Probabilistic Parsing Action Models for Multi-lingual Dependency Parsing

Xiangyu Duan
Institute of Automation, Chinese Academy of Sciences
xyduan@nlpr.ia.ac.cn

Jun Zhao
Institute of Automation, Chinese Academy of Sciences
jzhao@nlpr.ia.ac.cn

Bo Xu
Institute of Automation, Chinese Academy of Sciences
xubo@hitic.ia.ac.cn

Abstract

Deterministic dependency parsers use parsing actions to construct dependencies. These parsers do not compute the probability of the whole dependency tree. They only determine parsing actions stepwisely by a trained classifier. To globally model parsing actions of all steps that are taken on the input sentence, we propose two kinds of probabilistic parsing action models that can compute the probability of the whole dependency tree. The tree with the maximal probability is outputted. The experiments are carried on 10 languages, and the results show that our probabilistic parsing action models outperform the original deterministic dependency parser.

1 Introduction

The target of CoNLL 2007 shared task (Nivre et al., 2007) is to parse texts in multiple languages by using a single dependency parser that has the capacity to learn from treebank data. Among parsers participating in CoNLL 2006 shared task (Buchholz et al., 2006), deterministic dependency parser shows great efficiency in time and comparable performances for multi-lingual dependency parsing (Nivre et al., 2006). Deterministic parser regards parsing as a sequence of parsing actions that are taken step by step on the input sentence. Parsing actions construct dependency relations between words.

Deterministic dependency parser does not score the entire dependency tree as most of state-of-the-art parsers. They only stepwisely choose the most probable parsing action. In this paper, to globally model parsing actions of all steps that are taken on the input sentence, we propose two kinds of probabilistic parsing action models that can compute the entire dependency tree's probability. Experiments are evaluated on diverse data set of 10 languages provided by CoNLL 2007 shared-task (Nivre et al., 2007). Results show that our probabilistic parsing action models outperform the original deterministic dependency parser. We also present a general error analysis across a wide set of languages plus a detailed error analysis of Chinese.

Next we briefly introduce the original deterministic dependency parsing algorithm that is a basic component of our models.

2 Introduction of Deterministic Dependency Parsing

There are mainly two representative deterministic dependency parsing algorithms proposed respectively by Nivre (2003), Yamada and Matsumoto (2003). Here we briefly introduce Yamada and Matsumoto's algorithm, which is adopted by our models, to illustrate deterministic dependency parsing. The other representative method of Nivre also parses sentences in a similar deterministic manner except different data structure and parsing actions.

Yamada's method originally focuses on unlabeled dependency parsing. Three kinds of parsing actions are applied to construct the dependency between two focus words. The two focus words are the current sub tree's root and the succeeding (right) sub tree's root given the current ***parsing state***. Every parsing step results in a ***new parsing state***, which includes all elements of the current partially built tree. Features are extracted about these two focus words. In the training phase, features and the corresponding parsing action compose the training

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 940–946,
Prague, June 2007.

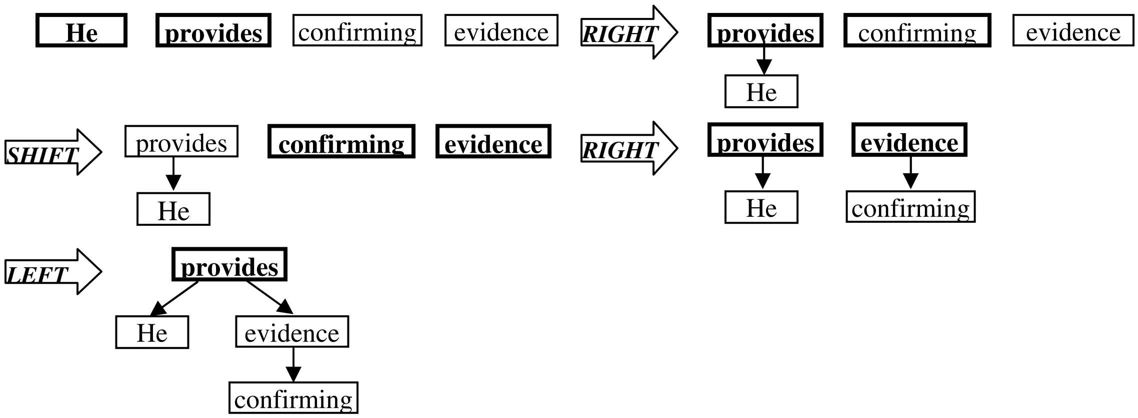

Figure 1. The example of the parsing process of Yamada and Matsumoto's method. The input sentence is "He provides confirming evidence."

data. In the testing phase, the classifier determines which parsing action should be taken based on the features. The parsing algorithm ends when there is no further dependency relation can be made on the whole sentence. The details of the three parsing actions are as follows:

LEFT: it constructs the dependency that the right focus word depends on the left focus word.

RIGHT: it constructs the dependency that the left focus word depends on the right focus word.

SHIFT: it does not construct dependency, just moves the parsing focus. That is, the new left focus word is the previous right focus word, whose succeeding sub tree's root is the new right focus word.

The illustration of these three actions and the parsing process is presented in figure 1. Note that the focus words are shown as bold black box.

We extend the set of parsing actions to do labeled dependency parsing. ***LEFT*** and ***RIGHT*** are concatenated by dependency labels, while ***SHIFT*** remains the same. For example in figure 1, the original action sequence "***RIGHT*** -> ***SHIFT*** -> ***RIGHT*** -> ***LEFT***" becomes "***RIGHT-SBJ*** -> ***SHIFT*** -> ***RIGHT-NMOD*** -> ***LEFT-OBJ***".

3 Probabilistic Parsing Action Models

Deterministic dependency parsing algorithms are greedy. They choose the most probable parsing action at every parsing step given the current parsing state, and do not score the entire dependency tree. To compute the probability of whole dependency tree, we propose two kinds of probabilistic models that are defined on parsing actins: parsing action chain model (PACM) and parsing action phrase model (PAPM).

3.1 Parsing Action Chain Model (PACM)

The parsing process can be viewed as a Markov Chain. At every parsing step, there are several candidate parsing actions. The objective of this model is to find the most probable sequence of parsing actions by taking the Markov assumption. As shown in figure 1, the action sequence "***RIGHT-SBJ*** -> ***SHIFT*** -> ***RIGHT-NMOD*** -> ***LEFT-OBJ***" constructs the right dependency tree of the example sentence. Choosing this action sequence among all candidate sequences is the objective of this model.

Firstly, we should define the probability of the dependency tree conditioned on the input sentence.

$$P(T \mid S) = \prod_{i=1...n} P(d_i \mid d_0 ... d_{i-1}, S) \qquad (1)$$

Where T denotes the dependency tree, S denotes the original input sentence, d_i denotes the parsing action at time step i. We add an artificial parsing action d_0 as initial action.

We introduce a variable $context_{d_i}$ to denote the resulting ***parsing state*** when the action d_i is taken on $context_{d_{i-1}}$. $context_{d_0}$ is the original input sentence.

Suppose $d_0 ... d_n$ are taken sequentially on the input sentence S, and result in a sequence of ***parsing states*** $context_{d_0} ... context_{d_n}$, then $P(T|S)$ defined in equation (1) becomes as below:

$$\prod_{i=1...n} P(context_{d_i} \mid context_{d_0},...,context_{d_{i-1}}) \quad (2)$$

$$\approx \prod_{i=1...n} P(context_{d_i} \mid context_{d_{i-1}}) \quad (3)$$

$$= \prod_{i=1...n} P(d_i \mid context_{d_{i-1}}) \quad (4)$$

Formula (3) comes from formula (2) by obeying the Markov assumption. Note that formula (4) is about the classifier of parsing actions. It denotes the probability of the parsing action d_i given the ***parsing state*** $context_{d_{i-1}}$. If we train a classifier that can predict with probability output, then we can compute $P(T|S)$ by computing the product of the probabilities of parsing actions. The classifier we use throughout this paper is SVM (Vapnik, 1995). We adopt Libsvm (Chang and Lin, 2005), which can train multi-class classifier and support training and predicting with probability output (Chang and Lin, 2005).

For this model, the objective is to choose the parsing action sequence that constructs the dependency tree with the maximal probability.

$$\max P(T \mid S) = \max_{d_1...d_n} \prod_{i=1...n} P(d_i \mid context_{d_{i-1}}) \quad (5)$$

Because this model chooses the most probable sequence, not the most probable parsing action at only one step, it avoids the greedy property of the original deterministic parsers.

We use beam search for the decoding of this model. We use m to denote the beam size. Then beam search is carried out as follows. At every parsing step, all ***parsing states*** are ordered (or partially m ordered) according to their probabilities. Probability of a ***parsing state*** is determined by multiplying the probabilities of actions that generate that state. Then we choose m best ***parsing states*** for this step, and next parsing step only consider these m best ***parsing states***. Parsing terminates when the first entire dependency tree is constructed. To obtain a list of n-best parses, we simply continue parsing until either n trees are found, or no further parsing can be fulfilled.

3.2 Parsing Action Phrase Model (PAPM)

In the Parsing Action Chain Model (PACM), actions are competing at every parsing step. Only m best parsing states resulted by the corresponding actions are kept for every step. But for the parsing problem, it is reasonable that actions are competing for which phrase should be built. For dependency syntax, one phrase consists of the head word and all its children. Based on this motivation, we propose Parsing Action Phrase Model (PAPM), which divides parsing actions into two classes: constructing action and shifting action.

If a phrase is built after an action is performed, the action is called constructing action. In original Yamada's algorithm, constructing actions are ***LEFT*** and ***RIGHT***. For example, if ***LEFT*** is taken, it indicates that the right focus word has found all its children and becomes the head of this new phrase. Note that one word with no children can also be viewed as a phrase if its dependency on other word is constructed. In the extended set of parsing actions for labeled parsing, compound actions, which consist of ***LEFT*** and ***RIGHT*** concatenated by dependency labels, are constructing actions.

If no phrase is built after an action is performed, the action is called shifting action. Such action is ***SHIFT***.

We denote a_j as constructing action and b_j as shifting action. j indexes the time step. Then we introduce a new concept: parsing action phrase. We use A_i to denote the ith parsing action phrase. It can be expanded as $A_i \rightarrow b_{j-k}...b_{j-1}a_j$. That is, parsing action phrase A_i is a sequence of parsing actions that constructs the next syntactic phrase.

For example, consider the parsing process in figure 1, A_1 is "***RIGHT-SBJ***", A_2 is "***SHIFT***, ***RIGHT-NMOD***", A_3 is "***LEFT-OBJ***". Note that A_1 consists of a constructing action, A_2 consists of a shifting action and a constructing action, A_3 consists of a constructing action.

The indexes are different for both sides of the expansion $A_i \rightarrow b_{j-k}...b_{j-1}a_j$, A_i is the ith parsing action phrase corresponding to both constructing action a_j at time step j and all its preceding shifting actions. Note that on the right side of the expansion, only one constructing action is allowed and is always at the last position, while shifting action can occur several times or does not occur at all. It is parsing action phrases, i.e. sequences of parsing actions, that are competing for which next phrase should be built.

The probability of the dependency tree given the input sentence is redefined as:

$$P(T \mid S) = \prod_{i=1...n} P(A_i \mid A_1...A_{i-1}, S) \qquad (6)$$

$$= \prod_{i=1...n} P(context_{A_i} \mid context_{A_1}...context_{A_{i-1}})$$

$$\approx \prod_{i=1...n} P(context_{A_i} \mid context_{A_{i-1}})$$

$$= \prod_{i=1...n} P(A_i \mid context_{A_{i-1}})$$

$$= \prod_{i=1...n} P(b_{j-k}...b_{j-1}a_j \mid context_{A_{i-1}})$$

$$= \prod_{i=1...n} P(b_{j-k} \mid context_{A_{i-1}}) \cdot$$

$$(\prod_{t=2...k} P(b_{j-t+1} \mid context_{b_{j-t}})) \cdot P(a_j \mid context_{b_{j-1}})$$

Where k represents the number of steps that shifting action can be taken. $context_{A_i}$ is the parsing state resulting from a sequence of actions $b_{j-k}...b_{j-1}a_j$ taken on $context_{A_{i-1}}$.

The objective in this model is to find the most probable sequence of parsing action phrases.

$$\max P(T \mid S) = \max_{A_1...A_n} \prod_{i=1...n} P(A_i \mid context_{A_{i-1}}) \qquad (7)$$

Similar with parsing action chain model (PACM), we use beam search for the decoding of parsing action phrase model (PAPM). The difference is that PAPM do not keep m best ***parsing states*** at every parsing step. Instead, PAPM keep m best states which are corresponding to m best current parsing action phrases (several steps of ***SHIFT*** and the last step of a constructing action).

4 Experiments and Results

Experiments are carried on 10 languages provided by CoNLL 2007 shared-task organizers (Nivre et al., 2007). Among these languages, Chinese (Chen et al., 2003), Catalan (Martí et al., 2007) and English (Johansson and Nugues, 2007) have low percentage of non-projective relations, which are 0.0%, 0.1% and 0.3% respectively. Except these three languages, we use software of projectivization/deprojectivization provided by Nivre and Nilsson (2005) for other languages. Because our algorithm only deals with projective parsing, we should projectivize training data at first to prepare for the following training of our algorithm. During testing, deprojectivization is applied to the output of the parser.

Considering the classifier of Libsvm (Chang and Lin, 2005), the features are extracted from the following fields of the data representation: FORM, LEMMA, CPOSTAG, POSTAG, FEATS and DEPREL. We split values of FEATS field into its atomic components. We only use available features of DEPREL field during deterministic parsing. We use similar feature context window as used in Yamada's algorithm (Yamada and Matsumoto, 2003). In detail, the size of feature context window is six, which consists of left two sub trees, two focus words related sub trees and right two sub trees. This feature template is used for all 10 languages.

4.1 Results of PACM and Yamada's Method

After submitting the testing results of Parsing Action Chain Model (PACM), we also perform original deterministic parsing proposed by Yamada and Matsumoto (2003). The total results are shown in table 1. The experimental results are mainly evaluated by labeled attachment score (*LAS*), unlabeled attachment score (*UAS*) and labeled accuracy (*LA*).

Table 1 shows that Parsing Action Chain Model (PACM) outperform original Yamada's parsing method for all languages. The *LAS* improvements range from 0.60 percentage points to 1.71 percentage points. Note that the original Yamada's method still gives testing results above the official reported average performance of all languages.

	Ara	Bas	Cat	Chi	Cze	Eng	Gre	Hun	Ita	Tur
LAS_{Yam}	69.31	69.67	83.26	81.88	74.63	84.81	72.75	76.24	80.08	73.94
UAS_{Yam}	78.93	75.86	88.53	86.17	80.11	85.83	79.45	79.97	83.69	79.79
LA_{Yam}	81.13	75.71	88.36	84.56	82.10	89.71	82.58	88.37	86.93	80.81
LAS_{PACM}	**69.91**	**71.26**	**84.95**	**82.58**	**75.34**	**85.83**	**74.29**	**77.06**	**80.75**	**75.03**
UAS_{PACM}	**79.04**	**77.57**	**89.71**	**86.88**	**80.82**	**86.97**	**80.77**	**80.66**	**84.20**	**81.03**
LA_{PACM}	**81.40**	**77.35**	**89.55**	**85.35**	**83.17**	**90.57**	**83.87**	**88.92**	**87.32**	**81.17**

Table 1. The performances of Yamada's method (*Yam*) and Parsing Action Chain Model (*PACM*).

4.2 Results of PAPM

Not all languages have only one root node of a sentence. Since Parsing Action Phrase Model (PAPM) only builds dependencies, and shifting action is not the ending action of a parsing action phrase, PAPM always ends with one root word. This property makes PAPM only suitable for Catalan, Chinese, English and Hungarian, which are unary root languages. PAPM result of Catalan was not submitted before deadline due to the shortage of time and computing resources. We report Catalan's PAPM result together with that of other three languages in table 2.

	Cat	Chi	Eng	Hun
LAS_{PAPM}	87.26	82.64	86.69	76.89
UAS_{PAPM}	92.07	86.94	87.87	80.53
LA_{PAPM}	91.89	85.41	92.04	89.73

Table 2. The performance of Parsing Action Phrase Model (*PAPM*) for Catalan, Chinese, English and Hungarian.

Compared with the results of PACM shown in table 1, the performance of PAPM differs among different languages. Catalan and English show that PAPM improves 2.31% and 0.86% respectively over PACM, while the improvement of Chinese is marginal, and there is a little decrease of Hungarian. Hungarian has relatively high percentage of non-projective relations. If phrase consists of head word and its non-projective children, the constructing actions that are main actions in PAPM will be very difficult to be learned because some non-projective children together with their heads have no chance to be simultaneously as focus words. Although projectivization is also performed for Hungarian, the built-in non-projective property still has negative influence on the performance.

5 Error Analysis

In the following we provide a general error analysis across a wide set of languages plus a detailed analysis of Chinese.

5.1 General Error Analysis

One of the main difficulties in dependency parsing is the determination of long distance dependencies. Although all kinds of evaluation scores differ dramatically among different languages, 69.91% to 85.83% regarding *LAS*, there are some general observations reflecting the difficulty of long distance dependency parsing. We study this difficulty from two aspects about our full submission of PACM: precision of dependencies of different arc lengths and precision of root nodes.

For arcs of length 1, all languages give high performances with lowest 91.62% of Czech (Böhmova et al., 2003) to highest 96.8% of Catalan (Martí et al., 2007). As arcs lengths grow longer, various degradations are caused. For Catalan, score of arc length 2 is similar with that of arc length 1, but there are dramatic degradations for longer arc lengths, from 94.94% of arc length 2 to 85.22% of length 3-6. For English (Johansson and Nugues, 2007) and Italian (Montemagni et al., 2003), there are graceful degradation for arcs of length 1,2 and 3-6, with 96-91-85 of English and 95-85-75 of Italian. For other languages, long arcs also give remarkable degradations that pull down the performance.

Precision of root nodes also reflects the performance of long arc dependencies because the arc between the root and its children are often long arcs. In fact, it is the precision of roots and arcs longer than 7 that mainly pull down the overall performance. Yamada's method is a bottom-up parsing algorithm that builds short distance dependencies at first. The difficulty of building long arc dependencies may partially be resulted from the errors of short distance dependencies. The deterministic manner causes error propagation, and it indirectly indicates that the errors of roots are the final results of error propagation of short distance dependencies. But there is an exception occurred in Chinese. The root precision is 90.48%, only below the precision of arcs of length 1. This phenomenon exists because the sentences in Chinese data set (Chen et al., 2003) are in fact clauses with average length of 5.9 rather than entire sentences. The root words are heads of clauses.

Both Parsing Action Chain Model (PACM) and Parsing Action Phrase Model (PAPM) avoid greedy property of original Yamada's method. It can be expected that there will be a precision improvement of long distance dependencies over original Yamada's method. For PACM, the results of Basque (Aduriz et al., 2003), Catalan (Martí et al., 2007), Chinese (Chen et al., 2003), English (Johansson and Nugues, 2007) and Greek (Prokopidis et al., 2005) show that the root precision improvement over Yamada's method is more conspicuous than that of other long distance dependencies. The largest improvement of roots precision is 10.7% of Greek. While for Arabic (Hajic et al., 2004), Czech (Böhmova et al., 2003), Hungarian (Csendes et al., 2005), Italian (Montemagni et al., 2003) and Turkish (Oflazer et al., 2003), the improvement of root precision is small, but dependencies of arcs longer than 1 give better scores. For PAPM, good performances of Catalan and English also give significant improvements of root precision over PACM. For Catalan, the root precision improvement is from 63.86% to 95.21%; for English, the root precision improvement is from 62.03% to 89.25%.

5.2 Error Analysis of Chinese

There are mainly two sources of errors regarding *LAS* in Chinese dependency parsing.

One is from conjunction words (C) that have a relatively high percentage of wrong heads (about 20%), and therefore 19% wrong dependency labels. In Chinese, conjunction words often concatenate clauses. Long distance dependencies between clauses are bridged by conjunction words. It is difficult for conjunction words to find their heads.

The other source of errors comes from auxiliary words (DE) and preposition words (P). Unlike conjunction words, auxiliary words and preposition words have high performance of finding right head, but label accuracy (LA) decrease significantly. The reason may lie in the large dependency label set consisting of 57 kinds of dependency labels in Chinese. Moreover, auxiliary words (DE) and preposition words (P) have more possible dependency labels than other coarse POS have. This introduces ambiguity for parsers.

Most common POS including noun and verb contribute much to the overall performance of 83% Labeled Attachment Scores (*LAS*). Adverbs obtain top score while adjectives give the worst.

6 Conclusion

We propose two kinds of probabilistic models defined on parsing actions to compute the probability of entire sentence. Compared with original Yamada and Matsumoto's deterministic dependency method which stepwisely chooses most probable parsing action, the two probabilistic models improve the performance regarding all 10 languages in CoNLL 2007 shared task. Through the study of parsing results, we find that long distance dependencies are hard to be determined for all 10 languages. Further analysis about this difficulty is needed to guide the research direction. Feature exploration is also necessary to provide more informative features for hard problems.

Ackowledgements

This work was supported by Hi-tech Research and Development Program of China under grant No. 2006AA01Z144, the Natural Sciences Foundation of China under grant No. 60673042, and the Natural Science Foundation of Beijing under grant No. 4052027, 4073043.

References

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. SIGNLL.

Chih-Chung Chang and Chih-Jen Lin. 2005. LIBSVM: A library for support vector machines.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL-2005*, pages 99–106.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, S. Marinov. 2006. Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. In *Proc. of the Tenth Conference on Computational Natural Language Learning (CoNLL)*.

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)*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*.

V. Vapnik. 1995. The Nature of StatisticalLearning Theory. Springer.

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.

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.

D. Csendes, 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.

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

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/.

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, 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 CoNLL 2007 Shared Task. 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.