Multilingual Dependency Parsing and Domain Adaptation using DeSR

Giuseppe Attardi
Felice Dell'Orletta
Maria Simi
Dipartimento di Informatica
largo B. Pontecorvo 3
I-56127 Pisa, Italy
attardi@di.unipi.it
felice.dellorletta@ilc.cnr.it
simi@di.unipi.it

Atanas Chanev
Università di Trento
via Matteo del Ben 5
I-38068 Rovereto, Italy
Fondazione Bruno Kessler-irst
via Sommarive 18
I-38050 Povo, Italy
chanev@form.unitn.it

Massimiliano Ciaramita
Yahoo! Research Barcelona
Ocata 1
S-08003 Barcelona, Spain
massi@yahoo-inc.com

Abstract

We describe our experiments using the DeSR parser in the multilingual and domain adaptation tracks of the CoNLL 2007 shared task. DeSR implements an incremental deterministic Shift/Reduce parsing algorithm, using specific rules to handle non-projective dependencies. For the multilingual track we adopted a second order averaged perceptron and performed feature selection to tune a feature model for each language. For the domain adaptation track we applied a tree revision method which learns how to correct the mistakes made by the base parser on the adaptation domain.

1 Introduction

Classifier-based dependency parsers (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004) learn from an annotated corpus how to select an appropriate sequence of Shift/Reduce actions to construct the dependency tree for a sentence. Learning is based on techniques such as SVM (Vapnik 1998) or Memory Based Learning (Daelemans 2003), which provide high accuracy but are often computationally expensive. For the multilingual track in the CoNLL 2007 Shared Task, we employed a Shift/Reduce parser which uses a perceptron algorithm with second-order feature maps, in order to verify whether a simpler and faster algorithm can still achieve comparable accuracy.

For the domain adaptation track we wished to explore the use of tree revisions in order to incorporate language knowledge from a new domain.

2 Multilingual Track

The overall parsing algorithm is a deterministic classifier-based statistical parser, which extends the approach by Yamada and Matsumoto (2003), by using different reduction rules that ensure deterministic incremental processing of the input sentence and by adding specific rules for handling non-projective dependencies. The parser also performs dependency labeling within a single processing step.

The parser is modular and can use several learning algorithms. The submitted runs used a second order Average Perceptron, derived from the multiclass perceptron of Crammer and Singer (2003).

No additional resources were used. No pre-processing or post-processing was used, except stemming for English, by means of the Snowball stemmer (Porter 2001).

3 Deterministic Classifier-based Parsing

DeSR (Attardi, 2006) is an incremental deterministic classifier-based parser. The parser constructs dependency trees employing a deterministic bottom-up algorithm which performs Shift/Reduce actions while analyzing input sentences in left-to-right order.

Using a notation similar to (Nivre and Scholz, 2003), the state of the parser is represented by a

Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 1112–1118,
Prague, June 2007.

quadruple $\langle S, I, T, A\rangle$, where S is the stack of past tokens, I is the list of (remaining) input tokens, T is a stack of temporary tokens and A is the arc relation for the dependency graph.

Given an input string W, the parser is initialized to $\langle (), W, (), ()\rangle$, and terminates when it reaches a configuration $\langle S, (), (), A\rangle$.

The three basic parsing rule schemas are as follows:

$$\textit{Shift} \quad \frac{\langle S, n|I, T, A\rangle}{\langle n|S, I, T, A\rangle}$$

$$\textit{Right}_d \quad \frac{\langle s|S, n|I, T, A\rangle}{\langle S, n|I, T, A\cup\{(s, d, n)\}\rangle}$$

$$\textit{Left}_d \quad \frac{\langle s|S, n|I, T, A\rangle}{\langle S, s|I, T, A\cup\{(n, d, s)\}\rangle}$$

The schemas for the *Left* and *Right* rules are instantiated for each dependency type $d \in D$, for a total of $2|D| + 1$ rules. These rules perform both attachment and labeling.

At each step the parser uses classifiers trained on a treebank corpus in order to predict which action to perform and which dependency label to assign given the current configuration.

4 Non-Projective Relations

For handling non-projective relations, Nivre and Nilsson (2005) suggested applying a preprocessing step to a dependency parser, which consists in lifting non-projective arcs to their head repeatedly, until the tree becomes pseudo-projective. A post-processing step is then required to restore the arcs to the proper heads.

In DeSR non-projective dependencies are handled in a single step by means of the following additional parsing rules, slightly different from those in (Attardi, 2006):

$$\textit{Right2}_d \quad \frac{\langle s_1|s_2|S, n|I, T, A\rangle}{\langle S, s_1|n|I, T, A\cup\{(s_2, d, n)\}\rangle}$$

$$\textit{Left2}_d \quad \frac{\langle s_1|s_2|S, n|I, T, A\rangle}{\langle s_2|S, s_1|I, T, A\cup\{(n, d, s_2)\}\rangle}$$

$$\textit{Right3}_d \quad \frac{\langle s_1|s_2|s_3|S, n|I, T, A\rangle}{\langle S, s_1|s_2|n|I, T, A\cup\{(s_3, d, n)\}\rangle}$$

$$\textit{Left3}_d \quad \frac{\langle s_1|s_2|s_3|S, n|I, T, A\rangle}{\langle s_2|s_3|S, s_1|I, T, A\cup\{(n, d, s_3)\}\rangle}$$

$$\textit{Extract} \quad \frac{\langle s_1|s_2|S, n|I, T, A\rangle}{\langle n|s_1|S, I, s_2|T, A\rangle}$$

$$\textit{Insert} \quad \frac{\langle S, I, s_1|T, A\rangle}{\langle s_1|S, I, T, A\rangle}$$

Left2, *Right2* are similar to *Left* and *Right*, except that they create links crossing one intermediate node, while *Left3* and *Right3* cross two intermediate nodes. Notice that the *RightX* actions put back on the input the intervening tokens, allowing the parser to complete the linking of tokens whose processing had been delayed. *Extract*/*Insert* generalize the previous rules by moving one token to the stack T and reinserting the top of T into S.

5 Perceptron Learning and 2nd-Order Feature Maps

The software architecture of the DeSR parser is modular. Several learning algorithms are available, including SVM, Maximum Entropy, Memory-Based Learning, Logistic Regression and a few variants of the perceptron algorithm.

We obtained the best accuracy with a multiclass averaged perceptron classifier based on the ultraconservative formulation of Crammer and Singer (2003) with uniform negative updates. The classifier function is:

$$F(x) = \arg\max_k \{\alpha_k \cdot x\}$$

where each parsing action k is associated with a weight vector α_k. To regularize the model the final weight vectors are computed as the average of all weight vectors posited during training. The number of learning iterations over the training data, which is the only adjustable parameter of the algorithm, was determined by cross-validation.

In order to overcome the limitations of a linear perceptron, we introduce a feature map Φ: $\mathbb{R}^d \rightarrow \mathbb{R}^{d(d+1)/2}$ that maps a feature vector $\boldsymbol{x}$ into a higher dimensional feature space consisting of all unordered feature pairs:

$$\Phi(\boldsymbol{x}) = \langle x_i x_j \mid i = 1, \ldots, d, j = i, \ldots, d\rangle$$

In other words we expand the original representation in the input space with a feature map that generates all second-order feature combinations from each observation. We call this the 2^{nd}-order model, where the inner products are computed as $\alpha_k \cdot \Phi(\boldsymbol{x})$, with α_k a vector of dimension $d(d+1)/2$. Applying a linear perceptron to this feature space corresponds to simulating a polynomial kernel of degree two.

A polynomial kernel of degree two for SVM was also used by Yamada and Matsumoto (2003). However, training SVMs on large data sets like those arising from a big training corpus was too computationally expensive, forcing them to resort to partitioning the training data (by POS) and to learn several models.

Our implementation of the perceptron algorithm uses sparse data structures (hash maps) so that it can handle efficiently even large feature spaces in a single model. For example the feature space for the 2nd-order model for English contains over 21 million. Parsing unseen data can be performed at tens of sentences per second. More details on such aspects of the DeSR parser can be found in (Ciaramita and Attardi 2007).

6 Tuning

The base parser was tuned on several parameters to optimize its accuracy as follows.

6.1 Feature Selection

Given the different characteristics of languages and corpus annotations, it is worth while to select a different set of features for each language. For example, certain corpora do not contain lemmas or morphological information so lexical information will be useful. Vice versa, when lemmas are present, lexical information might be avoided, reducing the size of the feature set.

We performed a series of feature selection experiments on each language, starting from a fairly comprehensive set of 43 features and trying all variants obtained by dropping a single feature. The best of these alternatives feature models was chosen and the process iterated until no further gains were achieved. The score for the alternatives was computed on a development set of approximately 5000 tokens, extracted from a split of the original training corpus.

Despite the process is not guaranteed to produce a global optimum, we noticed LAS improvements of up to 4 percentage points on some languages.

The set of features to be used by DeSR is controlled by a number of parameters supplied through a parameter file. Each parameter describes a feature and from which tokens to extract it. Tokens are referred through positive numbers for input tokens and negative numbers for tokens on the stack. For example

```
PosFeatures      -2 -1 0 1 2 3
```

means to use the POS tag of the first two tokens on the stack and of the first four tokens on the input. The parameter `PosPrev` refers to the POS of the preceding token in the original sentence, `PosLeftChild` refers to the POS of the left children of a token, `PastActions` tells how many previous actions to include as features.

The selection process was started from the following base feature model:

```
LexFeatures      -1 0 1
LemmaFeatures    -2 -1 0 1 2 3
LemmaPrev        -1 0
LemmaSucc        -1 0
LemmaLeftChild   -1 0
LemmaRightChild  -1
MorphoFeatures   -1 0 1 2
PosFeatures      -2 -1 0 1 2 3
PosNext          -1 0
PosPrev          -1 0
PosLeftChild     -1 0
PosRightChild    -1 0
CPosFeatures     -1 0 1
DepFeatures      -1 0
DepLeftChild     -1 0
DepRightChild    -1
PastActions      1
```

The selection process produced different variants for each language, sometimes suggesting dropping certain intermediate features, like the lemma of the third next input token in the case of Catalan:

```
LemmaFeatures    -2 -1 0 1 3
LemmaPrev        0
LemmaSucc        -1
LemmaLeftChild   0
LemmaRightChild  -1
PosFeatures      -2 -1 0 1 2 3
PosPrev          0
PosSucc          -1
PosLeftChild     -1 0
PosRightChild    -1 0
CPosFeatures     -1 0 1
MorphoFeatures   0 1
DepLeftChild     -1 0
DepRightChild    -1
```

For Italian, instead, we ran a series of tests in parallel using a set of manually prepared feature models. The best of these models achieved a LAS of 80.95%. The final run used this model with the addition of the morphological agreement feature discussed below.

English was the only language for which no feature selection was done and for which lexical features

Task	LAS			UAS		
	1st	DeSR	Avg	1st	DeSR	Avg
Arabic	76.52	72.66	68.34	86.09	82.53	78.84
Basque	76.92	69.48	68.06	82.80	76.86	75.15
Catalan	88.70	86.86	79.85	93.40	91.41	87.98
Chinese	84.69	81.50	76.59	88.94	86.73	81.98
Czech	80.19	77.37	70.12	86.28	83.40	77.56
English	89.61	85.85	80.95	90.63	86.99	82.67
Greek	76.31	73.92	70.22	84.08	80.75	77.78
Hungarian	80.27	76.81	71.49	83.55	81.81	76.34
Italian	84.40	81.34	78.06	87.91	85.54	82.45
Turkish	79.81	76.87	73.19	86.22	83.56	80.33

Table 1. Multilingual track official scores**.**

were used. English is also the language where the official score is significantly lower than what we had been getting on our development set (90.01% UAS).

6.2 Prepositional Attachment

Certain languages, such as Catalan, use detailed dependency labeling, that for instance distinguish between adverbials of location and time. We exploited this information by introducing a feature that captures the entity type of a child of the top word on the stack or in the input. During training a list of nouns occurring in the corpus as dependent on prepositions with label CCL (meaning 'complement of location' for Catalan) was created and similarly for CCT (complement of time). The entity type TIME is extracted as a feature depending on whether the noun occurs in the time list more than α times than in the location list, and similarly for the feature LOCATION. α was set to 1.5 in our experiments.

6.3 Morphological Agreement

Certain languages require gender and number agreement between head and dependent. The feature `MorphoAgreement` is computed for such languages and provided noticeable accuracy improvements.

For example, for Italian, the improvement was from:

LAS: 80.95%, UAS: 85.03%

to:

LAS: 81.34%, UAS: 85.54%

For Catalan, adding this feature we obtained an unofficial score of:

LAS: 87.64%, UAS: 92.20%

with respect to the official run:

LAS: 86.86%, UAS: 91.41%

7 Accuracy

Table 1 reports the accuracy scores in the multilingual track. They are all considerably above the average and within 2% from the best for Catalan, 3% for Chinese, Greek, Italian and Turkish.

8 Performance

The experiments were performed on a 2.4 Ghz AMD Opteron machine with 32 GB RAM. Training the parser using the 2^{nd}-order perceptron on the English corpus required less than 3 GB of memory and about one hour for each iteration over the whole dataset. Parsing the English test set required 39.97 sec. For comparison, we tested the MST parser version 0.4.3 (Mstparser, 2007), configured for second-order, on the same data: training took 73.9 minutes to perform 10 iterations and parsing took 97.5 sec. MST parser achieved:

LAS: 89.01%, UAS: 90.17%

9 Error Analysis on Catalan

The parser achieved its best score on Catalan, so we performed an analysis on its output for this language.

Among the 42 dependency relations that the parser had to assign to a sentence, the largest number of errors occurred assigning *CC* (124), *SP* (33), *CD* (27), *SUJ* (26), *CONJUNCT* (22), *SN* (23).

The submitted run for Catalan did not use the entity feature discussed earlier and indeed 67 errors were due to assigning CCT or CCL instead of CC (generic complement of circumstance). However over half of these appear as underspecified annotation errors in the corpus rather than parser errors.

By adding the `ChildEntityType` feature, which distinguishes better between CCT and CCL, the UAS improved, while the LAS dropped slightly, due to the effect of underspecified annotations in the corpus:

LAS: 87.22%, UAS: 91.71%

A peculiar aspect of the original Catalan corpus was the use of a large number (195) of dependency labels. These labels were reduced to 42 in the version used for CoNNL 2007, in order to make it comparable to other corpora. However, performing some preliminary experiments using the original Catalan collection with all 195 dependency labels, the DeSR parser achieved a significantly better score:

LAS: 88.80%, UAS: 91.43%

while with the modified one, the score dropped to:

LAS: 84.55%, UAS: 89.38%

This suggests that accuracy might improve for other languages as well if the training corpus was labeled with more precise dependencies.

10 Adaptation Track

The adaptation track originally covered two domains, the CHILDES and the Chemistry domain.

The CHILDES (Brown, 1973; MacWhinney, 2000) consists of transcriptions of dialogues with children, typically short sentences of the kind:

```
Would you like more grape juice ?
That 's a nice box of books .
```

Phrases are short, half of them are questions. The only difficulty that appeared from looking at the unlabeled collection supplied for training in the domain was the presence of truncated terms like *goin* (for *going*), *d* (for *did*), etc. However none of these unusually spelled words appeared in the test set, so a normal English parser performed reasonably well on this task. Because of certain inconsistencies in the annotation guidelines, the organizers decided to make this task optional and hence we submitted just the parse produced by the parser trained for English.

For the second adaptation task we were given a large collection of unlabeled data in the chemistry domain (Kulick et al, 2004) as well as a test set of 5000 tokens (200 sentences) to parse (*english_pchemtbtb_test.conll*).

There were three sets of unlabeled documents: we chose the smallest (*unlab1*) consisting of over 300,000 tokens (11663 sentences). *unlab1* was tokenized, POS and lemmas were added using our version of TreeTagger (Schmid, 1994), and lemmas replaced with stems, which had turned out to be more effective than lemmas. We call this set *pchemtb_unlab1.conll*.

We trained the DeSR parser on English using *english_ptb_train.conll*, the WSJ PTB collection provided for CoNLL 2007. This consists of WSJ sections 02-11, half of the usual set 02-23, for a total of 460,000 tokens with dependencies generated with the converter by Johansson and Nugues (2007).

We added stems and produced a parser called *DeSRwsj*. By parsing *english_pchem_test.conll* with *DeSRwsj* we obtained *pchemtb_test_base.desr*, our baseline for the task.

By visual inspection using DgAnnotator (DgAnnotator, 2006), the parses looked generally correct. Most of the errors seemed due to improper handling of conjunctions and disjunctions. The collection in fact contains several phrases like:

```
Specific antibodies raised against
P450IIB1 , P450 IA1 or IA2 ,
P450IIE1 , and P450IIIA2 inhibited
the activation in liver microsomes
from rats pretreated with PB , BNF ,
INH and DEX respectively
```

The parser did not seem to have much of a problem with terminology, possibly because the supplied gold POS were adequate.

For the adaptation we proceeded as follows. We parsed *pchemtb_unlab1.conll* using *DeSRwsj* obtaining *pchemtb_unlab1.desr*.

We then extracted a set of 12,500 sentences from *ptb_train.conll* and 7,500 sentences from *pchemtb_unlab1.desr*, creating a corpus of 20,000 sentences called *combined.conll*. In both cases the selection criteria was to choose sentences shorter than 30 tokens.

We then trained a low accuracy parser (called *DesrCombined*) on *combined.conll*, by using a 1st-order averaged perceptron. *DesrCombined* was used to parse *english_ptb_train.conll*, the original training corpus for English. By comparing this parse with the original, one can detect where such parser makes mistakes. The rationale for using an inaccurate parser is to obtain parses with many errors so that they form a suitably large training set for the next step: parser revision.

We then used a parsing revision technique (Attardi and Ciaramita, 2007) to learn how to correct these errors, producing a parse reviser called *DesrReviser*. The revision technique consists of comparing the parse trees produced by the parser with the gold standard parse trees, from the annotated corpus. Where a difference is noted, a revision rule is determined to correct the mistake. Such rules consist in movements of a single link to a different head. Learning how to revise a parse tree consists in training a classifier on a set of training examples consisting of pairs $\langle(w_i, d, w_j), t_i\rangle$, i.e. the link to be modified and the transformation rule to apply. Attardi and Ciaramita (2007) showed that 80% of the corrections can be typically dealt with just 20 tree revision rules. For the adaptation track we limited the training to errors recurring at least 20 times and to 30 rules.

DesrReviser was then applied to *pchemtb_test_base.desr* producing *pchemtb_test_rev.desr*, our final submission.

Many conjunction errors were corrected, in particular by moving the head of the sentence from a coordinate verb to the conjunction *'and'* linking two coordinate phrases.

The revision step produced an improvement of 0.42% LAS over the score achieved by using just the base *DeSRwsj* parser.

Table 2 reports the official accuracy scores on the closed adaptation track. DeSR achieved a close second best UAS on the *ptchemtb* test set and third best on *CHILDES*. The results are quite encouraging, particularly considering that the revision step does not yet correct the dependency labels and that our base English parser had a lower rank in the multilingual track.

Task	LAS			UAS		
	1st	DeSR	Avg	1st	DeSR	Avg
CHILDES				61.37	58.67	57.89
Pchemtb	81.06	80.40	73.03	83.42	83.08	76.42

Table 2. Closed adaptation track scores.

Notice that the adaptation process could be iterated. Since the combination *DeSRwsj+DesrReviser* is a more accurate parser than *DeSRwsj*, we could use it again to parse *pchemtb_unlab1.conll* and so on.

11 Conclusions

For performing multilingual parsing in the CoNLL 2007 shared task we employed DeSR, a classifier-based Shift/Reduce parser. We used a second order averaged perceptron as classifier and achieved accuracy scores quite above the average in all languages. For proper comparison with other approaches, one should take into account that the parser is incremental and deterministic; hence it is typically faster than other non linear algorithms.

For the adaptation track we used a novel approach, based on the technique of tree revision, applied to a parser trained on a corpus combining sentences from both the training and the adaptation domain. The technique achieved quite promising results and it also offers the interesting possibility of being iterated, allowing the parser to incorporate language knowledge from additional domains.

Since the technique is applicable to any parser, we plan to test it also with more accurate English parsers.

Acknowledgments. The following treebanks were used for training the parser: (Aduriz et al., 2003; Böhmovà et al., 2003; Chen et al., 2003; Hajiç et al., 2004; Marcus et al., 1993; Martí et al., 2002; Montemagni et al. 2003; Oflazer et al., 2003; Prokopidis et al., 2005; Csendes et al., 2005). Ryan McDonald and Jason Baldridge made available mstparser and helped us using it. We gratefully acknowledge Hugo Zaragoza and Ricardo Baeza-Yates for supporting the first author during a sabbatical at Yahoo! Research Barcelona.

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)*, 201–204.

G. Attardi. 2006. Experiments with a Multilanguage non-projective dependency parser. In *Proc. of the Tenth CoNLL, 2006*.

G. Attardi, M. Ciaramita. 2007. Tree Revision Learning for Dependency Parsing. In *Proc. of NAACL/HLTC 2007*.

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.

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, 231–248.

M. Ciaramita, G. Attardi. 2007. Dependency Parsing with Second-Order Feature Maps and Annotated Semantic Information. *Proc. of the 12th International Workshop on Parsing Technologies* (*IWPT*), *2007*.

K. Crammer, Y. Singer. 2003. Ultraconservative Online Algorithms for Multiclass Problems. *Journ. of Machine Learning Research*.

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

DgAnnotator. 2006. http://medialab.di.unipi.it/Project/Parser/DgAnnotator/.

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*, 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)*.

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, et al. 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proc. of HLT-EMNLP*.

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

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, 189–210.

Mstparser 0.4.3. 2007. http://sourceforge.net/projects/mstparser/

J. Nivre, et al. 2004. Memory-based Dependency Parsing. In *Proc.s of the Eighth CoNLL*, ed. H. T. Ng and E. Riloff, Boston, Massachusetts, 49–56.

J. Nivre and J. Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proc. of the 43rd Annual Meeting of the ACL*, 99–106.

J. Nivre and M. Scholz. 2004. Deterministic Dependency Parsing of English Text. In *Proc. of COLING 2004*, Geneva, Switzerland, 64–70.

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, 261–277.

M.F. Porter. 2001. Snowball Stemmer. http://www.snowball.tartarus.org/

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.

H. Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proc. of International Conference on New Methods in Language Processing*.

V. N. Vapnik. 1998. The Statistical Learning Theory. Springer.

H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proc. of the 8th International Workshop on Parsing Technologies* (*IWPT*), 195–206.