel function $K(\mathbf{a},\mathbf{b}) = (\mathbf{a}\cdot\mathbf{b}+1)^2$. Let $\mathbf{a},\mathbf{b}$ be two dimensional vectors (a_1,a_2) and (b_1,b_2) respectively. Suppose that a mapping function $\Phi : \mathbf{R^2} \mapsto \mathbf{R^6}$ is in Eq.(3.11).

$$\begin{aligned} K(\mathbf{a},\mathbf{b}) &= (\mathbf{a}\cdot\mathbf{b}+1)^2 && (3.9)\\ &= (a_1b_1+a_2b_2+1)^2 \\ &= a_1^2b_1^2+2a_1a_2b_1b_2+2a_1b_1 \\ &\quad +2a_2b_2+a_2^2b_2^2+1 \\ &= \Phi(\mathbf{a})\cdot\Phi(\mathbf{b}) && (3.10) \end{aligned}$$

$$\begin{aligned} \Phi(\mathbf{x}) &= (x_1^2,\sqrt{2}x_1x_2,\sqrt{2}x_1,\sqrt{2}x_2,x_2^2,1), && (3.11)\\ &\quad \mathbf{x}=(x_1,x_2) \end{aligned}$$

Eq.(3.9) can be written as Eq.(3.10), this is equal to the inner product in the mapping space $\mathbf{R}^6$. Note that the space mapped by the function Φ have some elements which represent a combination of two elements like $\sqrt{2}x_1x_2$ in the original space.

Thus the value of inner product $\Phi(\mathbf{a})\cdot\Phi(\mathbf{b})$ in the mapping space can be obtain by calculating value of function $K(\mathbf{a},\mathbf{b}) = (\mathbf{a}\cdot\mathbf{b}+1)^2$.

Applying kernel method to SVMs can only replace $\mathbf{x}_i\cdot\mathbf{x}_j$ with $K(\mathbf{x}_i\cdot\mathbf{x}_j)$ for optimization problem described previous section. Thus we can obtain the following *dual problem*:

Dual Problem with Kernel Function

$$\sum_{i=1}^{l} \alpha_i \quad - \quad \frac{1}{2} \sum_{i,j}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i \cdot \mathbf{x}_j) \tag{3.12}$$

$$\sum_{i=1}^{l} y_i \alpha_i = 0, \quad \alpha_i \geq 0 \tag{3.13}$$

And the decision function $f(x)$ can be written as the following:

$$f(\mathbf{x}) \quad = \quad sgn\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i \cdot \mathbf{x}) + b\right) \tag{3.14}$$

This is equal to linear classification problem in the high dimensional feature space mapped by function Φ.

The advantage of kernel method in NLP including partial language analysis is that it can handle any combination of features. For instance, collocations as an association among lexical items, where the probability of item *x* co-occurring with items *y*, *z*, ... is very important features to solve many NL ambiguity problems. Kernel method makes it possible to treat these collocations by mapping the original space to a higher dimensional space.

Conventionally, these features were selected by some heuristics based on the human institution, or using statistical information such as mutual information between words. This is because that to use a large number of features causes the problem of computational cost and over-fitting to data with increasing dimensions of feature space. The approaches based on feature selection, however, often deteriorate the overall performance of the learning, because it is difficult select all the information needed.

3.4 Multi-Class Classification

SVMs is basically introduced for solving binary classification, while most of the NLP tasks are a multi-class classification problem. In this section, we introduce two traditional methods for extension binary classifier to multi-class classifier:

- one versus rest method:
  In this method, *k* different classifiers are constructed, one classfier for each class. In the classification phase the classifier with the maximal output defines the estimated class label of the current input vector.

- pairwise method:
  For each possible pair of classes a binary classifier is constructed(i.e. $\frac{k(k-2)}{2}$). Each classifier is trained on a subset of the training data containing only training examples of the two involved classes. In the classification phase, all $\frac{k(k-2)}{2}$ binary classifiers are combined through a majority voting scheme to estimate the final classification. In use of SVMs as binary classifier, the class with the maximal number of votes among all $\frac{k(k-2)}{2}$ classifieres is the estimation.

Computational Cost

It has not well formulated the difference of the performance between one versus rest method and pairwise method when we use SVMs as a binary classifier. However, from computational cost point-of-view, these approaches are quite different from each other. A time complexity of SVMs is known as $O(l^2) \sim O(l^3)$, where l is the number of training data.

Suppose k categories classification of l training data, average of size of training data per one categories is $\frac{l}{k}$. And the time of training for all of categories represents as $k' \cdot T(l')$, where $T(l')$ represents the time for learning one binary classifier using l' training data. And k' is the number of binary classifier. We assume that $T(l')$ of one SVMs is roughly $T(l') = C \cdot l'^3$ where C is a constant. One versus rest method, learning is done in time Ckl^3 and pairwise method is $C\frac{k(k-1)}{2}(\frac{2l}{k})^3 = C\frac{4l^3(k-1)}{k^2}$.

In practical, it strongly depends on the task or a given data for which techniques, i.e. one v.s. rest or pairwise method is better. Therefore, we use either of these methods depending on computational time or accuracy of the task.

3.5 Summary

This chapter describes an overview of the SVMs, especially we mentioned that why SVMs is effective for partial language analysis. The generalization performance of SVMs does not depend on the size of dimensions of the feature space, even in a high dimensional feature space. Furthermore, SVMs can induce an optimal classifier which considers the combination of any features by virtue of polynomial kernel functions. In the next two chapters, we present two methods which apply SVMs to partial language analysis: Named Entity extraction and partial parsing.

Chapter 4

Japanese Named Entity Extraction using Support Vector Machines

In this chapter, we describe a method of Japanese named entity extraction by using SVMs[13]. The first section shows the definition of Japanese named entity extraction. Next, we explain the method for learning and extracting named entity. In section 4.3, we will report the experiments of applying our method to the CRL named entity data, then we will discuss the results and limitation of our method.

4.1 Japanese Named Entity Extraction in IREX

Information Retrieval and Extraction Exercise (IREX) was held in 1999 [14], and fifteen systems participated in the formal run of the Japanese NE exercise. Table 4.1 shows the definition of 8 categories of named entity and their examples provided by IREX.

We employ the definitions of named entity and use the data providing by IREX for comparing our method with some previous studies.

Table 4.1. The definition of Japanese named entity in IREX and their examples.

Type of Named Entity	Example
ARTIFACT	ノーベル文学賞
DATE	五月五日
LOCATION	日本, 韓国
MONEY	２０００万ドル
ORGANIZATION	社会党
PERCENT	二〇%, 三割
PERSON	村山富市
TIME	午前五時

4.2 Learning and Extracting Named Entity

4.2.1 Features

Whether a word is inside a named entity or not, it depends on the context of the word as well as the word itself. For example, “野村証券”, which consists of two words; “野村” and “証券”, is classified into ORGANIZATION. However, the context in which the word “野村” occurs alone, it is assigned to PERSON. “証券” which follows “野村” in “野村証券” is a *key* to classify “野村証券” into ORGANIZATION. Therefore, like much of the previous research, we use such contextual information as features.

Contextual information c can be written as a set of triplets, a triplet defines as (p,t,v), where p represents the position from current position i, t presents the type of a feature, and v is a value of the feature t. If $p = 0$, it shows that the position equals to the current position i. If $p < 0$, it is a preceding p-th position from current position. If $p > 0$, it is a seceding p-th position from i. A context length can be defined by the minimum value of p and the maximum value of p. In our experiments, the context length is fixed as 5 words i.e., from -2 to 2 of p similar to Uchimoto’s study.

Table 4.2 summarizes types of features and their values.

The feature of “part-of-speech” gives important clues for named entity extraction, since named entity is a kind of noun phrase. The values of the feature part-of-speech are those produced by a morphological analyser.

The “character type” is one of the important clues for discriminating whether a

type of feature	value
part-of-speech	Noun, Verb, Adverb etc.
character type	*Kanji*, *Hiragana*, *Katakana*, Numeric, Symbol
word	surface form of the word itself
previous NE-tag	B-ARTIFACT,I-ARTIFACT,..., I-MONEY, O

Table 4.2. Summary of features and their value

			i		
p	-2	-1	0	+1	+2
words	大統領	は	五	日	午前
part-of-speech	Noun	Particle	Noun	Noun	Noun
character types	Kanji	Hiragana	Kanji,Numeric	Kanji	Kanji
previous NE tags	O	O	B-DATE	I-DATE	B-TIME

Figure 4.1. An example of contextual information.

word is named entity or not in Japanese. The values of the feature are classified into five categories in table 4.2. And we use all types of characters in a word as features.

The “word” is surface form of the word in the sentence, and the value is a string of the word itself.

Some named entities consist of several words, for example “和歌山県高野山” as LOCATION which cosists of three word: “和歌山”, “県” and “高野山”. We can estimate “和歌山県高野山” as LOCATION, if we know that both “和歌山”and “県” are words which show LOCATION. Therefore, we use previous NE-tags as features .

Figure 4.1 shows an example of contextual information for the *i*-th words “五” in a sentence “··· 大統領は五日午前 ···”. In this example, the *i*-th contextual infomation is the following set of triplets: {(-2, word, 大統領), (-1, word, は), (-0, word, 五), (+1, word, 日), (+2, word, 午前), (-2, pos, Noun), (-1, pos, Particle), (-0, pos, Noun), (+1, pos, Noun), (+2, pos, Noun), (-2, char.type, Kanji), (-1, char.type, Hiragana), (0, char.type, Kanji), (0, char.type, Numeric), (+1, char.type, Kanji), (+2, char.type, Kanji), (-2, NE-tag, O), (-1, NE-tag, O)}.

Suppose that n kinds of triplets $c_k(k < n)$ for all of the training data in a context length. Consider the feature space X in which each axis corresponds to a triplet c_k, The feature vector corresponding to i-th word can be written as follow:
$\mathbf{x}_i = (b(c_0), b(c_1), \cdots, b(c_k), \cdots, b(c_n))$
where $b(c_k) = 1$ if i-th context includes the contextual information c_k, otherwise it is 0. Suppose further that y_i is the NE-tag of word w_i. An example corresponding to w_i can be represented as a pair : $(\mathbf{x}_i, y_i)$. Thus we can learn the classification rules for each named entity category using SVMs with the pairwise method for multi-class problems.

4.2.2 Extracting Named Entities

Named entity extraction is to estimate NE-tag for each word in a sentence by using a learned classifier. Thus, we first divide each sentence into some words and put part-of-speech tags by using morphological analyser. In this process, every contextual information of a word is assigned in advance except for NE-tags. However in our algorithm, features of NE-tags are assigned to words in a deterministic manner, i.e., they are estimated in the process of extraction. Such information is dynamically usable through the NE-tagging process.

In this chapter, **forward extraction** is to assign NE-tags from the beginning of a sentence, **backward extraction** is to assign NE-tags from the end of a sentence. Note that preceding NE-tags using forward extraction are different from those of using backward extraction, since "preceding" in forward extraction is the side of the beginning of the sentence, and that in backward extraction is the side of the end of the sentence.

4.3 Experiments

4.3.1 Data and Setting

The CRL (Communications Research Lab.) named entity data is a large-scale annotated corpus from Mainichi Simbun news articles . The data contains about 11,000 sentences in 1,174 articles, and the total number of named entities is 19,262. Table 4.3 illustrates the frequency distribution of each named entity categories and the ratio of the NEs in the data.

Table 4.3. The frequency distribution of named entity in the CRL data.

Type of NE	frequency	(%)
ARTIFACT	747	(3.88)
DATE	3,567	(18.52)
LOCATION	5,463	(28.36)
MONEY	390	(2.02)
OPTIONAL	585	(3.04)
ORGANIZATION	3,676	(19.08)
PERCENT	492	(2.55)
PERSON	3,840	(19.94)
TIME	502	(2.61)
Total	19,262	

"OPTIONAL" tag was assigned to a case in which a human annotator cannot decide uniquely the category for a named entity. When only the beginning and the end of NEs are estimated correctly, it is judged to be correct. Therefore, we also use this evaluation measure.

Parameters of SVMs

We have to decide two parameters for running SVMs,

- type of kernel function and its coefficients
  - polynomial function: $(A\mathbf{x}_i \cdot \mathbf{x} + j + B)^d$
  - RBF function:
  - Sigmoid function
- Soft margin parameter

In our experiments, we use d-th degree of polynomial function $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$ as the kernel function, and soft margin parameter C is to fixed 0.1 for faster learning time.

Evaluation measure

We use the standard evaluation measures which are widely used in IR research fields, i.e., Precision, Recall and F-measure:

- $precision = \frac{\text{the number of correctly extracted NEs}}{\text{the number of extracted NEs by the system}}$
- $recall = \frac{\text{the number of correctly extracted NEs}}{\text{the number of NEs annotated by a human}}$
- $F - measure = \frac{2Precision \times Recall}{Precision + Recall}$

We evaluate our method by using 5-fold cross validation for the CRL named entity data set. We do not use the formal run test data, since only the participants of the IREX workshop allows to use the formal run test data.

4.3.2 Results

We first show the extraction accuracy of the following 4 kinds of feature spaces and investigate which features contribute the extraction accuracy for each named entity category.
(1) word and major categories of part-of-speech
(2) word, major and minor categories of part-of-speech
(3) word, major categories of part-of-speech, and character types
(4) word, major and minor categories of part-of-speech, and character types

In this experiment, we fixed the setting as the following: representation of named entity is the IOB2 model, the direction of extraction is forward, the kernel function is polynomial function $(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$. Table 4.4 summarizes the results.

Table 4.4 shows that the best results is 82.0 of F-value in feature space (4), even the number of dimensions of the feature space become to be about 94,400. Part-of-speech minor categories contribute the accuracy for PERSON, ORGANIZATION and LOCATION categories, since some minor tags include directory clues like Noun-Pronoun-Person(名詞-固有名詞-人名).

This demonstrates that the performance of NE, especially to extract ORGANIZATION, PERSON and LOCATION categories depends on dictionaries(proper-nouns) prepared in morphological analysis systems.

Table 4.4. The extraction accuracy for each feature set.

	$F_{\beta=1}$ value			
feature type	(1)	(2)	(3)	(4)
ARTIFACT	46.5	45.8	**47.0**	46.8
DATE	89.4	91.1	90.7	**91.9**
LOCATION	76.3	81.6	78.1	**82.3**
MONEY	90.5	91.3	**91.5**	91.0
ORGANIZATION	64.2	**75.8**	67.8	75.7
PERCENT	91.2	**91.3**	**91.3**	91.1
PERSON	69.6	83.7	72.9	**85.0**
TIME	89.0	88.9	**89.3**	88.7
Total	74.8	81.5	76.8	**82.3**

There are no significant improvements for DATE and PERCENT by including part-of-speech minor tags as features. On the other hand, the accuracy of ARTIFACT, MONEY and TIME decreased. These results shows that it is necessary to select optimal feature space for each named entity category.

Named entity representation and direction of extraction

We investigate the extraction accuracy for each NE-representation models and each direction of extraction.

Table 4.5 shows that the best result is the IOB2 model and backward extraction and it attained 83.2 of F-value. For most of NE-representation models, the results of backward extraction is better than those of forward extraction, except for the IOB1 model.

Effectivness of Learning with Combination of Features

Table 4.6 illustrates the extraction accuracy using different number of degree of polynomial kernel function.

The best result is in $d = 2$, i.e., learning with combination of any two features. All of the results in $d \geq 2$ are superior to any one of $d = 1$. This means that learning with

Table 4.5. Extraction accuracy and combination of NE-representation models and direction of extraction

	$F_{\beta=1}$ value									
Chunk-tag	IOB1		IOB2		IOE1		IOE2		SE	
extraction direction	for.	back.	for.	back.	for.	back.	for.	back.	for.	back
ARTIFACT	47.1	44.5	46.8	47.1	47.2	**48.3**	44.8	46.3	44.1	47.1
DATE	92.0	91.9	91.9	92.2	91.9	**92.6**	91.3	92.4	91.0	91.8
LOCATION	82.6	82.3	82.3	82.5	82.5	**82.8**	82.1	81.9	81.7	81.8
MONEY	90.7	93.9	91.0	**94.3**	90.7	93.9	90.6	94.1	90.9	94.0
ORGANI.	76.1	78.0	75.7	**79.0**	76.3	76.7	75.2	75.7	74.7	77.9
PERCENT	89.2	**94.8**	91.1	94.2	89.1	92.5	88.6	91.8	91.4	93.7
PERSON	85.0	85.5	85.0	**86.3**	85.1	85.5	84.9	85.2	84.8	85.7
TIME	87.1	89.0	88.7	**89.2**	87.1	86.5	84.3	86.3	88.7	88.7
Total	82.4	82.1	82.3	**83.2**	82.4	82.9	81.2	81.7	81.2	82.4

combination of features is an important factor for named entity extraction.

Determistic extraction

Our method has somewhat incremental nature, i.e., it estimates deterministically NE-tag and use the results to analyse later sequence of a sentence. Therefore, it is important to ensure how we can estimate earlier NE-tag without false.

We extract approximately the optimal NE-tag sequence for the whole sentence by using beam search algorithm. A degree of priority for each named entity category defines as the number of vote using pairwise method. Then we compare with our deterministic extraction method.

Table 4.7 illustrates the results, here *w* represents the beam width, and processing time shows the ratio for the deterministic method.

The results show that searching an optimal NE-tag sequence using beam search does not improve extraction performance. This demonstrates that it is difficult to define the degree of priority by using outputs of SVMs. SVMs is a binary classifier, not a

Table 4.6. The number of degree of polynomial kernel function and the extraction accuracy.

	$F_{\beta=1}$ measure			
d	1	2	3	4
ARTIFACT	28.7	**47.1**	46.3	44.3
DATE	89.1	**92.2**	91.2	90.2
LOCATION	78.4	**82.5**	81.9	81.5
MONEY	94.1	**94.3**	93.7	93.5
ORGANIZATION	71.5	**79.0**	77.5	76.7
PERCENT	92.8	**94.2**	93.1	92.8
PERSON	82.0	**86.3**	85.6	85.2
TIME	86.5	**89.2**	87.4	86.9
Total	78.5	**83.2**	82.3	81.7

probability estimator. Therefore the number of votes in the pairwise method dose not represent preciously the degree of priority which a word is classified into an NE-tag.

4.3.3 Discussion

Failure Analysis

One of the causes of the error is that named entity boundary is often different from the word segmentation of the output by morphological analyser. For example, “高山町長” is segmented into two words: “高山” and “町長” by the morphological analyser ‘ChaSen’. However “高山町” is a LOCATION defined in IREX. We cannot extract this named entity in that case, because our method assigns NE-tag to a word segmented by the morphological analyser.

We investigate the effect of this problem by using the following two kinds of preprocessing, then compare those with our method.

- preprocessing (A): In the training data, if the word segmentation of the morphological analyser is different from the boundary of NE, the word in the test data

Table 4.7. The beam width and the extraction accuracy.

	F-measure			
w	1	3	5	10
ARTIFACT	47.1	**47.3**	46.3	46.6
DATE	92.2	**92.3**	**92.3**	**92.3**
LOCATION	82.5	82.6	82.6	**82.7**
MONEY	**94.3**	**94.3**	**94.3**	**94.3**
ORGANIZATION	**79.0**	78.8	78.8	78.7
PERCENT	94.2	94.2	94.3	**94.5**
PERSON	**86.3**	86.0	86.0	85.9
TIME	**89.2**	89.1	88.8	89.0
Total	**83.2**	**83.2**	83.1	**83.2**
Ratio of Time	1	3.42	5.52	10.73

is subdivided into two parts. For example, a word such as '町長' is subdivided into '町' and '長'.

- preprocess (B): For all of NEs in the test data, word segmentation is performed beforehand according to the annotated information.

Table 4.8 shows that the accuracy of preprocessing (A) is lower than those of non-preporcessing methods. One possible reason behind this is that the segmentation of preprocessing (A). We recall that in the preprocessing (A), a word such as '町長' is subdivided into '町' and '長' if the result of morphological analyser is different from the result of NE. As a result, '町長' which should be a meaningful word itself are subdivided into '町' and '長'. This yields a high recall rate, but low precision rate.

For further improvement, it is necessary to tag part-of-speech correctly and to segment words correctly. The results of the (B) can be view as the upper bound accuracy of our method in which the morphological analyser produces the complete word segmentation and the part-of-speech tags.

Table 4.8. Effects of word segmentation by morphological analyser.

	F-measure		
	our method	(A)	(B)
ARTIFACT	47.1	**48.3**	48.2
DATE	92.2	92.7	**92.8**
LOCATION	82.5	82.6	**87.8**
MONEY	94.3	94.3	**94.5**
ORGANIZATION	79.0	77.3	**81.4**
PERCENT	94.2	91.4	**97.1**
PERSON	86.3	85.7	**86.4**
TIME	89.2	85.2	**90.1**
Total	83.2	83.0	**85.9**
Precision (%)	86.4	85.1	88.1
Recall (%)	80.3	81.0	83.9

Comparison with Related Work

We compare our method with two related work for demonstrating the effectiveness of our method, although it is impossible to compare our method with their methods. This is because the IREX test data set is no more available. The two related works are Uchimoto's [17] and Sassano's [22] studies. Uchimoto's study is the best results using machine learning techniques of the participants in the IREX workshop [14]. Table 4.9 illustrates the summary of these results.

Table 4.10 shows that the extraction accuracy for each set of 5-fold cross-validation in our method, since we can not use IREX test data set. As we introduced in Chapter 2, Uchimoto's method uses a maximum entropy model and transformation rules for Japanese named entity extraction. Transformation rules are automatically acquired from the training data using the difference between word segmentation by morphological analyser and correct named entity boundary. Table4.9 shows that the results of their method attained 80.17 at F-value for GENERAL data in the IREX.

Sassano et al. uses *variable length model* for Japanese named entity extraction. Their method restricts the model to explicitly considering the cases of NEs of the length

Table 4.9. Comparison with related work

	Uchimoto	Sassano	our method
features	word, part-of-speech, character types		
morphological analyser	JUMAN	BREAKFAST	ChaSen
length of the context (num. of word)	±2	variable length	±2
Chunk representation	Start/End	Start/End	IOB2
word segmentation	transformation rule	nothing	nothing
training data	CRL NE data dry-run data	CRL NE data	CRL NE data
size(sentence)	about 12,000	about 11,000	about 8,800
Learning Algorithm	Maximum Entropy	Maximum Entropy	SVMs
F-value	80.17	82.8	**83.2**

Table 4.10. The accuracy for each data set in 5-fold cross-validation.

Total	1	2	3	4	5
83.2	82.8	83.3	**84.8**	81.6	83.6

up to three words and only implicitly considering those longer than three morphemes. It also restricts the model by considering two words in both left and right contexts of the NE. They use Start/End and IOB2 schemes for NE-representation. Like Uchimoto's method, they use three kinds of features: (i) a pair of word and part-of-speech tag, (ii) a pair of character types and part-of-speech, and (iii) part-of-speech tags. The character type consists of 6 elements: *Hiragana*, *Kanji*, *Katakana*, *Numeric*, *Alphabet*, *Symbol*, and their combination. The results show that the success rate of 82.8 of F-value.

Table 4.10 shows that the performance of our method is at least 81.6 which is similar to Uchimoto's result. The result of Sassano's method outperforms our method in some conditions. One possible reason is that they used *variable length model* which implicitly considering the length of named entity longer than three morphemes. For future work, it is necessary to incorporate this technique into our method.

Uchimoto and Sassano's method use a simple feature selection in the training data. For example, Uchimoto's method only uses words in the data with more than two occurrences. This means that it may deteriorate the performance on the training data itself. Actually, Uchimoto's study reports on the accuracy of about 85.0 at F-value for GENERAL domain training data in IREX. On the other hand, our method does not select features using any heuristics. The accuracy for the training data of our method is 99.7 despite the fact that our method dose not select features using any heuristics. From this point-of-view, our method is more effective than their method.

SVMs can learn with exhaustive combination of any features by virtue of polynomial kernel functions without explosively increasing computational costs. Furthermore, maximum margin strategy of SVMs makes it possible to avoid over-fitting even in a high dimensional feature space mapped implicitly by a polynomial kernel function.

4.4 Summary

In this chapter, we reported an experimental study for Japanese named entity extraction using Support Vector Machines. We also showed how SVMs can be used effectively to extract named entity using the CRL named entity data with the IREX named entity task.

We first described the definitions of named entity in IREX and then presented a method to extract them using SVMs. We reported the results of experiments. The re-

sults of cross validation showed that our method attained more than 83 of F-value. The results also show that in Japanese named entity extraction, backward extraction method is more effective than one of forward extraction. Further, throughout the experiments, we have found that a learning model concerning combination of features is necessary for extracting, especially combination of any two features is the best performance.

Chapter 5

Partial Parsing in English

In this chapter, we propose a method for partial parsing using SVMs. We first described our motivation: why partial parsing is needed for language analysis by comparing it with a parser which constructs a complete parse tree, and define some notation. Next, we explain our parsing algorithm and how to apply SVMs to learn for partial parsing. Finally, we report on our evaluations using Penn TreeBank.

5.1 Why Partial Parsing is needed for Document Processing?

Natural language parsing for language processing is to take a sentence as an input and returns syntactic representation as the output. In this process, we encounter a number of syntactic ambiguities, such as noun phrase ambiguities, prepositional phrase ambiguities and the scope of coordinate conjunction ambiguities and so on. Therefore, it is not easy to produce the correct parse result automatically. Sentence (1) shows one of the typical ambiguities in English sentences: PP attachment.

(1) I saw a girl with a small telescope.

Difficulties in resolving PP attachment ambiguity arise because of the Polyphemus usage of prepositions. In (1), there are at least two interpretations, and the results using conventional parsing techniques are illustrated in Figure 5.1 and 5.2.

In Figures 5.1 and 5.2, boxes at the bottom denote words with part-of-speech information which forms a sentence. Figure 5.1 shows that a phase ‘with a small tele-

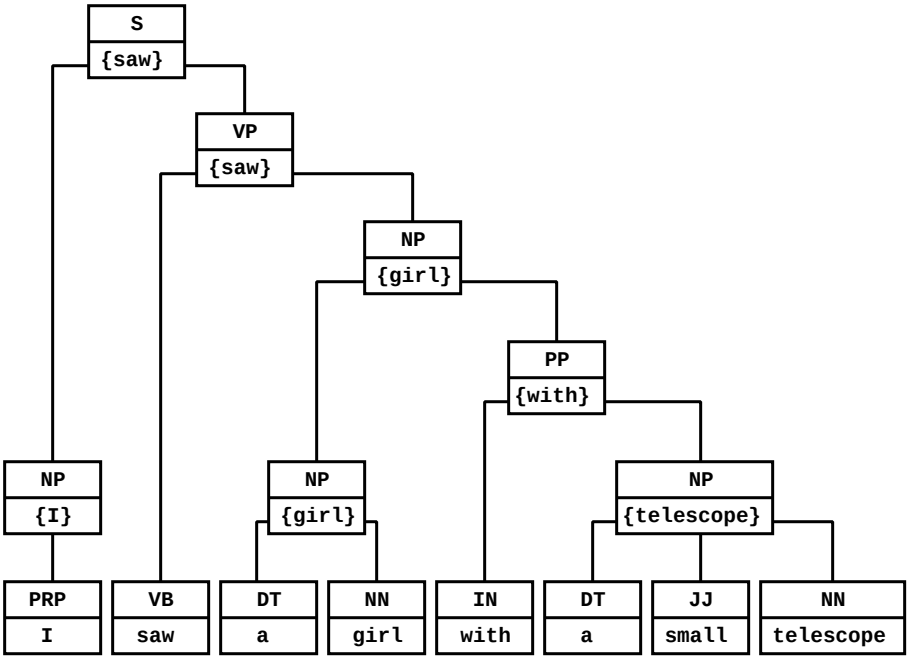

Figure 5.1. A parse tree for ’I saw a girl with a small telescope’.

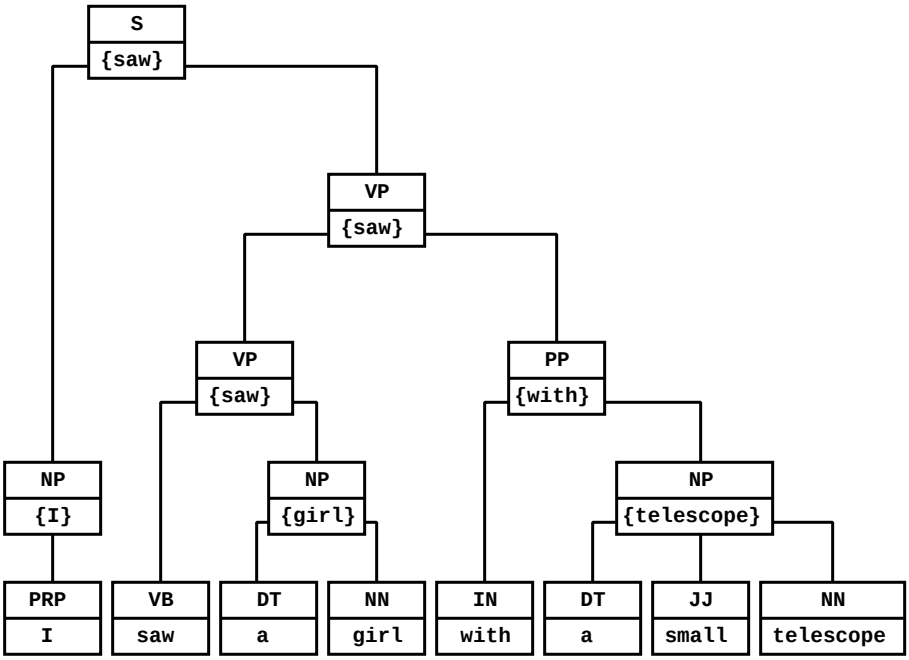

Figure 5.2. Other parse tree for ’I saw a girl with a small telescope’.

scope' modifies the noun word 'girl', whereas Figure 5.3 shows another interpretation: a phrase 'with a small telescope' modifies the main verb 'saw'. Syntactically, a prepositional phrase attaches to a noun phrase('a girl') slightly more often than to a verb('saw'). However, the difference is small and the real interpretation is decided using the *context* in which a preposition is used.

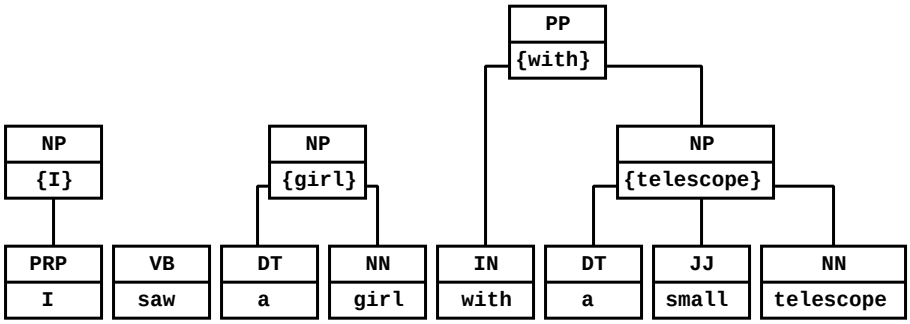

Figure 5.3. Partial parse tree for 'I saw a girl with a small telescope'.

One significant feature of partial parsing technique is to give an interpretation partially in the form of parse sub-tree which is not contain ambiguities. For instance, the result of (1) using partial parsing is illustrated in Figure 5.3, instead of Figures 5.1 and 5.2.

In this chapter, a *constituent* corresponding to a node in a tree, is represented as a pair of a phrase label and a head word:

label: label denotes a category of a constituent such as NP, VP and so on. In particular, the label constituent at a terminal node, i.e. a leaf, refers to a part-of-speech tag.

head : head refers to the head word of the constituent.

The notation of a constituent is 'label-{head}', especially, a terminal node can be written as 'I/PRP'. A sub-tree(or tree) can be written in a bracketed form. For example, a noun phrase 'a girl' in figure 5.1 can be written as follows:

(NP-{girl} (a /DT girl /NN))

5.2 Parsing Algorithm

Our parser employs a deterministic bottom-up algorithm, that constructs labeled syntactic binary trees using three kinds of *parsing actions*: **shift**, **unary:X**, and **binary:X,H**. Here, X represents a label, H denotes the head of the tree (i.e. left or right).

These parsing actions constitute a *model* which is learned by SVMs, and a sentence is parsed by repeatedly executing the action for the current constituent. Let c_i be the *i*-th constituent in a sentence. Three actions, **shift**, **unary:X**, and **binary:X,H** are described as follows.

- **unary:X**
  unary:X means to construct a new constituent labeled X from the current constituent c_i, and the new constituent X is inserted into a current constituent. Figure 5.4 shows the action unary:X.

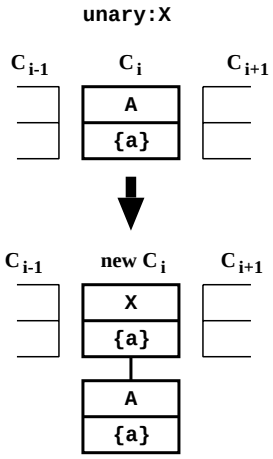

Figure 5.4. An example of the action 'unary:X'

  Figure 5.4illustrates the action unary:X, i.e. unary:X is inserted into a current constituent, 'a', where 'a' refers to the head word of the current constituent.

- **binary:X,H**
  binary:X,H is to construct a new constituent labeled X from the current and succeeding constituents, and 'X' is inserted into the current constituent in two ways.

If H is to ‘Left’, then the head of the new constituent is c_i, otherwise it is c_{i+1}. The left hand side of Figure 5.5 illustrate **binary:X,Left** action, and the right hand side shows the **binary:X,Right** action.

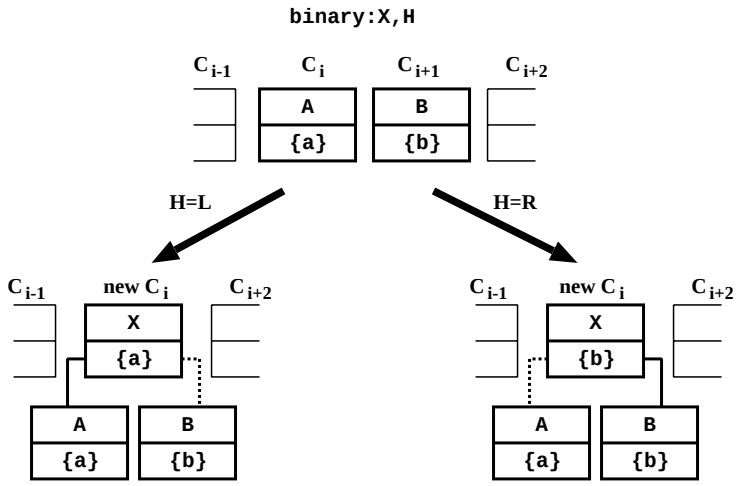

Figure 5.5. An example of the action ’binary:X,H’

- **shift**

  shift is to move the current position *i* to the next $i+1$ position, i.e. no construction of a new constituent.

Next, we describe an overview of a parsing mechanism step by step, especially the procedure to construct a parse tree in a bottom-up style using these actions. Initially, the input sentence *S* is given to the parser a part-of-speech tag is assigned to each of the words as shown in Figure 5.7. In Figure 5.7, down arrow $\downarrow$ indicates the current position, i.e, ‘I/PRP’, in this case.
As shown in Figure 5.8, for the constituent of current position, i.e., ‘I/PRP’, if the model outputs the action ’unary:NP’, the parser executes the action, creating a new constituent which is labeled by ’X’, and constructs a unary tree by inserting the new box, ‘NP-{I}’, into the current constituent, ‘I/PRP’. As the head word of the new constituent is ‘{I}’, the constituent at the current position is replaced by the created new constituent ’NP-{I}’.

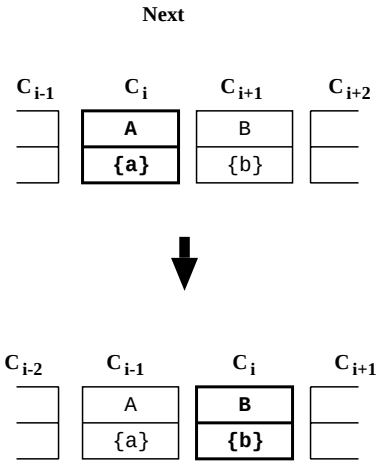

Figure 5.6. An example of the action ’shift’

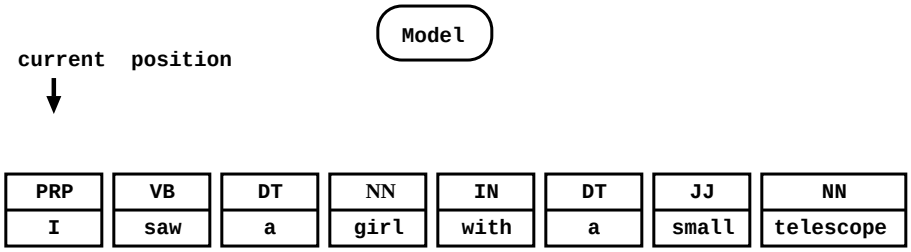

Figure 5.7. Initial state

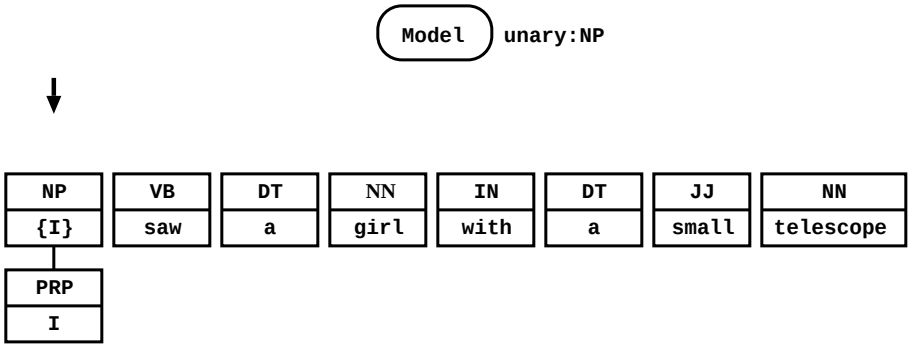

Figure 5.8. After execution of ’unary:NP’

The next step is that the parser recursively inquires to the model which action should take at the same position (Figure 5.9). If the model answers the action ‘shift’, the parser moves the current position i to the next $i+1$ position. Figure 5.14 illustrates the action.

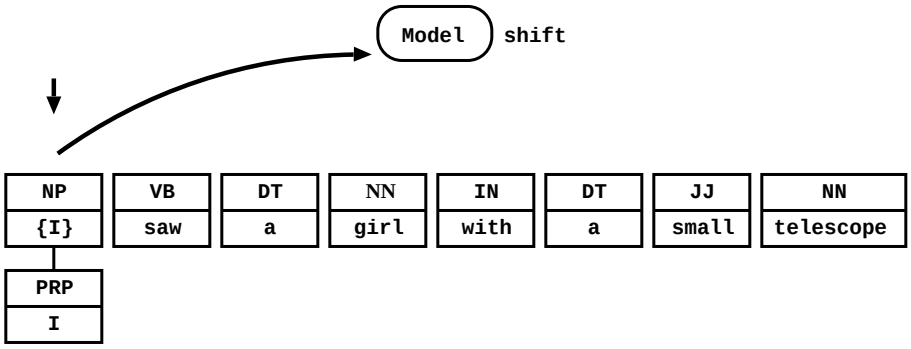

Figure 5.9. The model answers to the action ’shift’

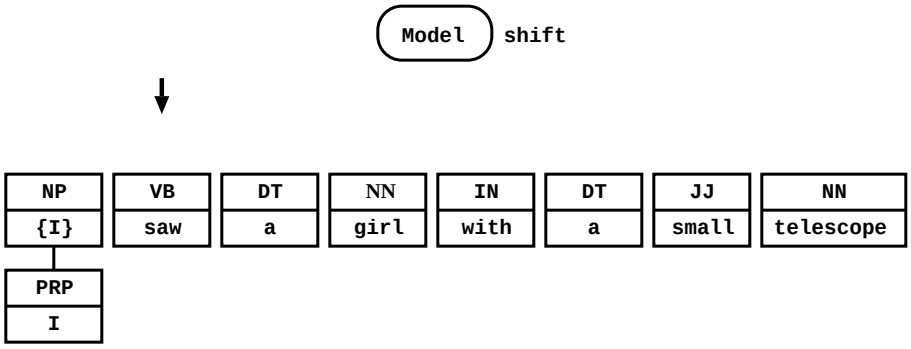

Figure 5.10. Execution of ’shift’

In a similar way shown in Figure 5.11, the model answers the action ‘shift’ for the constituent, ‘saw’, and the parser moves the current position to the next position.

The next step is that the parser executes the action ‘binary:NP,R’ according to answer from the model. The action **binary:X,H** constructs a new constituent labeled X using the current and next constituents, and ‘X’ is inserted into the current constituent in two steps according to the value of ‘H’. In this case, the parser creates a new constituent of label ‘NP’, and inserts this between a current position ‘a/DT’, and the

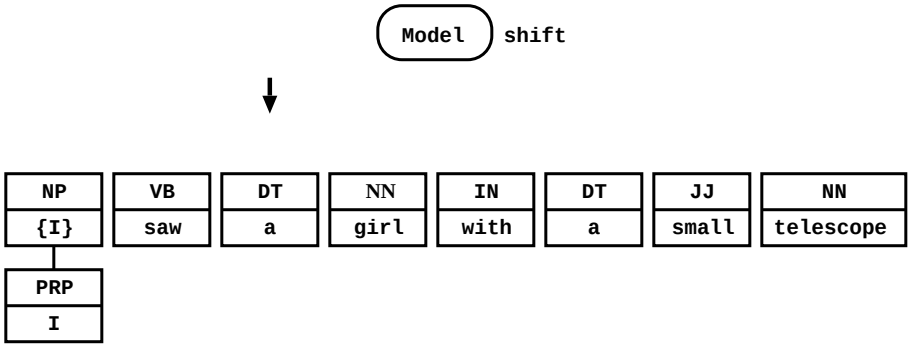

Figure 5.11. Execution of the action ’shift’

next position ‘girl/NN’. ‘girl’ is selected as the head, since ‘H’, in this case, equals to ‘Right’. The current position becomes the new constituent, ‘NP-{girl}’. Figure 5.12 summarizes these steps.

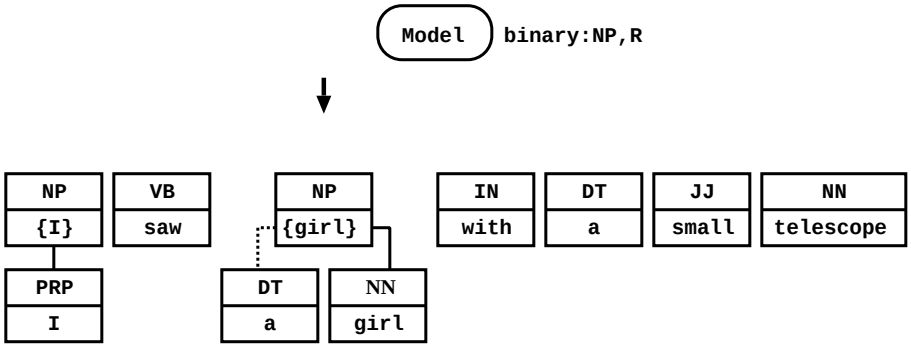

Figure 5.12. Execution of action ’binary:NP,R’

As shown in Figures 5.13 and 5.14, the procedure is repeated until the current position moves on the last constituent of the sentence. Then the procedure is repeated from the beginning of the sentence until any new reduction action is applicable.

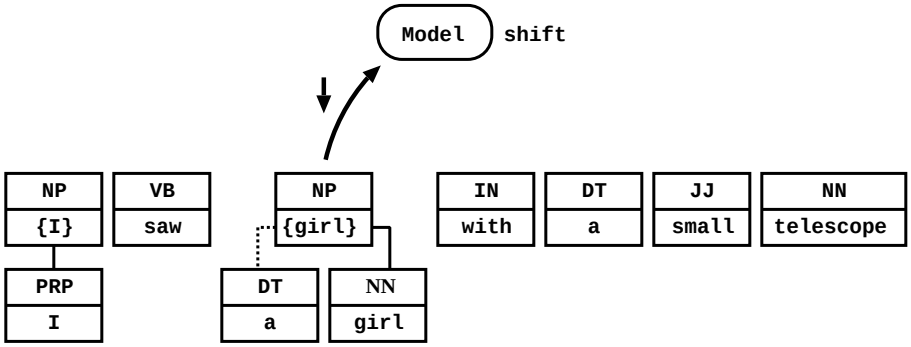

Figure 5.13. The model answers to the action ’shift’

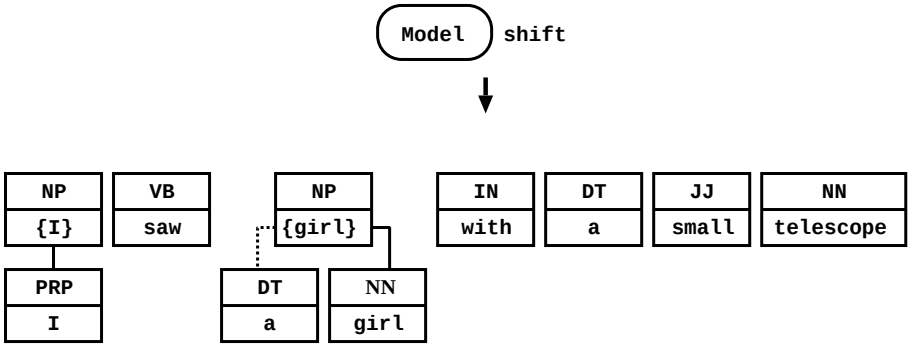

Figure 5.14. Execution of the action ’shift’

5.3 Applying SVMs to General Parsing

5.3.1 Converting N-ary tree into binary tree using head rules

The parsing algorithm in the preceding section constructs a binary tree for the input sentence in a bottom-up manner.

The trees used for learning, therefore, should be binary. However, most of previous researches are based on the Penn Treebank, which uses N-ary tree representation. Thus, the first step to apply SVMs to parsing is to convert N-ary trees into binary trees. We use Penn treebank parse trees and convert them into binary using head rules.

Table 5.1 shows a sample of NP head rule. Other rules are shown in Appendix B. The first column is the label of the mother, and the second shows the starting position in searching for the head. Starting position has either of two values, 'left' and 'right'. 'Left' indicates that the search should go from the first child of the children, and 'right' shows that the search should go from the last. Third column shows the priority order.

Table 5.1. NP head rule

label	starting position	priority
NP	right	(POS, NN, NNP, NNPS, NNS), NX, JJR, CD, JJ, JJS, RB, QP, NP

Figure 5.15 shows an example of converting an N-ary tree to a binary tree using the head rule. 'H' in Figure 5.15 is the head constituent of the N-ary tree 'X'.

Conversion starts from the left of the head constituent 'H', and each node is translated into binary tree. Each node of a binary tree is called 'intermediate node', and its label 'X' represents the intermediate node of 'X'. The head of the intermediate node, e.g., 'X' is obtained by applying head rules. This process is also used to convert binary tree into N-ary tree.

Figure 5.16 illustrates the binary tree that is obtained by converting the N-ary tree of Figure 5.1. In the next section, we present a method to learn syntactic rules by applying SVMs.

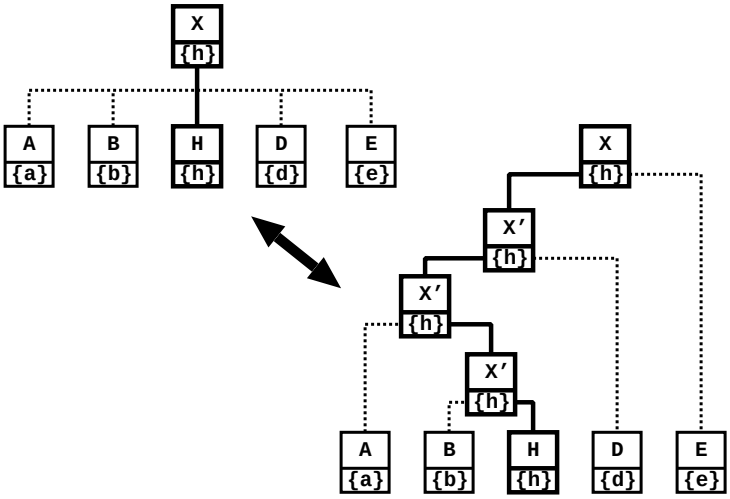

Figure 5.15. An example of converting N-ary tree to binary tree.

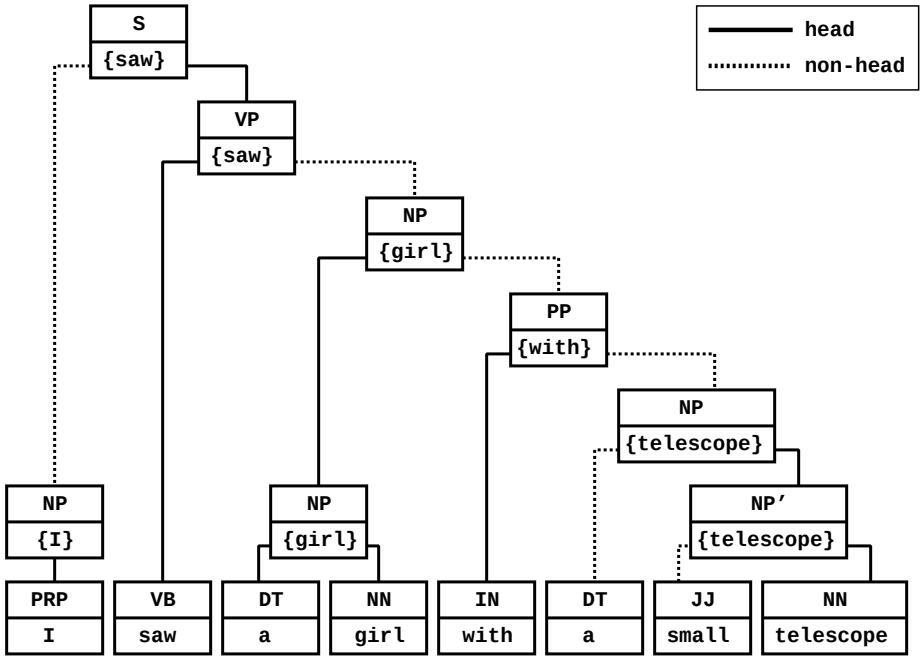

Figure 5.16. An example of the binary tree for the sentence (1): "I saw a girl with a small telescope."

5.3.2 Learning Parsing Actions

As we described in the previous sections, our parser constructs labeled syntactic binary trees using three kinds of parsing actions, instead of using a paticular grammar rules. Therefore, the model needs to answer the appropriate action in the context. Thus the learning for the parsing is to to learn which 'actions' should be taken in a particular context. Each training example corresponds to each action in parsing process, and is obtained by parsing training sentences itself with regarding the model as parsed tree corresponding the sentences. This is one of the classification task that classifies an arbitrary constituent or two adjacent constituent into the parsing action. We can apply SVMs to the task.

For learning rules for classifying into parsing action, we use contextual information as features. Contextual information c can be written as a set of triplets similar to that in Chapter 4. A triplet can be written as (p,t,v), where p represents the position from current position i, t presents the type of a feature as follows, and v is a value of the feature t.

- Default feature
  - *constituent-label*: the label of the constituent at the context. For example, if the constituent NP-{girl}, the value of '*constituent-label*' feature is NP. In this chapter, the feature is written as 'const-label' for short.
  - *head-word*: the head word of constituent. If the constituent NP-{girl}, the 'girl' is the value of *head-word* feature.
  - *head-pos*: this refers the part-of-speech tag of the head. If 'NP-{girl}', NN is the value of *head-pos* feature.
- Special feature for PP and CC
  - PP non-head information :
    Suppose a sentence, 'I buy a car with money'. This sentence an example of PP attachment, thus, a parsing algorithm constructs two syntactic trees like Figures 5.17 and 5.18. As we mentioned in the previous section, syntactically, prepositional phrases attach to noun phrases slightly more often than to verb, and the tree shown in Figure 5.18 is regarded to be a correct

parse tree. This judgment is made by a noun ‘money’ followed by ‘with’. However, default feature which uses only a head word information can not assigned correctly, since its feature is {with}. We therefore, use *const-label head-word*, and *head-pos* of non-head in prepositional phrase as features. These three special features for PP are distinguished from those of default features.

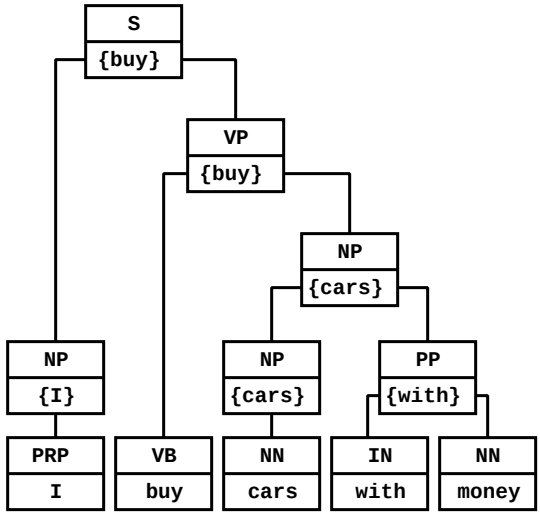

Figure 5.17. An unlikely parse tree for the sample sentence “I buy cars with money ”.

– CC : CC is one of most important clue whether a action ’binary X,H’ should take or not.

  In head rules, CC which followed by NP is not head word. Therefore, it is necessary to distinct whether or not NP accompanies with CC. We use intermediate nodes which accompanies with CC as features.

  Figure 5.19 illustrates an example of CC. ‘NP’ has ‘and/CC’. Arrow in Figure 5.19 shows that the intermediate node ‘NP’-{girl} accompanies with ‘and/CC’.

  ‘and/CC’ use as a feature until the sub-tree of NP are completely parsed.

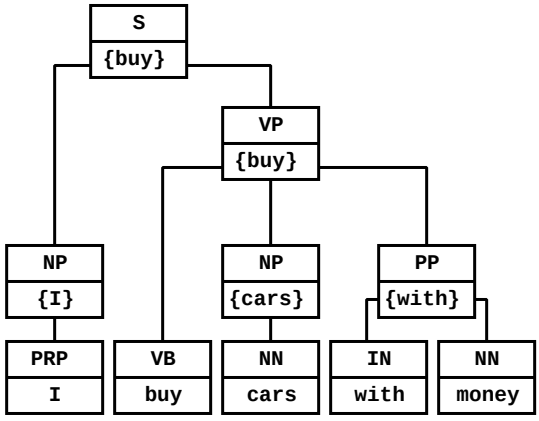

Figure 5.18. A likely parse tree for the sample sentence “I buy cars with money ”.

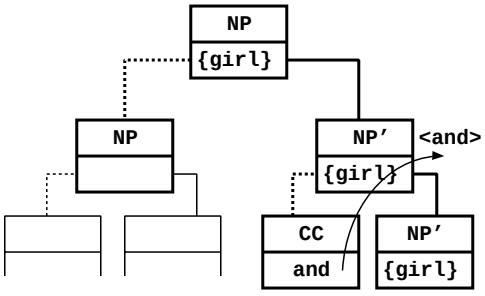

Figure 5.19. Special feature for CC

Suppose that $c_k (k < n)$ is a triplet as contextual information, where n is the number of different triplets for all of the training sentence. Consider the feature space X in which each axis corresponds to a triplet c_k, the feature vector $\mathbf{x}$ can be written as $\mathbf{x} = (b(c_0), b(c_1), \cdots, b(c_k), \cdots, b(c_n))$, where $b(c_k) = 1$ if a context in the parsing step includes the contextual information c_k, otherwise it is 0. Suppose further that y is the parsing action in the context. An example can be represented as a pair : $(y, \mathbf{x})$. Thus we can apply SVMs with one-versus-rest method in Chapter 3 to learn rules classifying a context into the parsing action.

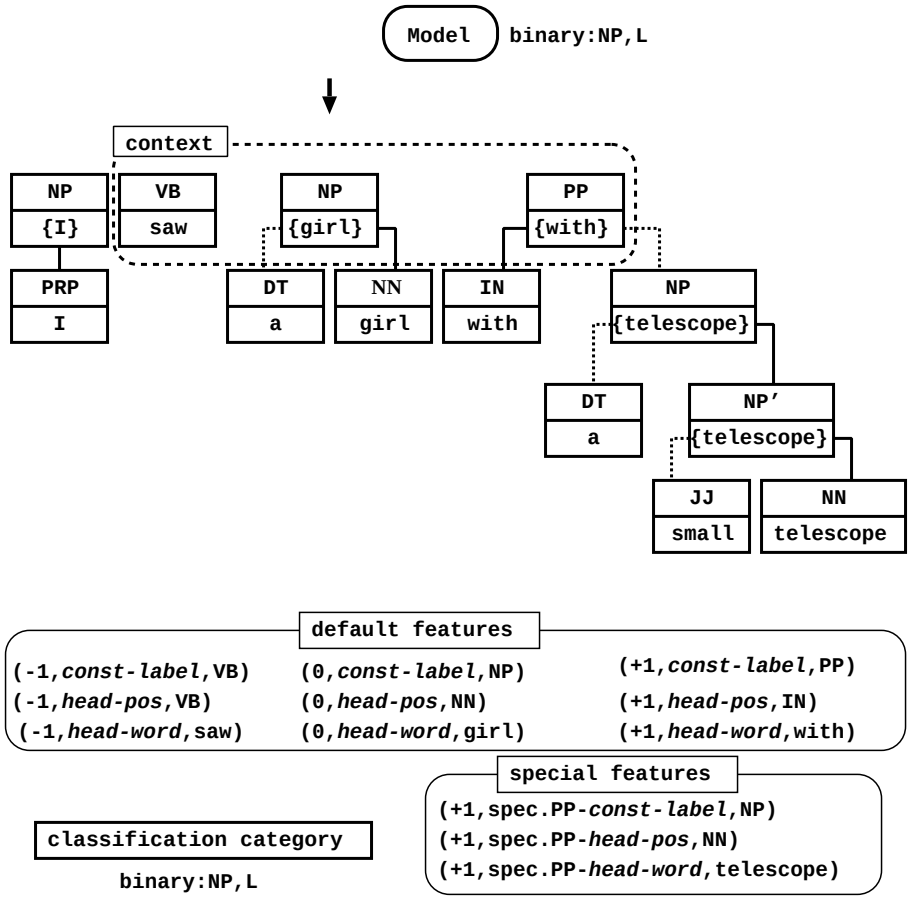

Figure 5.20. A sample of a training example.

Figure 5.20 shows a sample of a training example in a parsing process when each length of left and right context is 1. In the Figure 5.20, contextual information is the

following set of triplets: {(-1, *const-label*, VB), (-1, *head-word*, saw), (-1, *head-pos*, VB), (0, *const-label*, NP), (0, *head-word*, girl), (0, *head-pos*, NN), (1, *const-label*, PP), (1, *head-word*, with), (1, *head-pos*, IN), (1, *spec. PP const-label*, NP), (1, *spec. PP head-word*, telescope), (1, *spec. PP head-pos*, NN)}.

5.3.3 Variable context length model

In the fixed 2 context length, our parser cannot parse some phrase which include the punctuation of ',' or ':'. Figure 5.21 and 5.22 illustrate an example of this problem.

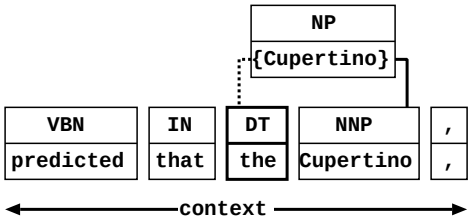

Figure 5.21. The Problem of fixed context length

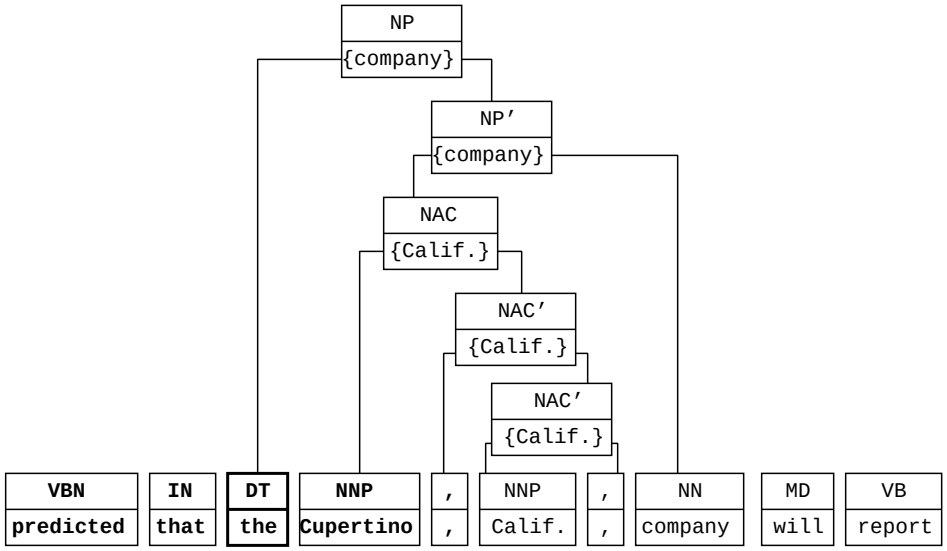

Figure 5.22. The Problem of fixed context length

Figure 5.21 shows the result when we use current position as 'the/DT', and the length of context is 5 (i.e., left and right context length is ± 2 constituents). In this context,it can be interpreted that 'the/DT' and 'Cupetino/NNP' make 'NP-$\{$Cupetino$\}$. The correct interpretation of Figure 5.21 is illustrated in Figure 5.22. We can see that the action which is shown in Figure 5.21 is not correctly interpreted.

To solve the problem, we extend our method, i.e. 1) variable length of context which use right context length so as to be longer than that of left context length, 2) when CC :, ", and " are included in the right context, variable length of context is applied to these features. This model is 'variable context length model'.

5.3.4 Dividing training examples into some parts

As mentioned in the preceding section, a training example corresponds to a pair of parsing action and feature vector as its contextual information.

However, the number of training examples corresponds to the number of parsing actions and become to be a large size when the number of training sentences increases. As a result, it is difficult to use all of the training data because of a large amount of computational cost (actually, when we use 5 context length as features, the number of training example is more than 3,000,000 for about 40,000 sentence of the standard training set in Penn treebank corpus).

For solving this problem, we divide a set of training example into some subsets according to the *const-label* features. Let S be a set of all of the training examples and L be a set of constituent label i.e., $L = \{\mathrm{NP}, \mathrm{VP}, \mathrm{PP}, \cdots l_k\}$. We define a set of example S_{l_p} as follows:

$$S_{l_p} = \{(\mathbf{x}, y) | (0, const-label, l_p) \in \mathbf{x}\}$$

Furthermore, some sets of $S_{l'_p}$ are subdivided into $S_{l'_p, l_q}$ defined as follows:

$$S_{l_p, lq} = \{(\mathbf{x}, y) | (0, const-label, l_p) \in \mathbf{x}, (1, const-label, l_q) \in \mathbf{x}\}$$

Using a simple heuristic, we determine whether S is subdivided or not, i.e., $|S_{l_p, l_q}| \leq 30,000$. Thus multi-class SVMs using the one versus rest method in Chapter 3 is applied to each different set of example independently. Then each classifier $f_{l_p}(\mathbf{x})$ and

$f_{l_p,l_q}(\mathbf{x})$ is constructed by learning from each set of S_{l_p} and S_{l_p,l_q}.

A test example **x'** including triplet $(0, const-label, l'_p)$ and $(1, const-label, l'_q)$ is classified into a parsing action by using the SVMs learned from the example set $S_{l'_p,l'_q}$. If there is no SVMs learned from the example $S_{l'_p,l'_q}$ (that means, a set $S_{l'_p}$ did not subdivide), **x'** is classified into a parsing action by using the SVMs learned from $S_{l'_p}$.

For instance, S_{NP} is the set of examples containing the triplet (0, *const-label*, NP). The $|S_{NP}|$ is more than 500,000 in the standard training set of Penn Treebank Wall Street Journal corpus, and $|S_{NP,IN}|$, $|S_{NP,,}|$, $|S_{NP,VP}|$, $|S_{NP,.}|$, $|S_{NP,PP}|$, $|S_{NP,VBD}|$ are more than 30,000 in the corpus. we divide the S_{NP} into some subsets : $S_{NP,IN}$, $S_{NP,,}$, $S_{NP,VP}$, $S_{NP,.}$, $S_{NP,PP}$, $S_{NP,VBD}$ and each resulting set which S_{NP} is removed from each set: $S_{NP,IN}$, $S_{NP,,}$, $S_{NP,VP}$, $S_{NP,.}$, $S_{NP,PP}$, $S_{NP,VBD}$.

5.4 Experiments

5.4.1 Data and Setting

The data we use in the experiment is from section 2 to 21 and 23 of the Penn Treebank Wall Street Journal corpus. This is a gold standard data set. We use 2-21 for training, and 23 for test data, i.e., we use the same data in our evaluations to make our results comparable with the results by others in the Penn Treebank benchmark evaluations described in Chapter 2.

Most of previous studies use the perfect post-edited output of part-of-speech tagging, or perfect annotation by human intervention. We also use the perfect analysis. Further, previous studies evaluate their method by using two types of data: one is a set of sentences consisting of less than 40 words, and another is a set consisting of all sentences in the target section.

5.4.2 Results

Fixed versus variable context length

We investigate parsing results to compare fixed length with variable length model. Table 5.2 illustrates the parsing results of fixed and variable length model. In Table 5.2, L and R in (L,R) shows that the length of left context and that of right context,

respectively. The result of the variable length model by far outperforms the fixed lenght model.

Table 5.2. Comparison fixed length with variable length model. (size of training = 39,832 sentences)

	(2,2)	
all sentences	fixed	variable
LR(%)	84.2	**87.0**
LP(%)	85.1	**87.9**
CB	1.51	1.18
0CB(%)	54.6	**61.3**
2CB(%)	77.5	**82.7**
length ≤ 40		
LR(%)	85.0	**87.8**
LP(%)	85.8	**88.6**
CB	1.30	1.00
0CB(%)	58.0	**64.5**
2CB(%)	80.3	**85.3**

Table 5.3 illustrates how the length of left and right context influences parsing accuracy.

Table 5.3 shows that the best result is when we use the model (2,3) (i.e., 2 left and 3 right context), and the labeled precision and recall attained 88.1% and 88.9%, respectively. The accuracy of model (1,3), (2,3) and (2,4) were superior to those of the model (1,2) and (2,2). This demonstrates that the parsing accuracy depends on the context length, and the longer the right context contributes the performance. The result of (2,4) was worse than that of (2,3). One reason behind this lies that features which are not effective for parsing are included in the context (2,4). Therefore we need to investigate the optimal context length for improving the accuracy of our parser.

Table 5.3. Context length and parsing acurracy. (size of training = 39,832 sentences)

all sentences	(1,2)	(1,3)	(2,2)	(2,3)	(2,4)
LR(%)	86.7	87.3	87.0	**88.1**	87.6
LP(%)	87.6	88.3	87.9	**88.9**	88.5
CB	1.16	1.05	1.18	1.02	1.05
0CB(%)	60.9	62.5	61.3	**63.5**	62.9
2CB(%)	83.1	84.4	82.7	**85.1**	84.6
length ≤ 40					
LR(%)	87.5	88.2	87.8	**88.8**	88.4
LP(%)	88.3	88.9	88.6	**89.5**	89.0
CB	0.98	0.90	1.00	0.87	0.90
0CB(%)	64.2	65.5	64.5	**66.3**	65.8
2CB(%)	85.7	86.6	85.3	**87.3**	86.7

5.5 Comparison with Related Work

We compared the variable context length method (i.e., left context is 2, right context is 3, respectively) with three related work: Ratnaparkhi [31], Collins [26], and Charniak method [8]. Table 5.4 summarizes our evaluation.

Table 5.4 shows that the result of our method outperforms Ratnaparkhi's and Collin's one, since the accuracy of our method attained 88.1%/88.9% labeled recall/precision, while Ratnaparkhi's method attained 86.3%/87,5% and Collins's parser attained 88.1%/88.3% for all of the test sentences.

Charniak method is superior to our method, this is because his method empolys two path approch: (i)first path is to generate candidate possible parse tree, (ii) then the second is to evaluate these candidates using probablistic generative model based on top down derivation using treebank grammar extracting from Penn Treebank copus. However, the effect of the treebank grammar in the second path strongly depends on the Penn treebank corpus, because the grammar are extracted from the parsed sentence of training data using statistics of context-free rules and some heuristics. Therefore, it is not clear if his approach is effective for different corpora.

Our approach, on the other hand, does not depend on the specific grammar rules or

Table 5.4. Comparison with related work (size of training = 39,832 sentences).

all sentences	our method	Ratna99	Coll00	Char00
LR(%)	88.1	86.3	88.1	**89.6**
LP(%)	88.9	87.5	88.3	**89.5**
CB	1.02	–	1.06	0.88
0CB(%)	63.5	–	64.0	**67.6**
2CB(%)	85.1	–	85.1	**87.7**
length ≤ 40				
LR(%)	88.8	–	88.5	**90.1**
LP(%)	89.5	–	88.7	**90.1**
CB	0.87	–	0.92	0.74
0CB(%)	66.3	–	66.7	**70.1**
2CB(%)	87.3	–	87.1	**89.6**

corpora, since it requires only head rules and a given syntactic trees for learning. This yields the best results among the previous one path approaches, although our parser employs a deterministic bottom-up algorithm with one path mechanism.

5.6 Summary

In this chapter, we reported an experimental study for partial parser using SVMs. We first described our motivation: why partial parsing is needed for language analysis by comparing it with a parser which constructs a complete parse tree. We then explained our parser which constructs a binary parse tree with a deterministic bottom-up algorithm. We regarded each parsing process as a classification task that classifies the context into the action for constructing parse tree, and applied SVMs to learn the classification rules. The result using Penn Tree bank corpus shows that our method outperforms Collins's one: the accuracy of our method attained 88.1%/88.9% labeled precision/recall without a paticular grammar and some semantic information.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we focused on partial language analysis: Japanese named entity extraction and partial parsing in English, and proposed a new method using Support Vector Machines for effective analysis.

Firstly, in Chapter 2, we reported on some studies related to the named entity extraction and parsing with a machine learning approach which are similar to our own. In Chapter 3, we briefly introduce an overview of the SVMs and its theoretical advantage for partial language analysis.

In Chapter 4, we described a method for Japanese named entity extraction using SVMs. The results of cross validation using the IREX data showed that our method attained more than 83 of F-value. Throughout the experiments, we found that (i) in Japanese named entity extraction, backward extraction method is more effective than one of forward extraction, and (ii) a learning model concerning combination of features is necessary for extracting, especially combination of any two features is the best performance.

In Chapter 5, we presented partial parser which constructs a pased binary tree with a bottom-up deterministic algorithm. We regard each parsing process as a classification task which classifies a context into the parsing actions, and learn the classification rules by uisng SVMs. Then we applied our parser to Penn Treebank Wall Street Journal corpus. Our parser attained over 88.1% labeled precision and recall without a particular grammar and some semantic information.

6.2 Future Work

We plan to continue our research in two directions. The first is to investigate for improvement of accuracy in two partial language analysis. In Japaneses named entity extraction, we will incorporate variable context length model into our method. In partial parsing in English, we will extend our parser to two path approach like Charniak's parser.

The seconds is to apply our method to other information retrieval tasks such as text categorization and information extraction system, and to show both effectiveness and robustness of our method.

References

[1] *Text Retrieval Conference (TREC), http://trec.nist.gov/.*

[2] Avrim Blum, Tom Mitchell. Combining Labeled and Unlableled Data with Co-Training. In *Annual Conference on Computational Learning Theory (COLT'98)*, pages 92–100, 1998.

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

[4] David M. Magerman. Statistical decision-tree models for parsing. In *In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, 1995.

[5] E. Black et al. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proceedings of the February 1001 DARPA Speech and Natural Language Workshop*, 1991.

[6] Erik F. Tjong Kim Sang and Jorn Veenstra. Representing text chunks. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, pages 173–179, 1999.

[7] Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence AAAI Press/MIT Press*, pages 598–603, 1997.

[8] Eugene Charniak. A Maximum-Entropy-Inspired Parser. In *Proceedings of the Second Meeting of North American Chapter of Association for Computational Linguistics (NAACL2000)*, pages 132–139, 2000.

[9] Hideki Isozaki. Japanese Named Entity Recognition based on a Simple Rule Generator and Decision Tree Learning. In *Proceedings of Association for Computational Linguistics*, pages 306–313, 2001.

[10] Hirotoshi Taira, Masahiko Haruno. Feature Selection in SVM Text Categorization. In *Proceedings Sixteenth National Conference on Artificial Intelligence /*

Eleventh Conference on Innovative Applications of Artificial Intelligence, pages 480–486, 1999.

[11] Hiroya Takamura and Yuji Matsumoto. Feature Space Restructuring for SVMs with Application to Text Categorization. In *Proceedings of 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 51–57, June 2001.

[12] Hiroya Takamura, Hiroyasu Yamada, Taku Kudoh, Kaoru Yamamoto and Yuji Matsumoto. Ensemble based on Feature Space Restructuring with Application to WSD. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium (NLPRS2001)*, pages 41–48, November 2001.

[13] Hiroyasu Yamada, Taku Kudo, Yuji Matsumoto. Japanese Named Entity Extraction using Support Vector Machine. *Transactions of IPSJ*, 43(1):44–53, 2002.

[14] IREX committee. *Proceedings of the IREX Workshop*, 1999.

[15] Kamal Nigam, Andrew McCallum, Sebastian Thrun, Tom Mitchell. Learning to Classify Text from Labeled and Unlabeled Documents. In *Proceedings of 6th National Conference on Artificial Intelligence / Eleventh Conference on Innovative Applications of Artificial Intelligence*, pages 792–799, 1998.

[16] Karen Jensen and George E. Heidorn. The Fitted Parse: 100% Parsing Capability in a Syntactic Grammar of English. In *Proceedings of First Conference on Applied Natural Language Processing*, pages 93–98, 1983.

[17] Kiyotaka Uchimoto, Masaki Murata, Qing Ma, Hiromi Ozaku, and Hitoshi Isahara. Named Entity Extraction Based on A Maximum Entropy Model ans Transformation Rules. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 326–335, 2000.

[18] Kiyotaka Uchimoto, Qing Ma, Masaki Murata, Hiromi Ozaku, and Hitoshi Isahara. Named Entity Extractin Based on A Maximum Entropy Model and Transformation Rules (in Japanese). In *Journal of Natural Language Processing*, volume 7, pages 63–90, 2000.

[19] Kudo Taku, Yuji Matsumoto. Japanese Dependency Analysis Based on Support Vector Machines. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 18–25, 2000.

[20] Kurohashi Sadao and Nagao Makoto. *Japanese Morphological Analysis System JUMAN version 3.61 Manual*. Kyoto University, 1998. (in Japanese).

[21] Lance A. Ramshaw and Mitchell P. Marcu. Text Chunking Using Transformation-Based Learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94, 1995.

[22] Manabu Sassano, Takehito Utsuro. Named Entity Chunking Techniques and their Evaluation in Japanese Statistical Named Entity Recognition. In *IPSJ SIG NOTES*, number 2000-NL-139, pages 1–8, 2000.

[23] Manabu Sassano, Takehito Utsuro. Named Entity Chunking Techniques in Supervised Learning for Japanese Named Entity Recognition. In *Proceedings of 18th International Conference on Computational Linguistics*, pages 705–711, 2000.

[24] Michael Collins. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, 1996.

[25] Michael Collins. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (jointly with the 8th Conference of the EACL)*, pages 16–23, 1997.

[26] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Ph.D. Dissertation, 1999.

[27] Michael Collins. Discriminative Reranking for Natural Language Parsing. In *Proceedings of 17th International Conference on Machine Learning (ICML2000)*, pages 175–182, 2000.

[28] Mitchell P. Marcus and Beatrice Santorini and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[29] Mitchell P. Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson and Karen Katz and Britta Schasberger. The Penn treebank: Annotating predicate argument structure. In *Proceedings of 1994 Human Language Technology Workshop*, pages 110–115, 1994.

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

[31] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175, 1999.

[32] Robert E. Schapire and Yoram Singer. BoosTexter : A boosting-based system for text categorization. In *Machine Learning*, 1998.

[33] S. Abney and R. Schapire and Y. Singer. Boosting applied to tagging and PP attachment. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[34] SAIC. *Proceedings of the 7th Message Understanding Conference (MUC-7)*, 1998.

[35] Satoshi Sekine and Ralph Grishman and Hiroyuki Shinnou. A Decision Tree Method for Finding and Classifying Names in Japanese Texts. In *Proceedings of the 6th Workshop on Very Large Corpora*, pages 171–178, 1998.

[36] Taku Kudoh and Yuji Matsumoto. Use of Support Vector Learning for Chunk Identification. In *Computational Natural Language Learning (CoNLL-2000)*, pages 142–144, 2000.

[37] Taku Kudoh and Yuji Matsumoto. Chunking with Supprot Vector Machines. In *Proceedings of the Second Meeting of North American Chapter of Association for Computational Linguistics (NAACL)*, pages 192–199, June 2001.

[38] Tetsuji Nakagawa, Taku Kudoh, Yuji Matsumoto. Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines. In *Proceedings of 6th Natural Language Processing Pacific Rim Symposium (NLPRS2001)*, pages 325–331, 2001.

[39] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of 10th European Conference on Machine Learning*, pages 137–142, 1998.

[40] Thorsten Joachims. Transductive Inference for Text Classification using Support Vector Machines. In *Proceedings of the 16th International Conference on Machine Learning (ICML '99)*, pages 200–209, 1998.

[41] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. New York, 1995.

[42] Vladimir N. Vapnik. *Statistical Learning Theory*. A Wiley-Interscience Publication, 1998.

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

[44] David Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In *Meeting of the Association for Computational Linguistics*, pages 88–95, 1994.

[45] Yiming Yang, Xin Liu. A Re-examination of Text Categorization Methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, pages 42–49, 1999.

[46] Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Hiroshi Matsuda, Kazuma Takaoka and Masayuki Asahara. *Japanese Morphological Analysis System ChaSen version 2.0 Manual 2nd edition. NAIST-IS-TR99009, Nara Institute of Science and Technology*, 12 1999.

[47] 野畑 周. 決定木を用いた学習に基づく固有表現抽出. In *Proceedings of the IREX Workshop*, pages 201–206, 1999.

Appendix

A Abbreviations for phrasal categories in the Penn Treebank.

S	Simple clause (sentence)
SBAR	S' clause with complementizer
SBARQ	*Wh*-question S' clause
SQ	Inverted YES/NO question S' clause
SINV	Declarative inverted S' clause
ADJP	Adjective Phrase
ADVP	Adverbial Phrase
NP	Noun Phrase
PP	Prepositional Phrase
QP	Quantifier Phrase (inside NP)
VP	Verb Phrase
WHNP	*Wh*-Noun phrase
WHPP	*WH*-Prepositional Phrase
CONJP	Multiword conjunction phrases
FRAG	Fragment
INTJ	Interjection
LST	List marker
NAC	Not A Constituent grouping
NX	Nominal constituent inside NP
PRN	Parenthetical
PRT	Particle
RRC	Reduced Relative Clause
UCP	Unlike Coordinated Phrase
X	Unknown or uncertain
WHADJP	*Wh*-Adjective Phrase
WHADVP	*Wh*-Adverb Phrase

B Head Rules

label	search position	priority
NP	right	(POS, NN, NNP, NNPS, NNS), NX, JJR, CD, JJ, JJS, RB, QP, NP
ADJP	right	NNS, QP, NN, $, ADVP, JJ, VBN, VBG, ADJP, JJR, NP, JJS, DT, FW, RBR, RBS, SBAR, RB
ADVP	left	RB, RBR, RBS, FW, ADVP, TO, CD, JJR, JJ, IN, NP, JJS, NN
CONJP	left	CC, RB, IN
FRAG	left	.
INTJ	right	.
LST	left	LS :
NAC	right	(NN,NNS,NNP,NNPS), NP, NAC, EX, $, CD, QP, PRP, VBG, JJ, JJS, JJR, ADJP, FW
PP	left	IN, TO, VBG, VBN, RP, FW
PRN	right	.
PRT	left	RP
QP	right	$, IN, NNS, NN, JJ, RB, DT, CD, NCD, QP, JJR, JJS
RRC	left	VP, NP, ADVP, ADJP, PP
S	right	TO, IN, VP, S, SBAR, ADJP, UCP, NP
SBAR	right	WHNP, WHPP, WHADVP, WHADJP, IN, DT, S, SQ, SINV, SBAR, FRAG
SBARQ	right	SQ, S, SINV, SBARQ, FRAG
SINV	right	VBZ, VBD, VBP, VB, MD, VP, S, SINV, ADJP, NP
SQ	right	VBZ, VBD, VBP, VB, MD, VP, SQ
UCP	left	.
VP	left	VBD, VBN, MD, VBZ, VB, VBG, VBP, VP, ADJP, NN, NNS, NP
WHADJP	right	CC, WRB, JJ, ADJP
WHADVP	left	CC, WRB
WHNP	right	WDT, WP, WP$, WHADJP, WHPP, WHNP
WHPP	left	IN, TO, FW

List of Publication

Journal Papers

- Hiroyasu Yamada, Taku Kudo, Yuji Matsumoto, “Japanese Named Entity Extraction using Support Vector Machine”, Transactions of IPSJ, Vol. 43, No. 1, pages 44-53, 2002.

International Conference

- Hiroyasu Yamada, Fumiyo Fukumoto and Atsumi Imamiya, “Integration of Speech and Deictic Gesture for Referent Identification”, Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS1997), pages 87-92, 1997.
- Hiroyasu Yamada, Fumiyo Fukumoto and Atsumi Imamiya, “Reference analysis of deictic expressions to visual objects”, Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP’97), pages 300-305, 1997
- Hiroya Takamura, Hiroyasu Yamada, Taku Kudoh, Kaoru Yamamoto and Yuji Matsumoto, “Ensemble based on Feature Space Restructuring with Application to WSD”, Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium (NLPRS2001), pages 41-48, 2001.

Other Publications

- 山田 寛康, 工藤 拓, 松本 裕治,“単語の部分文字列を考慮した専門用語抽出”, 情報処理学会研究会報告 NL-140-11, pages 77-84, 2000.
- 山田 寛康, 工藤 拓, 松本 裕治,“Support Vector Machine を用いた日本語固有表現抽出”, 情報処理学会研究会報告 NL-142-17, pages 121-128, 2001.
- 山田 寛康, 松本 裕治,“Support Vector Machine の多値分類問題への適用法について”, 情報処理学会研究会報告 NL-146-6, pages 33-38, 2001.
- 高村大也， 山田寛康， 工藤拓， 山本薫， 松本裕治, “素性空間再構成による Word-sense Disambiguation”, 情報処理学会研究報告, NL-144-13, pages 83-90, 2001.
- 松本裕治， 工藤拓， 高村大也， 山田寛康， 中川哲治, “自然言語処理におけるシステム混合法について”, 第 4 回情報論的学習理論ワークショップ予稿集, pages 19-24, 2001.
- 松本裕治， 山田寛康， 新保 仁, “学習に基づく専門用語分類”, 人工知能学会, 人工知能基礎論研究会, 知識ベースシステム研究 会合同研究会, SIG-FAI/KBS-J-13, pages 79-84, 2001.