Journal of Zhejiang University SCIENCE A
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Using SVM to construct a Chinese dependency parser

XU Yun (许 云)†, ZHANG Feng (张 锋)

(*Computer Department, Beijing Institution of Technology, Beijing 100081, China*)

†E-mail: yxu1976@126.com; yxu1976@gmail.com

Received Nov. 15, 2004; revision accepted Mar. 20, 2005

Abstract: In Chinese, dependency analysis has been shown to be a powerful syntactic parser because the order of phrases in a sentence is relatively free compared with English. Conventional dependency parsers require a number of sophisticated rules that have to be handcrafted by linguists, and are too cumbersome to maintain. To solve the problem, a parser using SVM (Support Vector Machine) is introduced. First, a new strategy of dependency analysis is proposed. Then some chosen feature types are used for learning and for creating the modification matrix using SVM. Finally, the dependency of phrases in the sentence is generated. Experiments conducted to analyze how each type of feature affects parsing accuracy, showed that the model can increase accuracy of the dependency parser by 9.2%.

Key words: Syntactic parsing, Dependency, Support Vector Machine (SVM)
doi:10.1631/jzus.2006.A0199 **Document code:** A **CLC number:** TP391

INTRODUCTION

Conventional parsers with practical levels of performance require a number of sophisticated rules that have to be handcrafted by linguists. It is time-consuming and cumbersome to maintain. A lot of rules will generate conflict, so such rules often deteriorate the overall performance of the parser. The stochastic approach, on the other hand, has the potential to overcome these difficulties. In the late 1980s and early 1990s, the induction and parameter estimation of probabilistic context free grammars (PCFGs) from corpora were intensively studied. Because these grammars comprise only non-terminal and part-of-speech tag symbols, their performances were not enough to be used in practical applications (Charniak, 1993). A broader range of information, in particular lexical information, was found to be essential in disambiguating the syntactic structures of real-world sentences. SPATTER (Magerman, 1995) augmented the pure PCFG by introducing a number of lexical attributes. The parser controlled applications of each rule by using the lexical constraints induced by decision tree algorithm. The SPATTER parser attained 87% accuracy and first made stochastic parsers a practical choice. The other type of high precision parser, based on dependency analysis was introduced by Collins (1996). Dependency analysis first segments a sentence into syntactically meaningful sequences of words and then considers the modification of each segment. Collins' parser computes the likelihood that each segment modifies the other (2 term relation) by using large corpora. These modification probabilities are conditioned by head words of two segments, distance between the two segments and other syntactic features. Although these two parsers have shown similar performance, the keys to their success are slightly different. SPATTER parser performance greatly depends on the feature selection ability of the decision tree algorithm rather than its linguistic representation. On the other hand, dependency analysis plays an essential role in Collins' parser for efficiently extracting information from corpora. The best Chinese dependency parsers that based on rules and statistics attained 62.1% accuracy (Guo and Zhou, 2000).

The many exceptions and idioms in natural languages will cause data to be too sparse for maxi-

mum likelihood estimate or make it difficult for decision tree algorithm to prune. To solve the problem, the SVM based Chinese dependency parser introduced in this paper uses more feature types to increase its accuracy. We evaluated the proposed parser using the Penn Chinese Treebank (CTB). Experiments showed that the dependency parser is more accurate than other Chinese dependency parsers.

DEPENDENCY ANALYSIS FOR CHINESE

In Chinese, dependency analysis has been shown to be powerful because the order of phrases in a sentence is relatively unimportant compared to English. Chinese dependency parsers generally proceed in three steps.

Step 1: Segment a sentence into a sequence of phrases;

Step 2: Judge whether there is any modification relationship of each phrase pair in the sentence;

Step 3: According to Step 2, prepare a modification matrix, each value of which represents how one phrase is likely to modify another.

Because there are no explicit delimiters between words in Chinese, input sentences are first word segmented, part-of-speech tagged, and then chunked into a sequence of phrases (Zhang and Zhou, 2002). The first step yields, for the following example, the sequence of phrases displayed below (Fig.1). The parentheses in the Chinese expressions represent the internal structures of the phrases.

例句： 昨天晚上邻居的小孩喝了很多酒
Sample: The neighbor children drank a lot of wine yesterday night.

(昨天)	(晚上)	((邻居)(的))			
Yesterday	night	neighbor			
1	2	3			
(小孩)	((喝)(了))	(很多)	(酒)		
children	drank	a lot of	wine.		
4	5	6	7		

Fig.1 Word segmented for the sample sentence

The second step of parsing is to judge the modification relationship between phrases, which is a binary classification problem. The value 1 means there is modification between two phrases, otherwise, the value –1 means there is no any modification between two phrases. In order to decrease the computational complexity and make the modification matrix change to triangular matrix, we make two assumptions in Chinese: (1) Every phrase except the last one modifies only one posterior phrase; (2) No modification crosses to other modifications in a sentence.

The two assumptions can decrease the computation amount by about 50%, because the model only need calculate half of the modification matrix. According to the statistical data, the sentences that do not meet the two assumptions have 2 percent of all the Chinese sentences. Most of the sentences are inversion sentences or specific idioms.

Table 1 shows a modification matrix for the example sentence. In the matrix, columns and rows represent anterior and posterior phrase, respectively. Because of the first assumption, the columns of the modification matrix do not have the first phrase, and the rows of the modification matrix do not have the last phrase. For example, the first phrase modifies the second phrase but not the third phrase in the sample sentence.

Table 1 Modification matrix of the sample sentence

	1	2	3	4	5	6
2	1					
3	–1	–1				
4	–1	–1	1			
5	–1	1	–1	1		
6	–1	–1	–1	–1	–1	
7	–1	–1	–1	–1	1	1

Before going into the SVM model, we introduce the notations that will be used in it. Let S be the input sentence comprised of a length m ($\{<b_1, f_1>,\ldots,<b_m, f_m>\}$) phrase set B in which b_i and f_i represent the ith phrase and its features, respectively. We define D to be a modification set $D=\{mod(1),\ldots,mod(m-1)\}$ in which $mod(i)$ indicates the number of phrase modified by the ith phrase. Because of the first assumption, the length of D is always m–1. Using these notations, the result of the third step for the example can be given as $D=\{2,5,4,5,7,7\}$ as displayed in Fig.2.

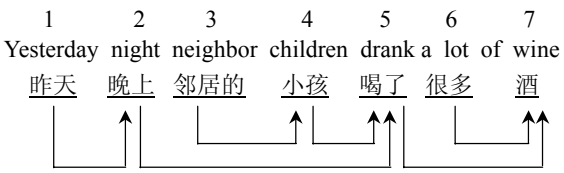

Fig.2 Modification set for sample sentence

SVM MODEL

Key ideas of SVM

This section briefly introduces the machine learning methodology of Support Vector Machine (Cortes and Vapnik, 1995). SVM offer the following advantages over conventional statistical learning algorithms (i.e., decision tree learning, maximum entropy method): (1) High generalization performance even with high dimension feature vectors; (2) The ability to manage kernel functions that map input data to higher dimensional space without increasing computational complexity.

The main idea of SVM is to create an optimal hyperplane to classify the data into two classes (positive and negative) and maximize distance between the hyperplane separating the two classes and the closest data points to the hyperplane (Fig.3).

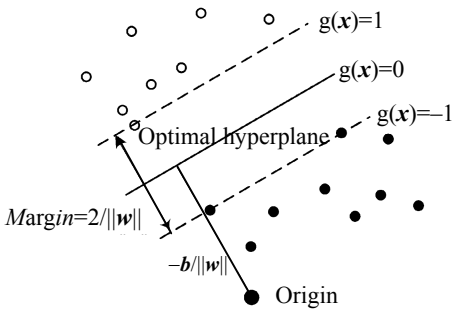

Fig.3 Support Vector Machine

The explanation of SVM starts with a set of l training data $(\boldsymbol{x}_1, y_1),...,(\boldsymbol{x}_l, y_l)$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ is an n-dimensional vector and $y_i \in \{-1, +1\}$ is the class label of ith data.

The optimal hyperplane $\boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b}=\boldsymbol{0}$ separates the training data into two classes. The basic idea of SVM is to maximize the margin between the positive and negative examples. Fig.3 shows training examples linearly separated into two classes.

In general, it is not necessary to separate training examples into each class. Variable $\xi_i \geq 0$ is introduced for misclassification errors and $C>0$ is a constant. This optimization problem is defined as follows:

$$\min \phi(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_1^l \xi_i , \quad (1)$$

$$\text{s.t.}: y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + \boldsymbol{b}) \geq 1,\ i = 1,\cdots,l . \quad (2)$$

The first term in Eq.(1) specifies the size of the margin and the second term represents the cost of the misclassification.

The decision function $f(\boldsymbol{x})$ can be written as:

$$f(\boldsymbol{x})=\text{sgn}(g(\boldsymbol{x})), \quad (3)$$

$$g(\boldsymbol{x}) = \sum_i \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{x}) + \boldsymbol{b} , \quad (4)$$

where $\alpha_i \geq 0$ are Lagrange multipliers. When the maximal margin hyperplane is found in feature space, only those points which lie closest to the hyperplane have $\alpha_i>0$ and these points are the support vectors. All other points have $\alpha_i=0$. This means that the representation of the hypothesis is solely given by those points that are closest to the hyperplane and they are the most informative patterns in the data.

We calculate the Kernel function defined as $\Phi(u)\cdot\Phi(v)=K(u,v)$ for a non-liner SVM classifier. Using a Kernel function, we can rewrite Eq.(4) as:

$$g(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + \boldsymbol{b}. \quad (5)$$

In this paper, we use polynomial Kernel functions that have been very effective when applied to other tasks, such as natural language processing (Kudo and Matsumoto, 2001; Hirao and Isozaki, 2002):

$$K(\boldsymbol{x}\cdot y)=(\boldsymbol{x}\cdot y+1)^d. \quad (6)$$

Application to dependency analysis

From the viewpoint of machine learning, dependency analysis is defined as the task of training and classifying the modification relationship of the phrase pairs into positives (modification) and negatives (no modification) in a sentence.

To apply SVM, we have to prepare a set of training data that contain feature vector $\boldsymbol{x}_i$ of the ith phrases modification relation. The system parameters that were computed in the process are recorded and used to create feature vector for each pair of phrase in the sentence. We used $f(\boldsymbol{x})$ in Eq.(3) to decide whether the two phrases have modification relation. If $f(\boldsymbol{x})=1$, then the two phrases have modification relation, otherwise, they have not any modification relation.

FEATURE TYPES USED FOR LEARNING

This section explains the concrete feature set used for learning. It is important for the SVM model to choose the feature set (Li *et al*., 2003). The feature set mainly focuses on the two phrases constituting each data. The class set consists of binary values which determine whether a sample (the two phrases) have a modification relation or not. We use 13 features for the task, 10 directly from the two phrases under consideration and 3 for other phrases information as summarized in Table 2.

Each phrase (anterior and posterior) has the 5 features: No. 1 to No. 5 in Table 2. Features No. 6 to No. 8 are related to phrase pairs. Feature No. 1 concerns the head word of the phrase, its value is the sequence code of the head word in HowNet which is a Chinese electronic dictionary. There are 120496 words defined in the HowNet. If the head word of the phrase does not exist in the HowNet, its value will be set to 0. No. 2 concerns the POS of the head word, such as verb, noun and so on. No. 3 deals with the phrase types, such as NP, VP. Because the Penn Chinese TreeBank has defined 33 tags for POS and 17 tags for phrase types, the value of No. 2 belongs to [1, 33] and the value of No. 3 belongs to [1, 17]. No. 4 and No. 5 are binary features and correspond to punctuation and parentheses, respectively. No. 6 represents how many phrase exist between the two phrases. Possible values are 0(0), 1(0~4) and 2(>5). No. 7 deals with the particle 'de' which greatly influences the long distance dependency modifications. Finally, No. 8 addresses the punctuation between the two phrases. The detailed values of each feature type are summarized in Table 2.

Table 2 Feature types used for learning

No.	Feature types	Values
1	Lexical information of head word	0,1,2,…,120496
2	Part-of-speech of head word	1,2,…,33
3	Type of phrase	1,2,…,17
4	Punctuation in phrase	0,1
5	Parentheses	0,1
6	Distance between two phrases	0,1,2
7	Particle 'de' between two phrases	0,1
8	Punctuation between two phrases	0,1

EXPERIMENTAL RESULTS

The experiments used the TinySVM software to compute the SVM model. We evaluated the proposed parser using the Penn Chinese TreeBank, which is the news of Xinhua Press from 1994~1998. The corpus includes about 4200 sentences, which is composed of about 10^5 Chinese words and the average length of the sentences is about 25 words. The reason why we choose the Penn Chinese TreeBank as the training and test data is that all sentences in the corpus were word segmented, part of speech tagged and chunked into a sequence of phrases. Sentences inconsistent with the two assumptions were removed from the TreeBank. We randomly chose 100 sentences from Penn Chinese TreeBank as test data, and chose 4000 sentences as training data. Before the experiment, the training data have to code the dependency relation of the sentences by using the value of feature types in Table 2. The format follows as:

$$\{1/-1,\ f_i,\ f_j,\ f_{ij}\},\ i<j,$$

where f_i represents the feature of the ith phrase with its type being No. 1~No. 5. f_{ij} represents the feature between the ith and jth phrases, and its type is No. 6~No. 8. 1 and –1 respectively represent there is modification or no modification between the ith and jth phrases.

Fig.4 shows how the number of training sentences influences parsing accuracy for the same 100 test sentences. They illustrate the following two characteristics of the learning curve:

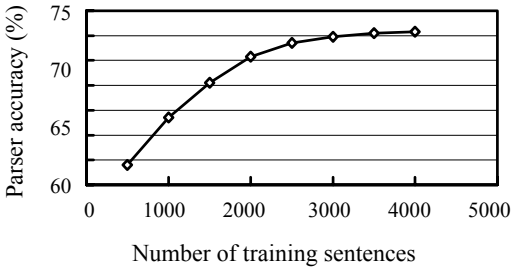

Fig.4 Learning curve of SVM

(1) The parsing accuracy rapidly rises up to 2500 sentences and converges at around 4000 sentences.

(2) The maximum parsing accuracy is 77.3% for 4000 training sentences.

Table 3 illustrates how the parsing accuracy is reduced when each type of feature is removed. The number of training sentences was 4000.

Table 3 clearly shows that the most significant feature types are the type of phrase and the distance between the two phrases. This result may partially support an often-used heuristic; phrase modification should be as short range as possible, provided the modification is syntactically possible. In particular, we need to concentrate on the types of phrase to attain a higher level of accuracy. Most features contribute, to some extent, to the parsing performance. In our experiment, information on parentheses has no effect on the performance. The reason may be that Penn Chinese TreeBank contains only a small number of parentheses.

Table 3 Decrease of parsing accuracy when each feature type is removed

No.	Feature types	Accuracy decrease
1	Lexical information of head word	–6.4%
2	Part-of-speech of head word	–3.6%
3	Type of phrase	–9.3%
4	Punctuation in phrase	–3.7%
5	Parentheses	–0.0%
6	Distance between two phrases	–8.9%
7	Particle 'de' between two phrases	–3.9%
8	Punctuation between two phrases	–1.2%

The accuracy of the Chinese dependency parser using SVM has attained 77.3%. The experimental results showed that it outperforms conventional Chinese dependency parsers based on rules and statistics by 9.2% (Guo and Zhou, 2000). Although the training data and test data in this paper are different from the conventional Chinese parsers, all of their test are in the open domain. Therefore, the result is authentic.

CONCLUSION

In Chinese, dependency analysis has been shown to be powerful because the order of the phrases in a sentence is relatively free compared with English. All the conventional Chinese dependency parsers are based on rules and statistics. In this paper, a dependency parser using SVM is introduced for analyzing each type of feature affects the parsing accuracy. The experimental results showed that the accuracy of the parser has attained 77.3%, and 9.2% higher than that of conventional Chinese dependency parsers.

The dependency parser using SVM has certain advantages:

(1) Freedom from using hand crafted rules causing the difficulty for the parser to maintain. What is more, a lot of rules will bring about conflict.

(2) Easy judging of how each type of feature affects the accuracy of the parser.

(3) Adding any new features will not modify the SVM model, so that the model is reusable in different domains.

References

Cortes, C., Vapnik, V.M., 1995. Support Vector Networks. *Machine Learning*, **20**:273-297.

Charniak, E., 1993. Statistical Language Learning. The MIT Press, Cambridge, p.116-121.

Collins, M., 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. Proceedings 34th Annual Meeting of Association for Computational Linguistics. University of California, Santa Cruz, CA, p.184-191.

Guo, Y.H., Zhou, C.L., 2000. A study of the parsing strategy and the generative algorithm for dependency relation network of Chinese sentences. *Journal of Zhejiang University (Engineering Science)*, **27**:637-645 (in Chinese).

Hirao, T., Isozaki, H., 2002. Extracting Important Sentences with Support Vector Machines. Proceedings of the 19th International Conference on Computational Linguistics. Taipei University, Taipei, p.342-348.

Kudo, T., Matsumoto, Y., 2001. Chunking with Support Vector Machines. Proceedings of NAACL 2001. Carnegie Mellon University, Pittsburgh, PA, USA, p.192-199.

Li, J.M., Zhang, B., Lin, F.Z., 2003. Training algorithms for Support Vector Machines. *Journal of Tsinghua Univ. (Sci. & Tech.)*, **43**:32-37 (in Chinese).

Magerman, D., 1995. Statistical Decision Tree Model for Parsing. Proceedings 33rd Annual Meeting of Association for Computational Linguistics. Massachusetts, Cambridge, p.276-283.

Zhang, Y.Q., Zhou, Q., 2002. Automatic identification of Chinese Base Phrase. *Journal of Chinese Information Processing*, **16**:1-8 (in Chinese).