Chinese Deterministic Dependency Analyzer: Examining Effects of Global Features and Root Node Finder

Yuchang CHENG, Masayuki ASAHARA and Yuji MATSUMOTO
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
{yuchan-c, masayu-a, matsu}@is.naist.jp

Abstract

We present a method for improving dependency structure analysis of Chinese. Our bottom-up deterministic analyzer adopt Nivre's algorithm (Nivre and Scholz, 2004). Support Vector Machines (SVMs) are utilized to determine the word dependency relations. We find that there are two problems in our analyzer and propose two methods to solve them. One problem is that some operations cannot be solved only using local feature. We utilize the global features to solve this. The other problem is that this bottom-up analyzer doesn't use top-down information. We supply the top-down information by constructing SVMs based root node finder to solve this problem. Experimental evaluation on the Penn Chinese Treebank Corpus shows that the proposed extensions improve the parsing accuracy significantly.

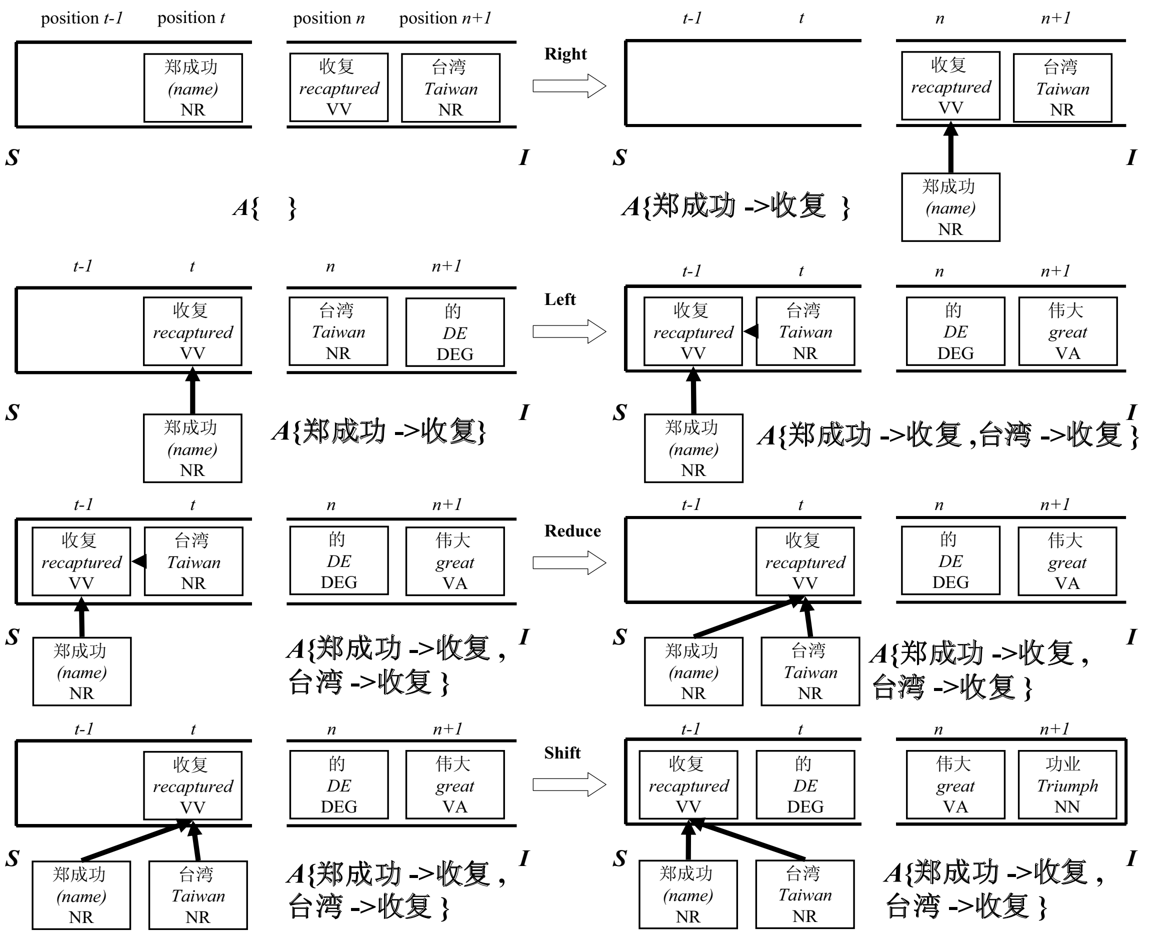
1 Introduction

Many syntactic analyzers for English have been implemented and have demonstrated good performance (Charniak, 2000; Collins, 1997; Ratnaparkhi, 1999). However, implementation of Chinese syntactic structure analyzers is still limited, since the structure of the Chinese language is quite different from other languages. Therefore the experience in processing western languages cannot be guaranteed that it can apply to Chinese language directly (Lee, 1991). Chinese language has many special syntactic phenomena substantially different from western languages. Discussions about such characteristics of Chinese language can be found in the literature (Chao 1968; Li and Thompson 1981; Huang 1982).

About the previous work of Chinese dependency structure analysis, Zhou proposed a rule based approach (Zhou, 2000). Lai et al. proposed a span-based statistical probability approach (Lai, 2001). Ma et al. proposed a statistic dependency parser by using probabilistic model (Ma, 2004). Using machine learning-based approaches for dependency analysis of Chinese is still limited. In this paper, we propose a deterministic Chinese syntactic structure analyzer by using global features and a root node finder.

Our analyzer is a dependency structure analyzer. We utilize a deterministic method for dependency relation construction. First, a dependency relation matrix is constructed, in which each element corresponds to a pair of tokens. A likelihood value is assigned to the dependency relation of each pair of tokens. Second, the optimal dependency structure is estimated using the likelihood of the whole sentence, provided there is no crossing between dependencies. A bottom-up algorithm proposed by (Nivre and Scholz, 2004) is use for a deterministic dependency structure analysis. Our dependency relations are composed by machine learners. SVMs (Vapnik, 1998) deterministically estimate if there is a dependency relation between a pair of words in the methods.

However, this method has two problems. First, some operations in the algorithm needs long distance information. However, the long distance information cannot be available if we assume a context of a fixed size in all operations.



Example: 郑成功收复台湾的伟大功业 (*The great triumph that Cheng Cheng-Kung recaptured Taiwan.*)

Fig. 1. The operations of the Nivre algorithm

The second problem is that the top-down information isn't used in the bottom-up approach. We use the global features to solve the first problem and we construct a SVM-based root node finder in our system to supplement the top-down information.

Our analyzer is trained on the Penn Chinese Treebank 5.0 (Xue et al., 2002), which is a phrase structure annotated corpus. The phrase structure is converted into a dependency structure according to the head rules. We perform experimental evaluation in several settings on this corpus.

In the next section, we describe our deterministic dependency structure analysis algorithm. Section 3 shows the global features and the two-step process. Section 4 describes the use of the root node finder. Section 5 describes the experimental setting and the results. Finally, we summarize our findings in the conclusion.

2 Parsing method

This chapter presents a basic parsing algorithm proposed by (Nivre and Scholz, 2004). The algorithm is the base of our dependency analyzer. This algorithm is based on a deterministic approach, in which the dependency relations are constructed by a bottom-up deterministic schema. While Nivre's method uses memory-based learning, we use SVMs instead. The algorithm consists of two major procedures:

(i) Extract the surrounding features for the focused node (or node pair).

(ii) Estimate the dependency relation operation for the focused node by a machine learning method.

2.1 Algorithm

We utilize a bottom-up deterministic algorithm proposed by (Nivre and Scholz, 2004) in our analyzer. In the algorithm, the states of analyzer are represented by a triple $\langle S, I, A\rangle$. *S* and *I* are stacks, *S* keeps the words being in consideration, and *I* keeps the words to be processed. *A* is a list of dependency relations decide during the algorithm. Given an input word sequence *W*, the analyzer is initialized by the triple $\langle nil, W, \phi\rangle$. The analyzer estimates the dependency relation between two words (the top elements of stack *S* and stack *I*). The algorithm iterates until the list *I* becomes empty. Then, the analyzer outputs the word dependency relations *A*.

There are four possible operations for the configuration at hand:

Right: Suppose the current triple is $\langle t \mid S, n \mid I, A\rangle$ (*t* and *n* are the top elements, *S* and *I* are the remaining elements in the stacks), if there is a dependency relation that the word *t* depends on word *n*, add the new dependency relation $(t \rightarrow n)$ into *A*, remove *t* from *S*. The configuration now becomes $\langle S, n \mid I, A \cup \{(t \rightarrow n)\}\rangle$.

Left: In the current triple is $\langle t \mid S, n \mid I, A\rangle$, if there is a dependency relation that the word *n* depends on the word *t*, adds the new dependency relation $(n \rightarrow t)$ into *A*, push *n* onto the stack *S*. The configuration now becomes $\langle n \mid t \mid S, I, A \cup \{(n \rightarrow t)\}\rangle$.

Suppose the current triple is $\langle t \mid S, n \mid I, A\rangle$, if there is no dependency relation between *n* and *t*, check the following conditions.

Reduce: If there are no more words n' ($n' \in I$) which may depend on *t*, and *t* has a parent on its left side, the analyzer removes *t* from the stack *S*. The configuration now becomes $\langle S, n \mid I, A\rangle$.

Shift: If there is no dependency between *n* and *t*, and the triple doesn't satisfy the conditions for **Reduce**, then push *n* onto the stack *S*. The configuration now becomes $\langle n \mid t \mid S, I, A\rangle$.

These operations are depicted in **Fig. 1**. Given an input sentence of length *N* (words), the analyzer is guaranteed to terminate after at most *2N* actions. The dependency structure given at the termination is well-formed if and only if the relations in *A* constitute a single connected tree. This means that the algorithm produces a well-formed dependency graph.

2.2 Machine learning method

A classification task usually involves with training and testing data which consist of annotated data instances. Each instance in the training set contains one "target value" (class label) and several "attributes" (features). The goal of a classifier is to produce a model which predicts target value of data instances in the testing set which only give the attributes.

SVMs are binary classifiers based on the maximal margin strategy. Suppose we have a set of training data for a binary classification problem: $(\boldsymbol{x}_1, y_1) \ldots (\boldsymbol{x}_n, y_n)$, where $\boldsymbol{x}_i \in R^n$ is the feature vector of the *i*-th sample in the training data and $y_i \in \{+1, -1\}$ is the class label of the sample. The goal is to find a decision function $f(x) = sign(\sum_{\mathbf{z}_i \in SV} a_i y_i K(\boldsymbol{x}, \boldsymbol{z}_i) + b)$ for an input vector $\boldsymbol{x}$. The vectors $\boldsymbol{z}_i \in SV$ are called support vectors, which are representative examples. Support vectors and other constants are determined by solving a quadratic programming problem. $K(\mathbf{x}, \mathbf{z})$ is a kernel function which maps vectors into a higher dimensional space. We use the polynomial kernel: $K(\mathbf{x}, \mathbf{z}) = (1 + \mathbf{x} \cdot \mathbf{z})^d$. The performance of SVMs is better than using other machine learning methods, such as memory based learning or maximum entropy method, in our analyzer. This is because that SVMs can adopt combining features automatically (using the polynomial kernel), whereas other method cannot. To extend binary classifiers to multiclass classifiers, we use the pair-wise method, which utilizes ${}_nC_2$ binary classifiers between all pairs of the classes (Kreel, 1998). We use Libsvm (Lin et al., 2001) in our experiments.

2.3 Features (Local features)

It should be noted that we use a different machine learner from the original method (Nivre, 2004). Nivre's work used memory based learning in their analyzer, we utilize SVMs in our analyzer. Therefore, the features of our analyzer are different from the original Nivre's method.

In our method, the analyzer considers the dependency of two nodes (*n,t*) which are in current triple. The nodes include the word, the POS-tag and the information of its children. The context features we use are 2 preceding nodes of node *t* (and *t* itself), 2 succeeding nodes of node *n* (and *n* itself), and their child nodes. The distance between nodes *n* and *t* is also used as a feature. We call these features as local features.

3 Global features and two-step process

In the algorithm, the operation **Reduce** needs the condition that the node *n* should have no child in *I*. However, it is difficult to check this condition. In a long sentence, the modifier of the focused node *n* may be far away from *n*. Moreover, some non-local dependency may cause this kind of error. In this section, we will describe this problem and a solution to it.

3.1 Global features

The analyzer selects features for deciding the optimum operation, and then gives these features to machine learner. The machine learner uses the same information to decide the optimum operation even when these operations essentially disagree. However, the different operation consists of different condition. In the deterministic bottom-up dependency analysis, we can generally consider the process as two tasks:

Task 1: Does the focused word depend on a neighbor node?

Task 2: Does the focused word may have a child in the remaining token sequence?

In the Task 1, the problem can be resolved by using the information of the neighbor nodes. This information is possibly the same as the features that we described in section 2.3. However, these features may not be able to resolve the problem in task 2. For resolving the problem in task 2, we need the information of long distance dependency. In **Fig. 2,** for example, the analyzer is considering the relation between focused words "告诉 (*tell*)" and "他 (*he*)". The features used in this original analysis are the information of words "请 (*please*)", "告诉 (*tell*)", "他(*he*)", "何时 (*what time*)" and "准备 (*prepare*)". These features are "local features". The correct answer in this situation is the operation "**Shift**". It is because the word "告诉 (*tell*)" has a child "出发 (*start*)" which is not yet analyzed and the focused words don't depend on each other. However, the local features do not include the information of word "出发 (*start*)". Therefore, the analyzer possibly estimates the answer as the operation "**Reduce**". The results make a mistake in this situation because of the lack of long distance information. To resolve this problem, we should refer some information of long distance dependency in machine learning. The information about long distance relations is defined as "global features". In this paper, we select the words which remain in stack *I* but don't be consider in local features as global features.

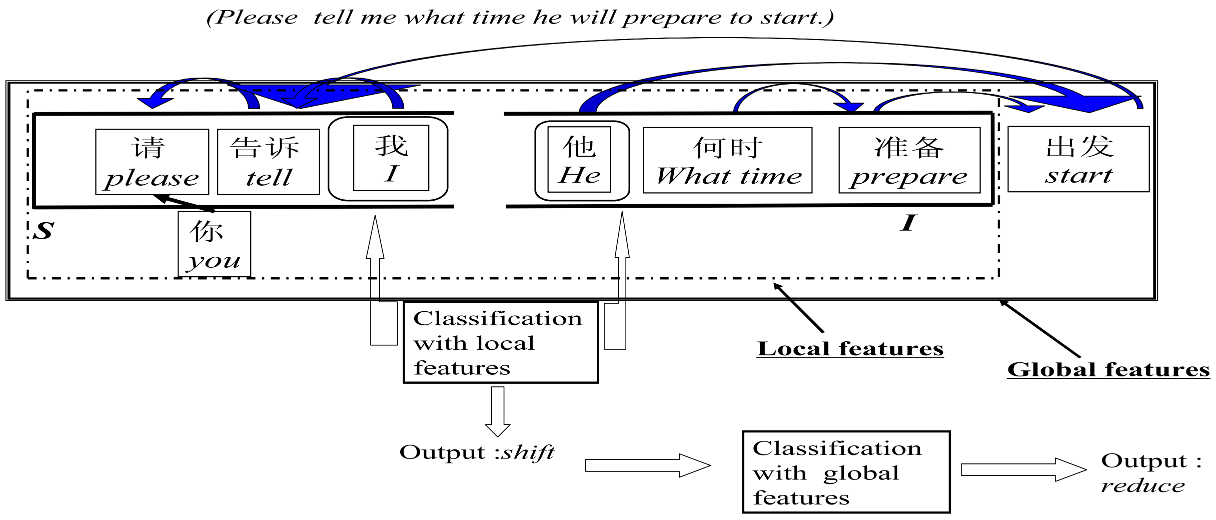


Fig. 2. An example of the ambiguity of deciding the long distance dependency relation and using two-steps classification dependency relation

3.2 two-step process

To use the global features, we cannot use them immediately because the global features are not effective in all operations. For using global features efficiently, we propose a two-step process in our analyzer. The analysis processes are divided to two processes. First, the analyzer uses only the local features (as described in Section 2.3) to decide the optimum operation. If the result is "**Reduce**" or "**Shift**", it means that the focused words do not have any dependency relation. The analyzer leaves the decision to another machine learner that makes use of global features. The analyzer will select global features for analyzing the Task 2. Then the analyzer outputs the final answer of this analysis process.

Fig. 2 describes an example of using two-step classification for analyzing dependency relation. In this example, the focused words are "我 (*I*)" and "他 (*He*)". The word "我 (*I*)" depends on the word "告诉 (*tell*)". The local features are surrounded by dotted line and the global features are surrounded by solid line. The analyzer used local features to analyze the operation of this situation. The result is the operation "***shift***". The analyzer then selected the global features to analyze again and the output is the operation "***reduce***". The final result of this situation is the operation "***reduce***".

4 The root node finder

In Isozaki's work (Isozaki et. al, 2004), they adopted a root finder in their system to find the root word of the input sentence. Their method used the information of the root word as a new feature for machine learning. Their experiments showed that information of root word was a beneficial feature. However, we think the information of root word can be used not only as the feature of machine learning, but also can be used to divide the sentence. Therefore, the complexity of the sentence can be alleviated by dividing the input sentence.

4.1 Root node and dividing sentence by using root finder

In the fundamental definition of dependency structure, there is one and only one head word in a dependency structure. An element cannot have dependents lying on the other side of its own governor.

These peculiarities imply that the head word divides the phrase into two independent parts and each part does not cross the head word. As in **Fig. 3**, the original input sentence has a root word (the head word of phrase) "与 (*and*)". There are not any dependency relation which crosses the root word. Therefore we can divide this sentence into two sub-sentence "出国 (*exodus*) / 去 (*do*) / 进修 (*study*) / 与 (*and*)" and "与 (*and*) / 到 (*go*) / 国外 (*foreign country*) / 去(*do*) / 旅行 (*visit*)". Both these sub-sentences have their root word and the root word is "与(and)". We can conceive that to analyze the dependency structure of the full sentence is to analyze the dependency structure of two sub-sentences. Combining structures of two sub-sentences, we can get the full structure of original sentence. Our dependency analyzer is a bottom-up deterministic analyzer. Instinctively, the accuracy of analyzing short sentence is significantly better than analyzing long sentence. Thus the performance of the dependency analyzer can be improved by this method.

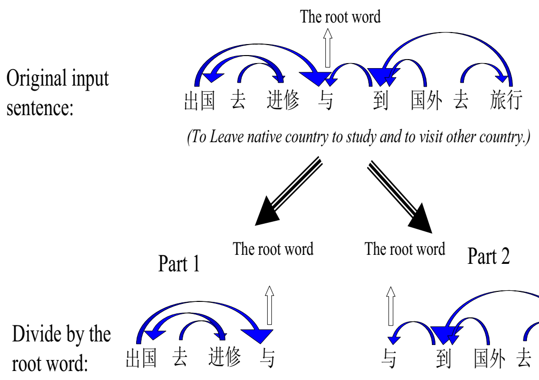


Fig. 3. Dividing the phrase as two phrases by the root word

4.2 Constructing a root finder

To use the root node, we should construct the root finder. Similarly to Isozaki's work, we use machine learner (SVMs) to construct the root finder. We refer to the features which are used in Isozaki's work and investigate other effective features. The performance of our root node finder is 90.71%. This is better than the root accuracy of our analyzer (86.22%, see **Table 2**).

Therefore, using the root finder can give the dependency analyzer more top-down information.

The tags and features of the root finding are shown in **Fig. 4**. We extract all root words in the training data and tagging every word to show that it is root word or not. For example, the root word in **Fig. 4** is "得到 (*get*)". The root finder analyzes each word in the sentence and gives the tag "true" or "false" to indicate the root word. The features for machine learning of root finder include the contextual features (the information about the focused word, the two preceding words, and two succeeding words) and the word relation features (the words which are in the outside of the window). Other effectual features include the Boolean features "root word is found" and "the focus word is the first/last word of sentence". For example, the contextual features of the word "经济 (*economic*)" include information of the focused (*n*) word "经济 (*economic*)", the "*n*-1"th word "宏观 (*wide*)", the "*n*-2"th word "的 (*DE*)", the "*n*+1"th word" 环境 (*environment*)" and the "*n*+2"th word "将 (*will*)". The word relation features include the preceding word set {中国 (*China*)}, the succeeding word set {得到, 进一步, 的, 改善} and the Boolean features are: "root_word_is_found=false", "first_word=false" ,"last_word=false".

When we use the root finder to analyze the root word of the sentence, we do not know the structure of input sentence (either the phrase structure or the dependency structure). It may look odd that the root finder can analyzes the root word without any information of the structure. However, this analysis is practicable. Naturally, the root word of a sentence is usually a verb (about 61% of sentences have a verb as the root word in our testing corpus). For example, in the example 1 of **Fig. 5** "我 / 去 / 学校 (*I go to school*)", we know the POS-tags are "noun, verb, noun" thus we can find that the root word is "去 (go)". However, many sentences include more then one verb or the root word is not verb (in NP or PP…etc.). We can not only choose the verbs as root word directly. To decide the root word of complex sentences, there are some special word/POS relations that can be used to estimate the root node of a sentence. Considering the root finder in **Fig. 4**, the root finder gives the root tag to each word of the sentence.

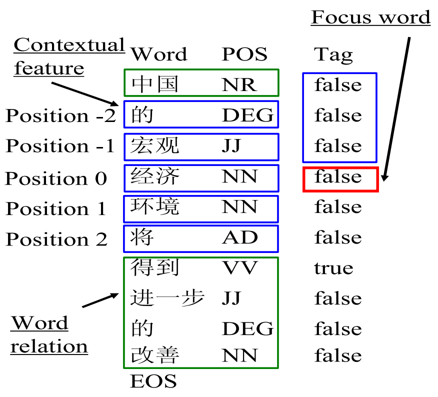
The processes of analyzing the root word can be thought as two tasks:

Task 1: Does the focus word depend on a neighbor word?

Task 2: Are there any special relation in the sentence?

In **Fig. 4**, the contextual features (two preceding words and two succeeding words) can be used to process the **Task 1**, and the word relation features can be used to process the **Task 2**. If the focused word possibly depends on neighbor words, it is impossible that the focused word is the root word. Therefore these words will be tagged as "false".

Alternately, considering the example 2 in **Fig. 5**, the sentence has a verb "收复 (*recapture*)", but the special word "的 (*DE*)" is in the right side of the verb "收复 (*recapture*)". Therefore, the verb "收复 (*recapture*)" is possibly in the 的 (*DE*)-phrase and the verb cannot be the root word. The special word "的 (*DE*)" resembles a preposition and it is always the last word of DE-phrase. Therefore, although we do not know the structure of sentence, we can identify which words can be the root word by the relation and position of the features. If the features of the focused word include the special word relations



	Word	POS	Tag
	中国	NR	false
Position -2	的	DEG	false
Position -1	宏观	JJ	false
Position 0	经济	NN	false
Position 1	环境	NN	false
Position 2	将	AD	false
	得到	VV	true
	进一步	JJ	false
	的	DEG	false
	改善	NN	false
	EOS		

Fig. 4. The features and tag of root finder

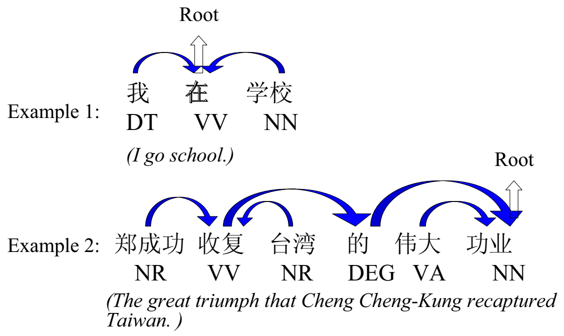

Root
Example 1:
我 在 学校
DT VV NN
(I go school.)
Root
Example 2:
郑成功 收复 台湾 的 伟大 功业
NR VV NR DEG VA NN
(The great triumph that Cheng Cheng-Kung recaptured Taiwan.)

Fig. 5. The examples of analyzing the root word of sentences

(for example, the focused word is in the prepositional phrase), it isn't the root word. The features "word relations" in **Fig. 5** can consider this situation.

5 Experiments

5.1 Corpus and estimation

We use Penn Chinese Treebank 5.0 (Xue et al., 2002) in our experiments. This Treebank is represented by phrase structure and doesn't include the head information of each phrase. The first step of using Penn Chinese Treebank is to derive the head rules for deciding the head word of each phrase. Some examples of head rules are shown in **Table 1.** We convert the Treebank by using these head rules. The training corpus includes about 377,408 words for learning and 63,886 words for testing. It should be noted that the punctuation mark "。" marks the end of a sentence in the Treebank. However, the punctuation mark "," also can be the end of a sentence. It is hard to determine the dependency rule of the clauses on the both side of comma. Therefore, to decide the dependency relation which crosses a punctuation mark "," is difficult. We do not deal with the ambiguity of commas and divide the sentence by the punctuation mark ",".

Phrase	The order of deciding the head of phrase (from left)
ADJP	CC PZ ADJP JJ
ADVP	CC PZ AD
CLP	PZ CLP M LC
DP	DP CLP QP DT
DVP	DEV DEC DEG
VCP	VC VV

Table 1. Some examples of head rules

The performance of our dependency structure analyzer is evaluated by the following three measures:

Dependency Accuracy:

$$= \frac{number\ of\ correctly\ analyzed\ dependency\ relations}{number\ of\ dependency\ relations}$$

Root Accuracy:

$$= \frac{number\ of\ correctly\ analyzed\ root\ nodes}{number\ of\ clauses}$$

Sentence Accuracy:

$$= \frac{number\ of\ fully\ correctly\ analyzed\ clause}{number\ of\ clauses}$$

5.2 Results and discussion

Our experimental results are shown in **Table. 2**. First row in the table is the result of our basic analyzer (Nivre algorithm with SVMs), second and third row show the effects of the proposed extensions. The last row is the result of combining the two extensions. We had used McNemar test to confirm the significance of the methods. The McNemar test proves that using the proposed methods improve the analyzers significantly. Comparing the results of our basic analyzer to related works, our analyzer (dep. Accuracy: 87.64) is better than (Ma et al., 2004, dep. Accuracy: 80.38) and (Zhou, 2000, dep. Accuracy of newspaper: 67.7). However, these researches used different corpus. We cannot compare the performances directly.

According to the second row of **Table. 2**, dividing the process of classification as two steps can improve the performance of dependency analyzer. However, the improvement of using this method is limited. This is because that long distance relations are not many in the corpus. The absence of global information does not occur in the sentences without long distance relations. Another reason is the distribution of operations. The instances of operations in our experimental corpus are not balanced. The operation "***reduce***" is the least (7.8%) and it is far less than other operations. Therefore the instances for creating the model of operation "***reduce***" are not satisfactory. These facts result in that our experiment of using two step classification cannot improve the analyzer remarkably.

About the experiment of utilizing root finder in our analyzer, we tried to adopt the root information to the analyzer (using the information as features for machine learning). However, the performance is worse than the baseline (the fundamental analyzer "Nivre+SVMs"). Therefore, we use our method to improve the analyzer by using root information (dividing the sentence according to root node).

According to the third row of **Table. 2**, dividing the sentence into two sub-sentences can improve the performance of dependency analyzer. However, the sentence accuracy cannot increase reliably. This result shows that using root finder and dividing sentence can reconstruct some mistakes in sentences. Certainly, the performance of the root finder influences the analyzer strongly. If we use a perfect root node finder into our analyzer, the performance will improve significantly.

The last row of **Table. 2** shows the results of combining the two proposed methods (using global features and root node finder) to improve our analyzer. Combining two methods can increase the dependency accuracy better than using either one of the methods. It means that some analysis errors of fundamental analyzer can be resolved by using both improvement methods. Therefore using combined method cannot supply higher improvement.

	Dep. Acc.	Root Acc.	Sent. Acc.
Baseline (Nivre with SVMs)	85.25	86.18	59.98
Baseline with two-step process	85.44	86.22	60.1
Baseline with root node finder	86.13	**90.94**	**61.33**
Baseline with two-step process and root node finder	**86.18**	**90.94**	**61.33**

Table 2. The experimental results

6 Conclusion and future work

In this paper, we present two methods to improve a deterministic dependency structure analyzer for Chinese. This basic analyzer implements a bottom-up deterministic algorithm with SVMs. We convert a phrase structure annotated corpus (Penn Chinese Treebank) to dependency tagged corpus by using head rules. According to the properties of Chinese language and dependency structure, we try to add a root finder in our dependency analyzer to improve the analyzer. Moreover, considering the machine learning process of our analyzer, we divide the process into two processes to improve the performance of analyzer. The improving methods (using root finder and dividing machine learning process) showed to improve the analyzer.

Future work includes three points. First, we should improve the performance of the root finder. Second, we should construct a useful prepositional phrase chunker, because the prepositional phrase is a major error source of our basic analyzer. The original analyzer tends to let the preposition governing a partial subtree of the full phrase. According to the properties of Chinese language, the prepositional phrases in Chinese are head-initial. Intuitively, if we can extract the prepositional phrases from sentence, the complexity of the sentence will decrease. Thus an important task is how to chunk the prepositional phrase in the sentence.

Finally, we should deal with the ambiguity of the meaning of punctuation mark "，". The definition of "sentence" is ambiguous in Chinese. In Chinese articles, the normal ending mark of a sentence is the punctuation mark "。". However, the mark "，" is often used at the end of a sentence. To distinguish the meaning of the punctuation mark "，" is difficult. Therefore, we should adopt semantic analysis in our analyzer.

References

1. Eugene Charniak, 2001. Immediate-Head Parsing for Language Models. pages 124-131, NAACL-2001.
2. Yuen Ren Chao, 1968. A Grammar of Spoken Chinese. Berkeley, CA: University of California Press.
3. Michael Collins, Brian Roark, 2004, Incremental parsing with the Perceptron algorithm. Pages 112-119, ACL-2004.
4. J. Huang, 1982. Logical relations in Chinese and the theory of grammar Doctoral dissertation, Massachusetts Institute of Technology, Cambridge.
5. Ulrich. H.-G. Kreβel, 1998. Pairwise classification and support vector machines. In Advances in Kernel Methods, pages 255–268. The MIT Press.
6. Chih Jen Lin, 2001. A practical guide to support vector classification, http://www.csie.ntu.edu.tw/~cjlin/libsvm/.
7. Lai, Bong Yeung Tom, Huang, Changning, 1994. Dependency Grammar and the Parsing of Chinese Sentences. PACLIC 1994
8. Hideki Isozaki, Hideto Kazawa, Tsutomu Hirao, 2004. A Deterministic Word Dependency Analyzer Enhanced With Preference Learning, pages 275-281, COLING-2004
9. Charles Li, and Thompson Sandra A., 1981. Mandarin Chinese. University of California Press.
10. Lin-Shan Lee, Long-Ji Lin, Keh-Jiann Chen, and James Huang, 1991. An Efficient Natural Language Processing System Specially Designed for the Chinese Language. Computational Linguistics, Volume 17, Number 4.
11. Ma Jinshan, Zhang yu, Liu ting, and Li sheng, 2004. A Statistical Dependency Parser of Chinese-under Small Training Data. IJCNLP 2004 Workshop: Beyond shallow analyses, Formalisms and statistical modeling for deep analyses.
12. Joakim Nivre and Mario Scholz, 2004. Deterministic Dependency Parsing of English Text. Pages 64-70, COLING-2004.
13. Adwait Ratnaparkhi, 1999. Learning to parse natural language with maximum entropy models. Machine Learning, 34(1-3) pages151–175.
14. Vladimir N. Vapnik, 1998. Statistical Learning Theory. A Wiley-Interscience Publication.
15. Nianwen Xue, Fu-Dong Chiou, Martha Stone Palmer, 2002. Building a Large-Scale Annotated Chinese Corpus. COLING 2002
16. Ming Zhou, 2000. A block-based robust dependency parser for unrestricted Chinese text. The second Chinese Language Processing Workshop attached to ACL-2000.