# Multilingual Deterministic Dependency Parsing Framework using Modified Finite Newton Method Support Vector Machines

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

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

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

## Abstract

In this paper, we present a three-step multilingual dependency parser based on a deterministic shift-reduce parsing algorithm. Different from last year, we separate the root-parsing strategy as sequential labeling task and try to link the neighbor word dependences via a near neighbor parsing. The outputs of the root and neighbor parsers were encoded as features for the shift-reduce parser. In addition, the learners we used for the two parsers and the shift-reduce parser are quite different (conditional random fields and the modified finite-Newton method support vector machines). We found that our method could benefit from the two-preprocessing stages. To speed up training, in this year, we employ the MFN-SVM (modified finite-Newton method support vector machines) which can be learned in linear time. The experimental results show that our method achieved the middle rank over the 23 teams. We expect that our method could be further improved via well-tuned parameter validations for different 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-2007 shared task (Nivre et al., 2007) focuses on multilingual dependency parsing based on ten different languages (Hajic et al., 2004; Aduriz et al., 2003; Martí et al., 2007; Chen et al., 2003; Böhmova et al., 2003; Marcus et al., 1993; Johansson and Nugues, 2007; Prokopidis et al., 2005; Czendes et al., 2005; Montemagni et al., 2003; Oflazer et al., 2003) and domain adaptation for English (Marcus et al., 1993; Johansson and Nugues, 2007; Kulick et al., 2004; MacWhinney, 2000; Brown, 1973) without taking the language-specific knowledge into consideration. The ultimate goal of them is to design ideal multilingual and domain portable dependency parsing systems.

To accomplish the multilingual and domain adaptation tasks, we present a three-pass parsing model based on a shift-reducing algorithm (Yamada and Matsumoto, 2003; Chang et al., 2006), namely, neighbor parsing, root relation parsing, and shift-reduce parsing. Our method favors examining the "un-parsed" tokens, which incrementally shrink. At the beginning, the parsing direction is mainly determined by the amount of un-parsed tokens in the sentence with either forward or backward parse. In this step, the projective parsing method can be used to evaluate most of the non-projective Treebank datasets. Once the direction is determined, the pseudo-projectivize transformation algorithm (Nivre and Nilsson, 2005) converts most non-projective training data into projective and decodes the parsed text into non-projective. Hereafter, both neighbor-parser and root-parser were trained to discovery additional features for the downstream shift-reduce parse model. We found that the two additional features could improve the performance. Subsequently, the modified shift-reduce parsing algorithm starts to parse the final dependencies with two-pass processing, i.e., predict parse action and label the relations.



*Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pp. 1175–1181,
Prague, June 2007. 

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. In Section 5, we draw the future direction and conclusion.

## 2 System Description

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), shift-reduce-based deterministic parsing (Yamada and Matsumoto, 2003; Chang et al., 2006; Nivre, 2003). Among them, the shift-reduce methods were shown to be the most efficient method, which only costs at most 2n~3n actions to parse a sentence (Chang et al., 2006; Nivre, 2003). Chang et al. (2006) further added the "wait-right" action to the words that had children and could not be reduced in current state. This could avoid the so-called "too early reduce" problems.

The overall parsing model can be found in Figure 1. Figure 2 illustrates the detail system spec of our parsing model.

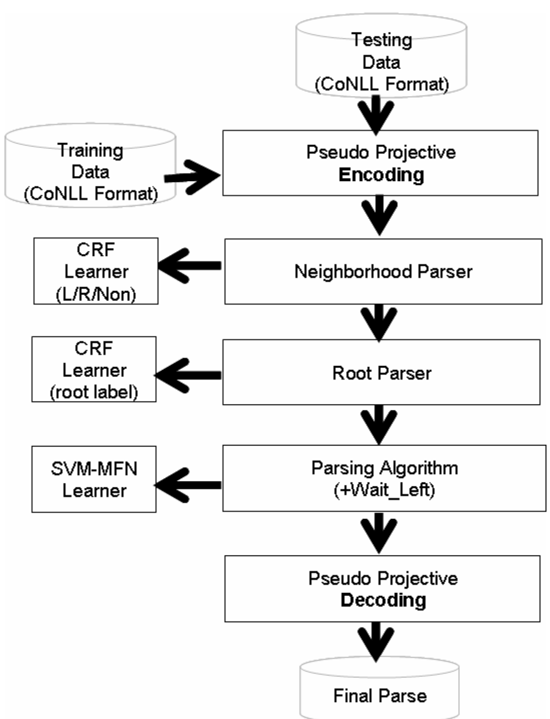


**Figure 1: System architecture**

### 2.1 Neighbor Parser

As shown in Figure 1, the first step is to identify the neighbor head-modifier relations between two consecutive words. Cheng et al. (2006) also reported that the use of neighboring dependency attachment tagger enhance the unlabeled attachment scores from 84.38 to 84.6 for 13 languages. Usually, it is the case that the select features are fixed and could not be tuned to capture the second order features (McDonald et al., 2006). At each location, there the focus and next words are always compared. It may fail to link the next and next+1 word pair since the next word might be reduced due to an earlier wrong decision.

| | | |
|---|---|---|
| Ⅰ. | Parsing Algorithm: | 1. Neighbor Parser<br>2. Root Parser<br>3. Shift-Reduce Algorithm (Yamada and Matsumoto, 2003) |
| Ⅱ. | Parser Characteristics: | 1. Deterministic<br>2. two-pass (Labeling separated)<br>3. Pseudo-Projective en(de)-coding (Nivre and Nilsson, 2005) |
| Ⅲ. | Learner: | MFN-SVM<br>(1) One-versus-All<br>(2) Linear Kernel |
| Ⅳ. | Feature Set: | 1. Lexical (Unigram/Bigram)<br>2. Fine-grained POS (and BiPOS)<br>3. Lemma/FEAT used |
| Ⅴ. | Post-Processing: | Non-Used |
| Ⅵ. | Additional/External Resources: | Non-Used |

**Figure 2: System spec**

However, starting parsing based on the result of neighbor parsing is not a good idea since it could produce error propagation problems. Rather, we include the result of our neighbor parsing as features to increase the original feature set. In the preliminary study, we found that the derived features are very useful for most languages.

As conventional sequential tagging problems, such part-of-speech tagging and phrase chunking, we employ the conditional random fields (CRF) as learners (Kudo et al., 2004). The basic idea of the neighbor parsing can be shown in Figure 3.

The first and second colums in Figure 3 represents the basic word and fine-grained POS froms, while the third column indicates if this word has the LH (left-head) or RH (right-head) with associated relations or O (no neighbor head in either left or right neighbor word). The used features are:
Word, fine-grained POS, bigram, and bi-POS with context window = 2(left) and 4(right)

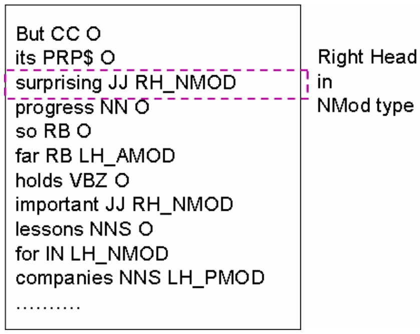


**Figure 3: Sequential tagging model for neighbor parse**

Unfortunately, for some languages, like Chinese and Czech, training with CRF is because of the large number of features and the head relations. To make it practical, we focus on just three types: left head, right head, and out-of-neighbor. This effectively reduces most of the feature space for the CRF. The training time for the neighbor parser with only three categories is less than 5 minutes while it takes three days with taking all the relation tag into account.

### 2.2 Root Parser

After the neighbor parse, the tagged labels are good features for the root parse. In the second stage, the root parser identifies the *root* words in the sentence. Nevertheless, for some languages, such as Arabic and Czech, the roots might be several types as against to Chinese and English in which the number of labels of roots is merely one. Similar to the neighbor parser, we also take the root label into account. As noted, for Chinese and English, the goal of the root parser can be reduced to determine whether the current word is root or not.

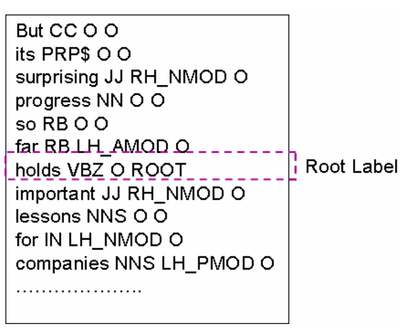


**Figure 4: Sequential tagging model for neighbor parse**

Similar to the neighbor parse, the root parsing task can also be treated as a sequential tagging problem. Figure 4 shows the basic concept of the root parser. The third column is mainly derived from the neighbor parser, while the fourth column represents whether the current word is a root with relation or not.

### 2.3 Parsing Algorithm

After adding the neighbor and root parser output as features, in the final stage, the modified Yamada's shift-reduce parsing algorithm (Yamada and Matsumoto, 2003) is then run. This method is deterministic and can deal with projective data only. There are three basic operation (action) types: Shift (S), Left (L), and Right (R). The operation is mainly determined via the classifier according to the selected features (see 2.4). Each time, the operation is applied to two unparsed words, namely, focus and next. If there exists an arc between the two words (either left or right), then the head of focus or next word is found; otherwise (i.e., shift), next two words are considered at next stage. This method could be economically performed via maintaining two pointers, focus, and next without an explicit stack. The parse operation is iteratively run until no more relation can be found in the sentence.

In 2006, Chang et al. (2006) further reported that the use of "step-back" in comparison to the original "stay". Furthermore, they also add the "wait-left" operations to prevent the "too early reduce" problems. In this way, the parse actions can be reduced to be bound in $3n$ where $n$ is the number of words in a sentence.

Now we compare the adopted parsing algorithm in this year to the one we employed last year (Wu et al., 2006a). The common characteristics are:

1. the same number of parse operations (4)
2. shift-reduce
3. linearly scaled
4. deterministic and projective

On the contrary, their parse actions are quite different. Therefore these two methods have different run time. This gives the two methods rise to different iterative times. The main reason is that the step-back might trace back to previous words, which can be viewed as pop the top words on the stack back to the unparsed strings, while the Nivre's method does not trace-back any two words in the stack. In other words, if a word is pushed into the stack, it will no longer be compared with the other deeper words inside the stack. Hence some of the non-root words in the stack remain to be parsed. A simple solution is to adopt an exhaustive post-processing step for the unparsed words in the stack (details in (Wu et al., 2006a, 2006b)).

A good advantage of the step-back is that it can trace back to the unparsed words in the stack. But theoretically, the required parse actions still more than the Nivre's algorithm (2*n* vs. 3*n*).

By adopting the projectivized en/de-coding over the modified Yamada's algorithm, we can treat the words that do not have a parent as roots. Thus, for some languages (e.g. Czech and Arabic), the multiple root problem can be easily solved. In this year we separate the parse action and the relation label into two stages as opposed to having one pass last year. In this way, we can simply adopt a sequential tagger to auto-assign the relation labels after the whole sentence is parsed.

### 2.4 Features and Learners

Unlike last year, we did separate the action prediction and the label recognition into two stages where the one of the learners could provide more information to another. The used features of the two learners are quite similar and listed as follows:

Basic feature type (for previous 2 and next 3 words):
*Word, POS (fine-grained), Lemma, FEAT, NParse, RParse*

Enhanced feature type:
*Bigram, BiPOS for focus and next words*
*previous two parse actions*

For label recognition:
*Label tag to its head*, *label tags for previous two words*

In this paper, we replicate and modify the modified finite Newton support vector machines (MFN-SVM) (Keerthi and DeCoste, 2005) as the learner.

The MFN-SVM is a very efficient SVM optimization method which linearly scales with the number of training examples. Usually, the trained models from MFN-SVM are quite large that could not be processed in practice. We therefore defined the positive lower bound ($10^{-10}$) and the negative upper bound ($-10^{-10}$) to eliminate values that tend to be zero.

However, the SVM is a binary classifier which only recognizes true or false. For multiclass problem, we use the so-called one-versus-all (OVA) method with linear kernel to combine the results of each individual classifier. The final class in testing phase is mainly determined by selecting the maximum similarity.

For all languages, our parser uses the same settings and features. For all the languages (except for Basque 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 (Nivre et al., 2007) which consists of 10 language treebanks. More detailed descriptions of the dataset can be found at the web site[1]. The experimental results are mainly evaluated by the unlabeled and labeled attachment scores. CoNLL also provided a perl script to automatic compute these rates.

### 3.2 Results

Table 1 presents the overall parsing performance of the 10 languages. As shown in Table 1, we list two parsing results at column B and column C (new and old). It is worth to note that the result B is produced by training the neighbor parser with full labels instead of the three categories, left/right/out-of-neighbor. A is the official provided parse results. Some of the parsing results in A did not include the enhanced feature type and neighbor/root parses due to the time limitation. For the domain adaptation task, we directly use the trained English model to classify the PChemtb and CHILDES corpora without further adjustment.

In addition, we also apply the Maltparser 0.4, which is implemented with the Nivre's algorithm (Nivre et al., 2006) to be compared. The Maltpaser also includes the SVM and memory-based learner (MBL). Nevertheless, the training time complexity of the SVM in Maltparser is not linear time as MFN-SVM. Therefore we use the default MBL and feature model 3 (M3) in this experiment. To make a fair comparison, the input training data was also projectivized through the same pseudo-projective encoding/decoding methods.

[1] http://nextens.uvt.nl/depparse-wiki/SharedTaskWebsite

**Table 1: A general statistical table of labeled attachment score, test and un-parsed rate (percentage)**

| Language | A (Official) | B (Corrected) | C (Malt-Parser 0.4) | Statistic test | | | Un-Parsed Rate | |
|---|---|---|---|---|---|---|---|---|
| | | | | A vs B | A vs C | B vs C | Old | New |
| Arabic | 66.16 | 70.71 | 56.67 | Yes | No | Yes | 1.08% | 0.69% |
| Basque | 70.71 | 72.26 | 57.79 | Yes | Yes | Yes | 3.04% | 3.72% |
| Catalan | 81.44 | 81.44 | 76.36 | Yes | No | No | 0.45% | 0.27% |
| Chinese | 74.69 | 79.29 | 68.15 | Yes | Yes | Yes | 0.00% | 0.00% |
| Czech | 66.72 | 70.24 | 56.96 | Yes | No | Yes | 4.17% | 3.87% |
| English | 79.49 | 84.27 | 75.53 | Yes | Yes | Yes | 1.66% | 0.84% |
| Greek | 70.63 | 77.64 | 58.81 | No | Yes | Yes | 2.26% | 2.12% |
| Hungarian | 69.08 | 71.98 | 59.41 | Yes | Yes | Yes | 3.88% | 5.38% |
| Italian | 78.79 | 78.38 | 74.08 | Yes | No | Yes | 0.63% | 0.63% |
| Turkish | 72.52 | 75.65 | 64.41 | Yes | Yes | Yes | 4.93% | 5.54% |
| pchemtb_closed | 55.31** | 73.35 | - | - | - | - | - | - |
| *CHILDES_closed | 52.89 | 58.29 | - | - | - | - | - | - |

* The CHILDES data does not contain the relation tag, instead, the unlabeled attachment score is listed

** The original submission of the pchemtb_closed task can not pass through the evaluator and hence is not the official score. After correcting the format problems, the actual LAS score should be 55.31.

To perform the significant test, we 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 ($p < 0.05$).

The final column of the Table 1 lists the non-root words unparsed rate of the modified Yamada's method and the Nivre's parsing model which we employed last year. Among 10 languages, we can find that the modified Yamada's method outperform our old method in five languages, while fail to win in three languages. We did not report the comparative study between the forward parsing and backward parsing directions here since only the two languages (Basque and Turkish) were better in performing forward direction.

## 4 Discussion

Now we turn to discuss the improvement of the use of the neighbor parse and root parse. All of the experiments were conducted by additional runs where we removed the neighbor and root parse outputs from the feature set. In this experiment, we report four representative languages that tend to achieve the best and worst improvements. Table 2 lists the comparative study of the four languages.

As listed in Table 2, both English and Chinese got substantial benefit from the use of the two parsers. As observed by (Isozaki et al., 2004), incorporating both top-down (root find) and bottom-up (base-NP) can yield better improvement over the Yamada's parsing algorithm. Thus, instead of pre-determining the root and base-phrase structures, the tagging results of the neighbor and root parsers were included as new features to add wider information for the shift-reduce parser. It is also interesting to link neighbors and determine the root before parsing. We plan to compare it with out method in the future.

**Table 2: The effective of the used Neighbor/Root Parser in the selected four languages**

| | With N/R Parser | Without |
|---|---|---|
| Chinese | 79.29 | 75.51 |
| English | 84.27 | 79.49 |
| Basque | 72.26 | 72.32 |
| Turkish | 75.65 | 76.60 |

On the other hand, we also found that 2 out of the 10 languages had been negatively affected by the neighbor and root parsers. In Basque they made a marginally negative improvement, and in the Turkish the two parsers did decrease the original parsing models. We further observed that the main cause is that the weak performance of the neighbor parser. In Turkish, the recall/precision rates of the neighbor dependence are 92.61/93.12 with include neighbor parse outputs, while it achieved 93.71/93.51 with purely run the modified Yamada's method. We can expect that the result could achieve higher LAS score when the neighbor parser is improved. As mentioned in section 2.1, 2.2, the selected features for the two parsers are unified for the 10 languages. It is not surprising that for certain data the fixed feature set might perform even worse than the original shift-reduce parser. A better way is to validate the features with variant settings for different languages. We left the feature engine task as future work.

## 5 Conclusion and Future Remarks

Multilingual dependency parsing investigates on proposing a general framework of dependence parsing algorithms. This paper presents and analyzes the impact of two preprocessing components, namely, neighbor parsing and root-parsing. Those two parsers provide very useful additional features for downstream shift-reduce parser. The experimental results also demonstrated that the use of the two components did improve results for the selected languages. In the error-analysis, we also observed that for some languages, parameter tuning and feature selection is very important for system performance.

In the future, we plan to report the actual performance with replacing the MFN-SVM by the polynomial kernel SVM. In our pilot study, the use of approximate-polynomial kernel (Wu et al., 2007) outperforms the linear kernel SVM in Chinese and Arabic. Also, we are investigating how to convert the shift-reduce parser into approximate *N*-best parser efficiently. In this way, the parse reranking algorithm can be adopted to further improve the performance.

## References


A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*. Kluwer.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Diaz de Ilarraza, A. Garmendia and M. Oronoz. 2003. Construction of a Basque Dependency Treebank. In *Proc. of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pages 201–204.

A. Böhmová, J. Hajic, E. Hajicová and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (2003), chapter 7, 103–127.

R. Brown. 1973. *A First Language: The Early Stages*. Harvard University Press.

M. W. Chang, Q. Do, and D. Roth. 2006. Multilingual Dependency Parsing: A Pipeline Approach. In *Recent Advances in Natural Language Processing*, pages 195-204.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang and Z. Gao. 2003. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In Abeillé (2003), chapter 13, pages 231–248.

Y. Cheng, M. Asahara and Y. Matsumoto. 2006. Multilingual Dependency Parsing at NAIST. *In Proc. of the 10th Conference on Natural Language Learning*, pages 191-195.

D. Czendes, J. Csirik, T. Gyimóthy, and A. Kocsor. 2005. *The Szeged Treebank*. Springer.

J. Hajic, O. Smrz, P. Zemánek, J. Snaidauf and E. Beska. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

H. Isozaki; H. Kazawa; T. Hirao. 2004. A Deterministic Word Dependency Analyzer Enhanced With Preference Learning. *In Proc. of the 20th International Conference on Computational Linguistics*, pages 275-281.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

S. Keerthi and D. DeCoste. 2005. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*. 6: 341-361.

S. Kulick, A. Bies, M. Liberman, M. Mandel, R. McDonald, M. Palmer, A. Schein, and L. Ungar. 2004. Integrated annotation for biomedical information extraction. In *Proc. of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*.

B. MacWhinney. 2000. The CHILDES Project: Tools for Analyzing Talk. Lawrence Erlbaum.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. A. Martí, M. Taulé, L. Màrquez and M. Bertran. 2007. CESS-ECE: A Multilingual and Multilevel Annotated Corpus. Available for download from: http://www.lsi.upc.edu/~mbertran/cess-ece/.

R. McDonald, K. Lerman and F. Pereira. 2006. Multilingual Dependency Analysis with a Two-Stage Discriminative. *In Proc. of the 10th Conference on Natural Language Learning*, pages 216-220.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, O. Corazzari, A. Lenci, A. Zampolli, F. Fanciulli, M. Massetani, R. Raffaelli, R. Basili, M. T. Pazienza, D. Saracino, F. Zanzotto, N. Nana, F. Pianesi, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Abeillé (2003), chapter 11, pages 189–210.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. *In Proc. of the International Workshop on Parsing Technology*, pages 149-160.

J. Nivre, and J. Nilsson. 2005. Pseudo-projective dependency Parsing. *In Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99-106.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. *In Proc. of the 10th Conference on Natural Language Learning*, pages 221-225.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (2003), chapter 15, pages 261–277.

P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis. 2005. Theoretical and practical issues in the construction of a Greek depen- dency treebank. *In Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 149–160.

T. Kudo, K, Yamamoto, and Y. Matsumoto. 2004. Appliying conditional random fields to Japanese morphological analysis, *In Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 230-237.

Y. C. Wu, Y. S. Lee, and J. C. Yang. 2006a. The exploration of deterministic and efficient dependency parsing. *In Proc. of the 10th Conference on Computational Natural Language Learning*, pages 241-245.

Y. C. Wu, J. C. Yang, and Q. X. Lin. 2006b. Description of the NCU Chinese word segmentation and named entity recognition system for SIGHAN bakeoff 2006. *In Proc. of the 5th SIGHAN Workshop on Chinese Language Processing*, pages 209-212.

Y. C. Wu, J. C. Yang, and Y. S. Lee. 2007. An Approximate Approach for Training Polynomial Kernel SVMs in Linear Time. *In Proc. of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, in press.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. *In Proc. of the 8th International Workshop on Parsing Technologies*, pages 195–206.