Multi-lingual Dependency Parsing at NAIST

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

In this paper, we present a framework for multi-lingual dependency parsing. Our bottom-up deterministic parser adopts Nivre's algorithm (Nivre, 2004) with a preprocessor. Support Vector Machines (SVMs) are utilized to determine the word dependency attachments. Then, a maximum entropy method (MaxEnt) is used for determining the label of the dependency relation. To improve the performance of the parser, we construct a tagger based on SVMs to find neighboring attachment as a preprocessor. Experimental evaluation shows that the proposed extension improves the parsing accuracy of our base parser in 9 languages. (Hajič et al., 2004; Simov et al., 2005; Simov and Osenova, 2003; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit and Martí, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003).

1 Introduction

The presented dependency parser is based on our preceding work (Cheng, 2005a) for Chinese. The parser is a bottom-up deterministic dependency parser based on the algorithm proposed by (Nivre, 2004). A dependency attachment matrix is constructed, in which each element corresponds to a pair of tokens. Each dependency attachment is incrementally constructed, with no crossing constraint. In the parser, SVMs (Vapnik, 1998) deterministically estimate whether a pair of words has either of four relations: right, left, shift and reduce. While dependency attachment is estimated by SVMs, we use a MaxEnt (Ratnaparkhi, 1999) based tagger with the output of the parser to estimate the label of dependency relations. This tagger uses the same features as for the word dependency analysis.

In our preceding work (Cheng, 2005a), we not only adopted the Nivre algorithm with SVMs, but also tried some preprocessing methods. We investigated several preprocessing methods on a Chinese Treebank. In this shared task (Buchholz et. al, 2006), we also investigate which preprocessing method is effective on other languages. We found that only the method that uses a tagger to extract the word dependency attachment between two neighboring words works effectively in most of the languages.

2 System Description

The main part of our dependency parser is based on Nivre's algorithm (Nivre, 2004), in which the dependency relations are constructed by a bottom-up deterministic schema. While Nivre's method uses memory-based learning to estimate the dependency attachment and the label, we use SVMs to estimate the attachment and MaxEnt to estimate

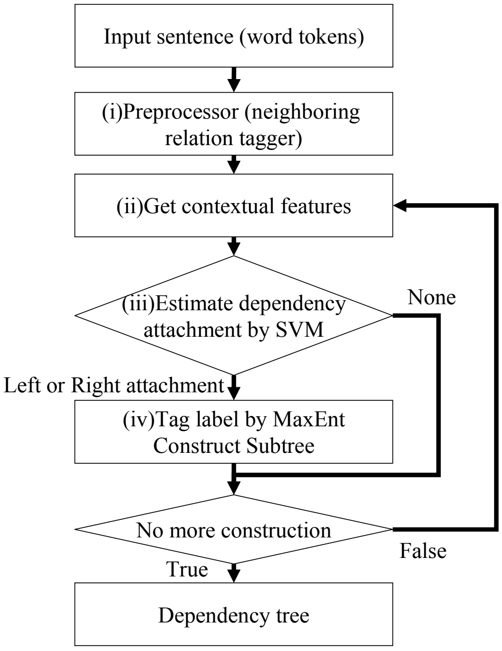
Input sentence (word tokens)
(i)Preprocessor (neighboring relation tagger)
(ii)Get contextual features
(iii)Estimate dependency attachment by SVM
None
Left or Right attachment
(iv)Tag label by MaxEnt Construct Subtree
No more construction
False
True
Dependency tree

Fig. 1 The architecture of our parser

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 191–195, New York City, June 2006. ©2006 Association for Computational Linguistics

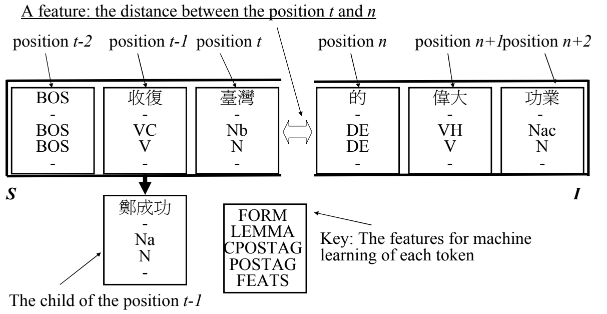

Fig. 2. The features for dependency analysis

the label. The architecture of the parser consists of four major procedures and as in **Fig.1**:

(i) Decide the neighboring dependency attachment between all adjacent words in the input sentence by SVM-based tagger (as a preprocessing)

(ii) Extract the surrounding features for the focused pair of nodes.

(iii) Estimate the dependency attachment operation of the focused pair of nodes by SVMs.

(iv) If there is a left or right attachment, estimate the label of dependency relation by MaxEnt.

We will explain the main procedures (steps (ii)-(iv)) in sections 2.1 and 2.2, and the preprocessing in section 2.3.

2.1 Word dependency analysis

In the algorithm, the state of the parser is 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 attachments decided in the algorithm. Given an input word sequence *W*, the parser is initialized by the triple $\langle nil, W, \phi \rangle$. The parser estimates the dependency attachment between two words (the top elements of stacks *S* and *I*). The algorithm iterates until the list *I* becomes empty. There are four possible operations (**Right**, **Left**, **Shift** and **Reduce**) for the configuration at hand.

Right or **Left:** If there is a dependency relation that the word *t* or *n* attaches to word *n* or *t*, add the new dependency relation $(t \rightarrow n)$ or $(n \rightarrow t)$ into *A*, remove *t* or *n* from *S* or *I*.

If there is no dependency relation between *n* and *t,* check the following conditions.

Reduce: If there is no word n' ($n' \in I$) which may depend on *t*, and *t* has a parent on its left side, the parser removes *t* from the stack *S*.

Shift: If there is no dependency between *n* and *t,* and the triple does not satisfy the conditions for **Reduce**, then push *n* onto the stack *S*.

In this work, we adopt SVMs for estimating the word dependency attachments. SVMs are binary classifiers based on the maximal margin strategy. We use the polynomial kernel: $K(\mathbf{x},\mathbf{z}) = (1+\mathbf{x}\cdot\mathbf{z})^d$ with d =2. The performance of SVMs is better than that of the maximum entropy method in our preceding work for Chinese dependency analysis (Cheng, 2005b). This is because that SVMs can combine features automatically (using the polynomial kernel), whereas the maximum entropy method cannot. To extend binary classifiers to multi-class classifiers, we use the pair-wise method, in which we make $_nC_2$ [1] binary classifiers between all pairs of the classes (Kreβel, 1998). We use Libsvm (Lin et al., 2001) in our experiments.

In our method, the parser considers the dependency attachment of two nodes (*n,t*). The features of a node are the word itself, the POS-tag and the information of its child node(s). The context features 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. The features are shown in **Fig.2**.

2.2 Label tagging

We adopt MaxEnt to estimate the label of dependency relations. We have tried to use linear-chain conditional random fields (CRFs) for estimating the labels after the dependency relation analysis. This means that the parser first analyzes the word dependency (head-modifier relation) of the input sentence, then the CRFs model analyzes the most suitable label set with the basic information of input sentence (FORM, LEMMA, POSTAG……etc) and the head information (FORM and POSTAG) of each word. However, as the number of possible labels in some languages is large, training a CRF model with these corpora (we use CRF++ (Kudo, 2005)) cost huge memory and time.

Instead, we combine the maximum entropy method in the word dependency analysis to tag the label of dependency relation. As shown in **Fig. 1**, the parser first gets the contextual features to estimate the word dependency. If the parsing operation

[1] To estimate the current operation (Left, Right, Shift and Reduce) by SVMs, we need to build 6 classifiers(Left-Right, Left-Shift, Left-Reduce, Right-Shift, Right-Reduce and Shift-Reduce).

is “Left” or “Right”, the parser then use MaxEnt with the same features to tag the label of relation. This strategy can tag the label according to the current states of the focused word pair. We divide the training instances according to the CPOSTAG of the focused word *n*, so that a classifier is constructed for each of distinct POS-tag of the word *n*.

2.3 Preprocessing

2.3.1 Preceding work

In our preceding work (Cheng, 2005a), we discussed three problems of our basic methods (adopt Nivre’s algorithm with SVMs) and proposed three preprocessing methods to resolve these problems. The methods include: (1) using global features and a two-steps process to resolve the ambiguity between the parsing operations “Shift” and “Reduce”. (2) using a root node finder and dividing the sentence at the root node to make use of the top-down information. (3) extracting the prepositional phrase (PP) to resolve the problem of identifying the boundary of PP.

We incorporated Nivre’s method with these preprocessing methods for Chinese dependency analysis with Penn Chinese Treebank and Sinica Treebank (Chen et al., 2003). This was effective because of the properties of Chinese: First, there is no multi-root in Chinese Treebank. Second, the boundary of prepositional phrases is ambiguous. We found that these methods do not always improve the accuracy of all the languages in the shared task.

We have tried the method (1) in some languages to see if there is any improvement in the parser. We attempted to use global features and two-step analysis to resolve the ambiguity of the operations. In Chinese (Chen et al., 2003) and Danish (Kromann, 2003), this method can improve the parser performance. However, in other languages, such as Arabic (Hajič et al., 2004), this method decreased the performance. The reason is that the sentence in some languages is too long to use global features. In our preceding work, the global features include the information of all the un-analyzed words. However, for analyzing long sentences, the global features usually include some useless information and will confuse the two-step process. Therefore, we do not use this method in this shared task.

In the method (2), we construct an SVM-based root node finder to identify the root node and divided the sentence at the root node in the Chinese Treebank. This method is based on the properties of dependency structures “One and only one element is independent” and “An element cannot have modifiers lying on the other side of its own head”. However, there are some languages that include multi-root sentences, such as Arabic, Czech, and Spanish (Civit and Martí, 2002), and it is difficult to divide the sentence at the roots. In multi-root sentences, deciding the head of the words between roots is difficult. Therefore, we do not use the method (2) in the share task.

The method (3) –namely PP chunker– can identify the boundary of PP in Chinese and resolve the ambiguity of PP boundary, but we cannot guarantee that to identify the boundary of PP can improve the parser in other languages. Even we do not understand construction of PP in all languages. Therefore, for the robustness in analyzing different languages, we do not use this method.

2.3.2 Neighboring dependency attachment tagger

In the bottom-up dependency parsing approach, the features and the strategies for parsing in early stage (the dependency between adjacent[2] words) is different from parsing in upper stage (the dependency between phrases). Parsing in upper stage needs the information at the phrases not at the words alone. The features and the strategies for parsing in early and upper stages should be separated into distinct. Therefore, we divide the neighboring dependency attachment (for early stage) and normal dependency attachment (for upper stage), and set the neighboring dependency attachment tagger as a preprocessor.

When the parser analyzes an input sentence, it extracts the neighboring dependency attachments first, then analyzes the sentence as described before. The results show that tagging the neighboring dependency word-pairs can improve 9 languages out of 12 scoring languages, although in some languages it degrades the performance a little. Potentially, there may be a number of ways for decomposing the parsing process, and the current method is just the simplest decomposition of the process. The best method of decomposition or dynamic changing of parsing models should be investigated as the future research.

[2] We extract all words that depend on the adjacent word (right or left).

3 Experiment

3.1 Experimental setting

Our system consists of three parts; first, the SVM-based tagger extracts the neighboring attachment relations of the input sentence. Second, the parser analyzes further dependency attachments. If a new dependency attachment is generated, the MaxEnt based tagger estimates the label of the relation. The three parts of our parser are trained on the available data of the languages.

In our experiment, we used the full information of each token (FORM, LEMMA, CPOSTAG, POSTAG, FEATS) when we train and test the model. **Fig. 2** describes the features of each token. Some languages do not include all columns; such that the Chinese data does not include LEMMA and FEATURES, these empty columns are shown by the symbol "-" in **Fig. 2**. The features for the neighboring dependency tagging are the information of the focused word, two preceding words and two succeeding words. **Fig. 2** shows the window size of our features for estimating the word dependency in the main procedures. These features include the focused words (*n*, *t*), two preceding words and two succeeding words and their children. The features for estimating the relation label are the same as the features used for word dependency analysis. For example, if the machine learner estimates the operation of this situation as "**Left**" or "**Right**" by using the features in **Fig. 2**, the parser uses the same features in **Fig. 2** and the dependency relation to estimate the label of this relation.

For training the models efficiently, we divided the training instances of all languages at the CPOSTAG of the focused word *n* in **Fig .2.** In our preceding work, we found this procedure can get better performance than training with all the instances at once. However, only the instances in Czech are divided at the CPOSTAG of the focused word-pair *t*-*n*[3]. The performance of this procedure is worse than using the CPOSTAG of the focused word *n*, because the training instances of each CPOSTAG-pair will become scarce. However, the data size of Czech is much larger than other languages; we couldn't finish the training of Czech using the CPOSTAG of the focused word *n*, before the deadline for submitting. Therefore we used this procedure only for the experiment of Czech.

All our experiments were run on a Linux machine with XEON 2.4GHz and 4.0GB memory. The program is implemented in JAVA.

3.2 Results

Table 1 shows the results of our parser. We do not take into consideration the problem of cross relation. Although these cross relations are few in training data, they would make our performance worse in some languages. We expect that this is one reason that the result of Dutch is not good. The average length of sentences and the size of training data may have affected the performance of our parser. Sentences of Arabic are longer and training data size of Arabic is smaller than other languages; therefore our parser is worse in Arabic. Similarly, our result in Turkish is also not good because the data size is small.

We compare the result of Chinese with our preceding work. The score of this shared task is better than our preceding work. It is expected that we selected the FORM and CPOSTAG of each nodes as features in the preceding work. However, the POSTAG is also a useful feature for Chinese, and we grouped the original POS tags of Sinica Treebank from 303 to 54 in our preceding work. The number of CPOSTAG(54) in our preceding work is more than the number of CPOSTAG(22) in this shared task, the training data of each CPOSTAG in our preceding work is smaller than in this work. Therefore the performance of our preceding work in Sinica Treebank is worse than this task.

The last column of the **Table 1** shows the unlabeled scores of our parser without the preprocessing. Because our parser estimates the label after the dependency relation is generated. We only consider whether the preprocessing can improve the unlabeled scores. Although the preprocessing can not improve some languages (such as Chinese, Spanish and Swedish), the average score shows that using preprocessing is better than parsing without preprocessing.

Comparing the gold standard data and the system output of Chinese, we find the CPOSTAG with lowest accuracy is "P (preposition)", the accuracy that both dependency and head are correct is 71%. As we described in our preceding work and Section 2.3, we found that boundaries of prepositional phrases are ambiguous for Chinese. The bottom-up algorithm usually wrongly parses the prepositional phrase short. The parser does not capture the correct information of the children of the preposition. According to the results, this problem does not cause the accuracy of head of

[3] For example, we have 15 SVM-models for Arabic according to the CPOSTAG of Arabic (A, C, D, F, G…etc.). However, we have 139 SVM-models for Czech according the CPOSTAG pair of focused words (A-A, A-C, A-D…etc.)

Language:	LAS:	UAS:	LAcc.	UAS with out preprocessing:
Arabic	65.19	77.74	79.02	76.74
Chinese	84.27	89.46	86.42	90.03
Czech	76.24	83.4	83.52	82.88
Danish	81.72	88.64	86.11	88.45
Dutch	71.77	75.49	75.83	74.97
German	84.11	87.66	90.67	87.53
Japanese	89.91	93.12	92.40	92.99
Portugese	85.07	90.3	88.00	90.21
Slovene	71.42	81.14	80.96	80.43
Spanish	80.46	85.15	88.90	85.19
Swedish	81.08	88.57	83.99	88.83
Turkish	61.22	74.49	73.91	74.3
AV:	77.7	84.6	84.1	84.38
SD:	8.67	6.15	5.78	6.42
Bulgarian	86.34	91.3	89.27	91.44

Table 1: Results

CPOSTAG “P” decrease. Actually, the head accuracy of “P” is better than the CPOSTAG “C” or “V”. However, the dep. accuracy of “P” is worse. We should consider the properties of prepositions in Chinese to resolve this question. In Chinese, prepositions are derived from verbs; therefore some prepositions can be used as a verb. Naturally, the dependency relation of a preposition is different from that of a verb. Important information for distinguishing whether the preposition is a verb or a preposition is the information of the children of the preposition. The real POS tag of a preposition which includes few children is usually a verb; on the other hand, the real POS tag of a preposition is usually a preposition.

If our parser considers the preposition which leads a short phrase, the parser will estimate the relation of the preposition as a verb. At the same time, if the boundary of prepositional phrase is analyzed incorrectly, other succeeding words will be wrongly analyzed, too.

Error analysis of Japanese data (Kawata and Bartels, 2000) shows that CNJ (Conjunction) is a difficult POS tag. The parser does not have any module to detect coordinate structures. (Kurohashi, 1995) proposed a method in which coordinate structure with punctuation is detected by a coefficient of similarity. Similar framework is necessary for solving the problem.

Another characteristic error in Japanese is seen at adnominal dependency attachment for a compound noun. In such dependency relations, adjectives and nouns with "no" (genitive marker) can be a dependent and compound nouns which consist of more than one consecutive nouns can be a head. The constituent of compound nouns have same POSTAG, CPOSTAG and FEATS. So, the machine learner has to disambiguate the dependency attachment with sparce feature LEMMA and FORM. Compound noun analysis by semantic feature is necessary for addressing the issue.

4 Conclusion

This paper reported on multi-lingual dependency parsing on combining SVMs and MaxEnt. The system uses SVMs for word dependency attachment analysis and MaxEnt for the label tagging when the new dependency attachment is generated. We discussed some preprocessing methods that are useful in our preceding work for Chinese dependency analysis, but these methods, except one, cannot be used in multi-lingual dependency parsing. Only using the SVM-based tagger to extract the neighbor relation could improve many languages in our experiment, therefore we use the tagger in the parser as its preprocessing.

References

S. Buchholz, E. Marsi, A. Dubey and Y. Krymolowski. 2006. *CoNLL-X: Shared Task on Multilingual Dependency Parsing*, CoNLL 2006.

Yuchang Cheng, Masayuki Asahara and Yuji Matsumoto. 2005a. *Chinese Deterministic Dependency Parser: Examining Effects of Global Features and Root Node Finder*, Fourth SIGHAN Workshop, pp.17-24.

Yuchang Cheng, Masayuki Asahara and Yuji Matsumoto. 2005b. *Machine Learning-based Dependency Parser for Chinese*, the International Conference on Chinese Computing, pp.66-73.

Ulrich. H.-G. Kreβel, 1998. *Pairwise classification and support vector machines*. In Advances in Kernel Methods, pp. 255-268. The MIT Press.

Taku Kudo. *CRF++: Yet Another CRF toolkit*, http://www.chasen.org/~taku/software/CRF++/.

Sadao Kurohashi. 1995. *Analyzing Coordinate Structures Including Punctuation in English*, In IWPT-95, pp. 136-147.

Chih Jen Lin, 2001. *A practical guide to support vector classification*, http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Joakim Nivre, 2004. *Incrementality in Deterministic Dependency Parsing*, In Incremental Parsing: Bringing Engineering and Cognition Together. Workshop at ACL-2004, pp. 50-57.

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

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