MaltParser at the EVALITA 2009 Dependency Parsing Task

Alberto Lavelli[1], Johan Hall[2], Jens Nilsson[3], and Joakim Nivre[2]

[1] FBK-irst,
via Sommarive 18, I-38123 Povo (TN), Italy
lavelli@fbk.eu
[2] Department of Linguistics and Philology Uppsala University
Box 635, 75126 Uppsala, Sweden
{joakim.nivre,johan.hall}@lingfil.uu.se
[3] School of Mathematics and Systems Engineering
Växjö University
35195 Växjö, Sweden
jens.nilsson@vxu.se

Abstract. This paper describes our participation in the EVALITA 2009 Dependency Parsing Task with a version of MaltParser. Reusing feature models developed in the CoNLL shared task 2007, we evaluated four different parsing algorithms implemented in MaltParser and found that the best results were achieved with Covington's non-projective parsing algorithm. In the final evaluation, our system finished third in the main task and second in the pilot task.

Keywords: Dependency parsing, Italian.

1 Introduction

In the Dependency Parsing Task of EVALITA 2009, dependency parsers are applied to two different Italian dependency treebanks, the Turin University Treebank (TUT[4]) and the Italian Syntactic-Semantic Treebank (ISST-CoNLL [1]). A previous version of TUT was used in 2007 for the first edition of the EVALITA Dependency Parsing Task [2], while a previous version of ISST-CoNLL was used in the CoNLL-2007 shared task [3].

The parser used in this paper is MaltParser,[5] a system for data-driven dependency parsing that can be used to induce a parsing model from treebank data and to parse new data using the induced model. MaltParser was one of the top performing systems in the multilingual track of the CoNLL shared tasks on dependency parsing in 2006 and 2007 [4, 5]. In 2007, an ensemble system composed of six different instances of MaltParser attained the highest labeled attachment score for Italian, while a single-parser version of the system was ranked in sixth place. All the results reported in this paper are for single-parser systems.

[4] http://www.di.unito.it/~tutreeb/
[5] Freely available at http://www.maltparser.org/

2 System Description

MaltParser [6] implements the transition-based approach to dependency parsing, which has two essential components:

- A nondeterministic transition system for mapping sentences to dependency trees
- A classifier that predicts the next transition for every possible system configuration

Given these two components, dependency parsing can be performed as greedy deterministic search through the transition system, guided by the classifier. With this technique, it is possible to perform parsing in linear time for projective dependency trees and quadratic time for arbitrary (non-projective) trees [7].

2.1 Transition Systems

MaltParser has four built-in transition systems:

- Nivre's arc-eager system [8]
- Nivre's arc-standard system [9]
- Covington's projective system [10]
- Covington's non-projective system [10]

The two versions of Nivre's transition system are inspired by shift-reduce parsing and use a single stack to store partially processed words. The only difference is that the arc-standard version builds trees strictly bottom-up, while the arc-eager version builds structure incrementally from left to right. In both cases, the system is restricted to projective dependency trees. Covington's system uses two stacks and can therefore process arbitrary non-projective trees, but projectivity can be enforced by restricting the use of the second stack. A more detailed description of all four systems, including proofs of correctness and complexity can be found in [7].

2.2 Classifiers

Classifiers can be induced from treebank data using a wide variety of different machine learning methods, but all experiments reported below use support vector machines with a polynomial kernel, as implemented in the LIBSVM package [11] included in MaltParser. The task of the classifier is to map a high-dimensional feature vector representation of a parser configuration to the optimal transition out of that configuration. Features typically represent word forms, parts of speech and other linguistic attributes of words that appear near the top of the stack(s) or in the input buffer. For the experiments reported below, we have reused the feature representations that gave the best performance for the various transition systems in the 2007 CoNLL shared tasks.[6]

[6] More information about the features can be found at `http://maltparser.org/conll/conll07/`.

2.3 Pseudo-Projective Parsing

As noted earlier, three of the four transition systems used in the experiments can in principle only derive projective dependency trees. In order to overcome this limitation, we have also experimented with pseudo-projective parsing [12], which is a technique for recovering non-projective dependencies with a strictly projective parser. First, all dependency trees in the training set are projectivized by moving arcs in the tree and encoding information about these movements using complex arc labels. A parser trained on these transformed training data will ideally learn to produce trees that are strictly projective but where some arcs have complex labels indicating that they have undergone movement. These arcs can then be moved back in a post-processing step, guided by the extended arc labels, which results in non-projective trees. This technique has been combined with the two versions of Nivre's transition system and with the projective version of Covington's system.

3 Experimental results

We compared the four transition systems (Nivre standard, Nivre eager, Covington projective, Covington non-projective), in the first three cases with and without pseudo-projective parsing, using the feature representations that produced the best results on Italian for the MaltParser system at CoNLL-2007.

3.1 Results on the training set

First of all, we report the results obtained on the training set, used to choose the transition system for parsing the test set. Two different experimental setups were adopted for the two treebanks. For TUT, we adopted a 10-fold cross validation on the entire training set. For ISST, given that there was an explicit distinction between a training and a development set, we trained the parser on the training set and tested it on the development set. Tables 1 and 2 show the results for TUT and for ISST respectively.

In Table 1 the results of the two best performing systems at EVALITA 2007 are reported too. Note that the best performing parser (UniTO Lesmo) was a rule-based parser developed in parallel with TUT and tuned on the data set. The second parser (UniPI Attardi) is a multilingual deterministic shift-reduce dependency parser that handles nonprojective dependencies incrementally and learns by means of a second-order multiclass averaged perceptron classifier.

In Table 2, besides the results on the ISST-EVALITA development set, the following results obtained by parsers at CoNLL-2007 on Italian are reported: the best performing system, MaltParser (single-parser system with ensemble system in parentheses), and average performance.

3.2 Results on the test set

Given the results shown in Tables 1 and 2, Covington's non-projective system was chosen for performing the official run on both treebanks. In [13] the official results for the

Table 1. Dependency parsing – main subtask: TUT; results of 10-fold cross validation.

	LAS	UAS	LA
Nivre standard	78.17	85.70	85.34
Nivre standard pseudo-projective	78.19	85.68	85.43
Nivre eager	80.65	88.42	86.97
Nivre eager pseudo-projective	80.58	88.34	86.95
Covington projective	83.27	87.70	88.96
Covington projective pseudo-projective	83.28	87.72	89.42
Covington non-projective	83.43	87.91	89.57
UniTo_Lesmo @ EVALITA-2007	86.94	90.90	91.59
UniPi_Attardi @ EVALITA-2007	77.88	88.43	83.00
Average @ EVALITA-2007	72.48	83.10	78.63

Table 2. Dependency parsing – pilot Subtask: ISST-CoNLL; results on development set.

	LAS	UAS	LA
Nivre standard	76.57	81.17	84.20
Nivre standard pseudo-projective	79.31	83.80	88.29
Nivre eager	80.96	85.34	89.09
Nivre eager pseudo-projective	81.21	85.63	89.55
Covington projective	80.68	84.79	88.54
Covington projective pseudo-projective	81.12	85.13	89.26
Covington non-projective	81.99	85.80	90.12
Best @ CoNLL-2007	84.40	87.91	
MaltParser @ CoNLL-2007	82.48 (84.40)	86.26 (87.77)	
Average @ CoNLL-2007	78.06	82.45	

dependency parsing task can be found. Our system obtained the third best result in the main subtask (LAS: 86.50 vs. 88.73 and 88.68 of the two best performing systems) and the second best result on the pilot subtask (LAS: 80.54 vs. 83.38 of the best performing system). Note that the difference between the results of the two top-performing systems in the main task is not statistically significant.

While the results on the TUT test set with the different transition systems (see Table 3) are coherent with those performed on the training set, those on the ISST test set (Table 4) show a different pattern. While on the training set the best system was Covington non-projective with Nivre eager pseudo-projective as second best, on the test set the situation is the opposite. However, the difference in performance between the first two results was evaluated not to be statistically significant (using the CoNLL-2007 procedure).

Table 3. Dependency parsing – main subtask: results obtained with different transition systems on the test set.

	LAS	UAS	LA
Nivre standard	81.75	90.13	86.36
Nivre standard pseudo-projective	81.56	89.73	86.36
Nivre eager	83.03	91.28	87.21
Nivre eager pseudo-projective	82.88	91.15	87.20
Covington projective	86.51	90.98	90.24
Covington projective pseudo-projective	86.48	90.98	90.77
Covington non-projective	86.50	90.88	90.96

Table 4. Dependency parsing – pilot subtask: results obtained with different transition systems on the test set.

	LAS	UAS	LA
Nivre standard	76.71	81.54	84.45
Nivre standard pseudo-projective	77.77	82.72	86.85
Nivre eager	80.84	85.55	88.70
Nivre eager pseudo-projective	81.12	85.59	89.32
Covington projective	79.92	84.11	87.79
Covington projective pseudo-projective	80.30	84.79	88.43
Covington non-projective	80.54	84.85	88.88

4 Conclusions

Although our system cannot compete with the best performing system on the two test sets, our results are nevertheless fair given that we have not performed any feature optimization but simply reused the models developed for the CoNLL shared task 2007. It is therefore likely that the accuracy could be improved further.

The dependency representations produced by MaltParser trained on the ISST-CoNLL treebank have been used by FBK participants in other EVALITA 2009 tasks, e.g., Textual Entailment and Local Entity Detection and Recognition. To make MaltParser usable within other more complex NLP systems (e.g., textual entailment, question answering, ...) we plan to integrate it in the TextPro tool suite [14].

References

1. Montemagni, S., Barsotti, F., Battista, M., Calzolari, N., Corazzari, O., Lenci, A., Zampolli, A., Fanciulli, F., Massetani, M., Raffaelli, R., Basili, R., Pazienza, M.T., Saracino, D., Zanzotto, F., Mana, N., Pianesi, F., Delmonte, R.: Building the Italian Syntactic-Semantic Treebank. In: Abeillé, A. (ed.) Building and Using syntactically annotated corpora, pp. 189–210. Kluwer, Dordrecht (2003)
2. Bosco, C., Mazzei, A., Lombardo, V., Attardi, G., Corazza, A., Lavelli, A., Lesmo, L., Satta, G., Simi, M.: Comparing italian parsers on a common treebank: the EVALITA experience.

In: Proceedings of the Sixth International Conference on Language Resources and Evaluation. Marrakech, Morocco (2008)
3. Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., Yuret, D.: The CoNLL 2007 shared task on dependency parsing. In: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 915–932. Prague, Czech Republic (2007)
4. Nivre, J., Hall, J., Nilsson, J., Eryiğit, G., Marinov, S.: Labeled pseudo-projective dependency parsing with support vector machines. In: Proceedings of the 10th Conference on Computational Natural Language Learning, pp. 221–225 (2006)
5. Hall, J., Nilsson, J., Nivre, J., Eryigit, G., Megyesi, B., Nilsson, M., Saers, M.: Single malt or blended? A study in multilingual parser optimization. In: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 933–939. Prague, Czech Republic (2007)
6. Nivre, J., Hall, J., Nilsson, J.: Maltparser: A data-driven parser-generator for dependency parsing. In: Proceedings of the 5th International Conference on Language Resources and Evaluation, pp. 2216–2219 (2006)
7. Nivre, J.: Algorithms for deterministic incremental dependency parsing. Computational Linguistics, vol. 34, pp. 513–553 (2008)
8. Nivre, J.: An efficient algorithm for projective dependency parsing. In: Proceedings of the 8th International Workshop on Parsing Technologies, pp. 149–160 (2003)
9. Nivre, J.: Incrementality in deterministic dependency parsing. In: Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together (ACL), pp. 50–57 (2004)
10. Covington, M.A.: A fundamental algorithm for dependency parsing. In: Proceedings of the 39th Annual ACM Southeast Conference, pp. 95–102 (2001)
11. Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm (2001)
12. Nivre, J., Nilsson, J.: Pseudo-projective dependency parsing. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL), pp. 99–106 (2005)
13. Bosco, C., Montemagni, S., Mazzei, A., Lombardo, V., dell'Orletta, F., Lenci, A.: Evalita'09 Parsing Task: comparing dependency parsers and treebanks. In: Proceedings of EVALITA 2009 (2009)
14. Pianta, E., Girardi, C., Zanoli, R.: The TextPro tool suite. In: Proceedings of the Sixth International Conference on Language Resources and Evaluation. Marrakech, Morocco (2008)