The Exploration of Deterministic and Efficient Dependency Parsing

Yu-Chieh Wu
Dept. of Computer Science and Information Engineering
National Central University
Taoyuan, Taiwan
bcbb@db.csie.ncu.edu.tw

Yue-Shi Lee
Dept. of Computer Science and Information Engineering
Ming Chuan University
Taoyuan, Taiwan
lees@mcu.edu.tw

Jie-Chi Yang
Graduate Institute of Network Learning Technology
National Central University
Taoyuan, Taiwan
yang@cl.ncu.edu.tw

Abstract

In this paper, we propose a three-step multilingual dependency parser, which generalizes an efficient parsing algorithm at first phase, a root parser and post-processor at the second and third stages. The main focus of our work is to provide an efficient parser that is practical to use with combining only lexical and part-of-speech features toward language independent parsing. The experimental results show that our method outperforms Maltparser in 13 languages. We expect that such an efficient model is applicable for most languages.

1 Introduction

The target of dependency parsing is to automatically recognize the head-modifier relationships between words in natural language sentences. Usually, a dependency parser can construct a similar grammar tree with the dependency graph. In this year, CoNLL-X shared task (Buchholz et al., 2006) focuses on multilingual dependency parsing without taking the language-specific knowledge into account. The ultimate goal of this task is to design an ideal multilingual portable dependency parsing system.

To accomplish the shared task, we present a very light-weight and efficient parsing model to the 13 distinct treebanks (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; Nivre et al., 2005; Oflazer et al., 2003; Atalay et al., 2003) with a three-step process, Nivre's algorithm (Nivre, 2003), root parser, and post-processing. Our method is quite different from the conventional three-pass processing, which usually exhaustively processes the whole dataset three times, while our method favors examining the "un-parsed" tokens, which incrementally shrink. At the beginning, we slightly modify the original parsing algorithm (proposed by (Nivre, 2003)) to construct the initial dependency graph. A root parser is then used to recognize root words, which were not parsed during the previous step. At the third phase, the post-processor (which is another learner) recognizes the still un-parsed words. However, in this paper, we aim to build a multilingual portable parsing model without employing deep language-specific knowledge, such as lemmatization, morphologic analyzer etc. Instead, we only make use of surface lexical and part-of-speech (POS) information. Combining these shallow features, our parser achieves a satisfactory result for most languages, especially Japanese.

In the remainder of this paper, Section 2 describes the proposed parsing model, and Section 3 lists the experimental settings and results. Section 4 presents the discussion and analysis of our parser with three selected languages. In Section 5, we draw the future direction and conclusion.

2 System Description

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 241–245, New York City, June 2006.

Over the past decades, many state-of-the-art parsing algorithm were proposed, such as head-word lexicalized PCFG (Collins, 1998), Maximum Entropy (Charniak, 2000), Maximum/Minimum spanning tree (MST) (McDonald et al., 2005), Bottom-up deterministic parsing (Yamada and Matsumoto, 2003), and Constant-time deterministic parsing (Nivre, 2003). Among them, the Nivre's algorithm (Nivre, 2003) was shown to be most efficient method, which only costs at most 2n transition actions to parse a sentence (O(n^3) for the bottom-up or MST approaches). Nivre's method is mainly consists of four transition actions, Left/Right/Reduce/Shift. We further extend these four actions by dividing the "reduce" into "reduce" and "sleep (reduce-but-shift)" two actions. Because the too early reduce action makes the following words difficult to find the parents. Thus, during training, if a word which is the child of the top of the stack, it is then assigned to the "sleep" category and pushed into stack, otherwise, the conventional reduce action is applied. Besides, we do not arrange these transition actions with priority order, instead, the decision is made by the classifier. The overall parsing model can be found in Figure 1. Table 1 lists the detail system spec of our model.

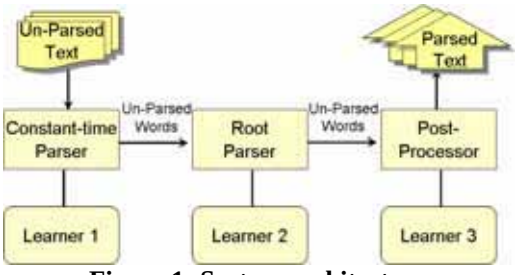

Figure 1: System architecture

Table 1: Overall parsing system summary

Ⅰ.	Parsing Algorithm:	1. Nivre's Algorithm (Nivre, 2003) 2. Root Parser 3. Exhaustive-based Post-processing
Ⅱ.	Parser Characteristics:	1. Top-down + Bottom-up 2. Deterministic + Exhaustive 3. Labeling integrated 4. Non-Projective
Ⅲ.	Learner:	SVMLight (Joachims, 1998) (1) One-versus-One (2) Linear Kernel
Ⅳ.	Feature Set:	1. Lexical (Unigram/Bigram) 2. Fine-grained POS and Coarse grained BiCPOS
Ⅴ.	Post-Processing:	Another learner is used to re-recognize heads in stacks
Ⅵ.	Additional/External Resources:	Non-Used

2.1 Constant-time Parser and Analysis

The Nivre's algorithm makes use of a stack and an input list to model the word dependency relations via identifying the transition action of the top token on the stack (*Top*) and the next token of the input list (*Next*). Typically a learning algorithm can be used to recognize these actions via encoding features of the two terms (*Top* and *Next*). The "Left" and "Reduce" pops the *Top* from stack whereas the "Right", "Reduce-But-Shift", and "Shift" push token *Next* into the top of stack. Nivre (Nivre, 2003) had proved that this algorithm can accomplish dependency parsing at most 2*n* transition actions.

Although, the Nivre's algorithm is much more efficient than the others, it produces three problems.

1. It does not explicitly indicate which words are the roots.
2. Some of the terms in the stack do not belong to the root but still should be parsed.
3. It always only compares the *Top* and *Next* words.

The problem (2) and (3) are complement with each other. A straightforward way resolution is to adopt the exhaustive parsing strategy (Covington, 2001). Unfortunately, such a brute-force way may cause exponential training and testing spaces, which is impractical to apply to the large-scale corpus, for example, the Czech Treebank (1.3 million words). To overcome this and keep the efficiency, we design a post-processor that re-cycles the residuum in the stack and re-identify the heads of them. Since most of the terms (90-95%) of the terms had be processed in previous stages, the post-processor just exhaustively parses a small part. In addition, for problem (1), we propose a root parser based on the parsed result of the Nivre's algorithm. We discuss the root-parser and post-processor in the next two subsections.

2.2 Root Parser

After the first stage, the stack may contain root and un-parsed words. The root parser identifies the root word in the stack. The main advantage of this strategy could avoid sequential classification process, which only focuses on terms in the stack.

We build a classifier, which learns to find root word based on encoding context and children features. However, most of the dependency relations were constructed at the first stage. Thus, we have more sufficient head-modifier information rather

than only taking the contexts into account. The used features are listed as follows.

Neighbor terms,bigrams,POS,BiCPOS (+/-2 window)
Left most child term, POS, Bigram, BiCPOS
Right most child term, POS, Bigram, BiCPOS

2.3 Post-Processing

Before post-processing, we remove the root words from stack, which were identified by root-parser. The remaining un-parsed words in stack were used to construct the actual dependency graph via exhaustive comparing with parsed-words. It is necessary to build a post-processor since there are about 10% un-parsed words in each training set. We provide the un-parsed rate of each language in Table 2 (the r.h.s. part).

By applying previous two steps (constant-time parser and root parser) to the training data, the remaining un-parsed tokens were recorded. Not only using the forward parsing direction, the backward direction is also taken into account in this statistics. Averagely, the un-parsed rates of the forward and backward directions are 13% and 4% respectively. The back ward parsing often achieves lower un-parsed rate among all languages (except for Japanese and Turkish).

To find the heads of the un-parsed words, we copy the whole sentence into the word list again, and re-compare the un-parsed tokens (in stack) and all of the words in the input list. Comparing with the same words is disallowed. The comparing process is going on until the actual head is found. Acquiescently, we use the nearest root words as its head. Although such a brute force way is time-consuming. However, it only parses a small part of un-parsed tokens (usually, 2 or 3 words per sentence).

2.4 Features and Learners

For the constant-time parser of the first stage, we employ the features as follows.

Basic features:

Top.word,*Top*.pos,*Top*.lchild.pos,*Top*.lchild.relation,
Top.rchild.pos, *Top*.rchild.relation,*Top*.head.pos,
Top.head.relation,
Next.word, *Next*.pos, *Next*.lchild.pos,
Next.lchild.relation, $Next_{+1}$.pos, $Next_{+2}$.pos, $Next_{+3}$.pos

Enhanced features:

Top.bigram,*Top*.bicpos,*Next*.bigram,*Next*.bicpos,
$Next_{+1}$.word,$Next_{+2}$.word,$Next_{+3}$.word

In this paper, we use the support vector machines (SVM) (Joachims, 1998) as the learner. SVM is widely used in many natural language processing (NLP) areas, for example, POS tagging (Wu et al., 2006). However, the SVM is a binary classifier which only recognizes true or false. For multiclass problem, we use the so-called one-versus-one (OVO) method with linear kernel to combine the results of each pairwise subclassifier. The final class in testing phase is mainly determined by majority voting.

For all languages, our parser uses the same settings and features. For all the languages (except Japanese and Turkish), we use backward parsing direction to keep the un-parsed token rate low.

3 Experimental Result

3.1 Dataset and Evaluation Metrics

The testing data is provided by the (Buchholz et al., 2006) which consists of 13 language treebanks. The experimental results are mainly evaluated by the unlabeled and labeled attachment scores. The CoNLL also provided a perl-scripter to automatic compute these rates.

3.2 System Results

Table 2 presents the overall parsing performance of the 13 languages. As shown in Table 2, we list two parsing results at the second and third columns (new and old). It is worth to note that the result B is produced by removing the enhanced features and the post-processing step from our parser, while the result A is the complete use of the enhanced features and the overall three-step parsing. In this year, we submit result B to the CoNLL shared task due to the time limitation.

In addition, we also apply the Maltparser, which is implemented with the Nivre's algorithm (Nivre, 2003) to be compared. The Maltpaser also includes the SVM and memory-based learner (MBL). Nevertheless, it does not optimize the SVM where the training and testing times are too long to be compared even the linear kernel is used. Therefore we use the default MBL and feature model 3 (M3) in this experiment. We also perform the significant test to evaluate the statistical difference among the three results. If the answer is "Yes", it means the two systems are significant difference under at least 95% confidence score ($p < 0.05$).

Table 2: A general statistical table of labeled attachment score, test and un-parsed rate (percentage)

	A (New result)	B (Old result)	C (Maltparser)	Statistic test			Un-Parsed Rate	
				A vs. B	B vs. C	A vs. C	Forward	Backward
Arabic	63.75	**63.81**	54.11	No	Yes	Yes	10.3	**1.4**
Chinese	**81.25**	74.81	73.92	Yes	No	Yes	4.01	**2.3**
Czech	**71.24**	59.36	59.36	Yes	No	Yes	16.1	**5.6**
Danish	**79.52**	78.38	77.31	No	No	No	12.8	**2.5**
Dutch	68.45	**68.45**	63.61	No	Yes	Yes	18.4	**9.8**
German	**79.57**	76.52	76.52	Yes	No	Yes	12.7	**9.2**
Japanese	**91.43**	90.11	89.07	Yes	No	Yes	**1.1**	4.4
Portugese	81.33	**81.47**	75.38	No	Yes	Yes	24.3	**3.17**
Slovene	**68.41**	67.83	55.04	No	Yes	Yes	14.9	**5.5**
Spanish	**74.65**	72.99	72.81	Yes	No	Yes	20	**0.5**
Swedish	**79.53**	71.72	76.28	Yes	Yes	Yes	19.1	**2.8**
Turkish	**55.33**	55.09	52.18	No	Yes	Yes	**2.5**	4
Bulgarian	**81.23**	79.73	79.73	No	No	No	15.7	**1.2**
AVG	75.05	72.32	69.64				13.22	4.02

4 Discussion

4.1 Analysis of Overview Aspect

Although our method is efficient for parsing that achieves satisfactory result, it is still away from the state-of-the-art performance. Many problems give rise to not only the language-specific characteristics, but also the parsing strategy. We found that our method is weak to the large-scale training size and large dependency class datasets, for example, German (Brants et al., 2002) and Czech. For Dutch, we observe that the large non-projective tokens and relations in this set. Overall, we conclude the four main limitations of our parsing model.

1. Unbalanced and large dependency relation classes
2. Too fine or coarse POS tag
3. Long sentences and non-projective token rates
4. Feature engineering and root accuracy

The main reason of the first problem is still caused by the unbalanced distribution of the training data. Usually, the right-action categories obtain much fewer training examples. For example, in the Turkish data, 50 % of the categories receive less than 0.1% of the training examples, 2/3 are the right dependency group. For the Czech, 74.6% of the categories receive less than 0.1% of the training examples.

Second, the too fine grained size of POS tag set often cause the features too specific that is difficult to be generalized by the learner. Although we found the grained size is not the critical factor of our parser, it is closely related to the fourth problem, feature engineering. For example, in Chinese (Chen et al., 2003), there are 303 fine grained POS types which achieves better result on the labeled attachment score is higher than the coarse grained (81.25 vs. 81.17). Intuitively, the feature combinations deeply affect the system performance (see A vs. C where we extend more features than the original Nivre's algorithm).

Problem 3 exposes the disadvantage of our method, which is weak to identify the long distance dependency. The main reason is resulted from the Nivre's algorithm in step 1. This method is quite sensitive and non error-recovered since it is a deterministic parsing strategy. Abnormal or wrong push or pop actions usually cause the error propagation to the remaining words in the list. For example, there are large parts of errors are caused by too early reduce or missed left arc makes some words could not find the actual heads. On the contrary, one can use an *N*-best selection to choose the optimal dependency graph or applying MST or exhaustive parsing schema. Usually, these approaches are quite inefficient which requires at least $O(n^3)$.

Finally, in this paper, we only take the surface lexical word and POS tag into account without employing the language-specific features, such as Lemma, Morph…etc. Actually, it is an open question to compile and investigate the feature engineering. On the other hand, we also find the performance of the root parser in some languages is poor. For example, for Dutch the root precision rate is only 38.52, while the recall rate is 76.07. It indicates most of the words in stack were wrongly recognized as root. This is because there are substantially un-parsed rate that left many un-parsed words remain in stack. One way to remedy the problem can adjust the root parser to independently identify root word by sequential word classification at first step and then apply the Nivre's algorithm. We left the comparison of the issue as future work.

4.2 Analysis of Specific View

We select three languages, Arabic, Japanese, and Turkish to be more detail analysis. Figure 2 illustrates the learning curve of the three languages and Table 3 summarizes the comparisons of "fine vs. coarse" POS types and "forward vs. backward" parsing directions.

For the three languages, we found that most of the errors frequently appear to the noun POS tags which often denominate half of the training set. In Turkish, the lower performance on the noun POS attachment rate deeply influents the overall parsing. For example, the error rate of Noun in Turkish is 39% which is the highest error rate. On the contrary, the head error rates fall in the middle rank for the other two languages.

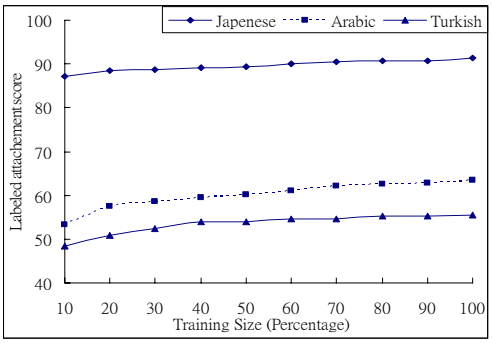

Figure 2: Learning curve of the three datasets

Table 3: Parsing performance of different grained POS tags and forward/backward parsing directions

	Parsing direction	LA-Score		POS grained	LA-Score
Ja	**Forward**	**91.35**	Forward	**Fine**	**91.35**
	Backward	85.75		Coarse	91.25
Ar	Forward	60.62	Backward	Fine	63.55
	Backward	**63.55**		**Coarse**	**63.63**
Tu	Forward	55.47	Forward	Fine	55.47
	Backward	**55.59**		**Coarse**	**55.59**

In Turkish, we also find an interesting result where the recall rate of the distance=2 parsing (56.87) is lower than distance=3-6, and >7 (62.65, 57.83). In other words, for Turkish, our parser failed to recognize the distance=2 dependency relations. For the other languages, usually the identification rate of the longer distance parsing should be lower than the smaller distance. Thus, a future work to parsing Turkish, should put more emphasis on improving not only the noun POS type, but also the distance=2 parsing.

Besides, the root parsing accuracy is also an important factor to most languages. In Japanese, although our parser achieves more than 97% left/right arc rates. However, for the root word precision rate is quite lower (85.97). Among all dependency relation classification rates, the root class usually locates in the lowest rank for the three languages.

5 Conclusion and Future Remarks

Dependency parsing is one of the most important issues in NLP community. This paper presents and analyzes the impact of the efficient parsing model that only combines with lexical and part-of-speech information. To go language-independent, we did not tune any parameter settings in our model and exclude most of the language-dependent feature set, which provided by the CoNLL (Buchholz et al., 2006). The main focus of our work coincides with the target goal of the CoNLL shared task, i.e., go multilingual dependency parsing without taking the language-specific knowledge into account. A future work on the deterministic parsing strategy is to convert the existing model toward *N*-best parsing.

References

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing, *In Proceedings of the Tenth Conf. on Computational Natural Language Learning CoNLL-X*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. *In NAACL*, pages 132-139.

Michael Collins. 1998. Head-driven statistical models for natural language processing. Ph.D. thesis. University of Pennsylvania.

Michael A. Covington. 2001. A fundamental Algorithm for Dependency Parsing. *In Proceedings of the Annual ACM Southeast Conference*, pages 95-102.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. *In COLING*, pages 340-345.

Thornsten Joachims. 1998. Text categorization with support vector machines: learning with many relevant features. *In ECML*, pages 137-142.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers, *In ACL*, pages 91-98.

Joakim Nivre. 2003. An Efficient Algorithm for Projective Dependency Parsing. *In Proceedings of the International Workshop on Parsing Technology*, pages 149-160.

Yu C. Wu, Chia H. Chang, and Yue S. Lee. 2006. A General and Multi-lingual Phrase Chunking Model based on Masking Method. *In CICLING*, pages 144-155.

Hiroyasu Yamada, and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. *In Proceedings of the International Workshop on Parsing Technology*, pages 195-206.