10.12871/clicit201423

Comparing State-of-the-art Dependency Parsers for the EVALITA 2014 Dependency Parsing Task

Alberto Lavelli
FBK-irst
via Sommarive, 18 - Povo
I-38123 Trento (TN) - ITALY
lavelli@fbk.eu

Abstract

English. This paper describes our participation in the EVALITA 2014 Dependency Parsing Task. In the 2011 edition we compared the performance of MaltParser with the one of an ensemble model, participating with the latter. This year, we have compared the results obtained by a wide range of state-of-the-art parsing algorithms (MaltParser, the ensemble model made available by Mihai Surdeanu, MATE parsers, TurboParser, ZPar). When evaluated on the development set according to the standard measure (i.e., Labeled Accuracy Score, LAS), three systems have obtained results whose difference is not statistically significant. So we have decided to submit the results of the three systems at the official competition. In the final evaluation, our best system, when evaluated according to LAS, ranked fourth (with a score very close to the best systems), and, when evaluated on the Stanford Dependencies, ranked fifth. The efforts reported in this paper are part of an investigation on how simple it is to apply freely available state-of-the-art dependency parsers to a new language/treebank.

Italiano. *Questo articolo descrive la partecipazione al Dependency Parsing Task a EVALITA 2014. Nell'edizione 2011 avevamo confrontato le prestazioni di MaltParser con un ensemble model, partecipando con quest'ultimo. Quest'anno abbiamo confrontato i risultati ottenuti da un insieme di algoritmi di parsing allo stato dell'arte (MaltParser, l'ensemble model di Mihai Surdeanu, i MATE parser, TurboParser, ZPar). Valutati sul development set in base alla misura standard (Labeled Accuracy Score, LAS), tre sistemi hanno ottenuto risultati le cui differenze non sono statisticamente significativi. Così abbiamo deciso di sottomettere i risultati dei tre sistemi alla competizione. Nella valutazione ufficiale, il nostro miglior sistema è risultato quarto, valutato in base a LAS (con un valore molto vicino a quello dei migliori sistemi) ed è risultato quinto, valutato in base alle Stanford Dependency. Gli sforzi riportati in questo articolo sono parte di un'indagine su quanto è facile applicare analizzatori sintattici a dipendenza liberamente disponibili a una nuova lingua / treebank.*

1 Introduction

Recently, there has been an increasing interest in dependency parsing, witnessed by the organisation of a number of shared tasks, e.g. Buchholz and Marsi (2006), Nivre et al. (2007). Concerning Italian, there have been tasks on dependency parsing in all the editions of the EVALITA evaluation campaign (Bosco et al., 2008; Bosco et al., 2009; Bosco and Mazzei, 2011). In the 2014 edition, the task on dependency parsing exploits the Italian Stanford Dependency Treebank (ISDT), a new treebank featuring an annotation based on Stanford Dependencies (de Marneffe and Manning, 2008).

This paper reports the efforts involved in applying several state-of-the-art dependency parsers for comparing their performance and participating in the EVALITA 2014 task on dependency parsing. Apart from participating in the EVALITA 2014 task, a second motivation was to investigate how simple is to apply freely available state-of-the-art dependency parsers to a new language/treebank following the instructions available together with the code and possibly having a few interactions

with the developers (Lavelli, 2014).

As in many other NLP fields, there are very few comparative articles when the performance of different parsers is compared. Most of the papers simply present the results of the newly proposed approach and compare them with the results reported in previous articles. In other cases, the papers are devoted to the application of the same tool to different languages/treebanks.

It is important to stress that the comparison concerns tools used more or less out of the box and that the results cannot be used to compare specific characteristics like: parsing algorithms, learning systems, ...

2 Description of the Systems

The choice of the parsers used in this study started from the two we already applied at EVALITA 2011, i.e. MaltParser and the ensemble method described by Surdeanu and Manning (2010). We then identified a number of other dependency parsers that in the last years have shown state-of-the-art performance, that are freely available and with the possibility of training on new treebanks. The ones included in the preliminary comparison reported in this paper are the MATE dependency parsers, TurboParser, and ZPar. In the near future, we plan to include other dependency parsers in our comparison. We have not been able to exploit some of the dependency parsers because of lack of time and some others because of different reasons: they are not yet available online, they lack documentation on how to train the parser on new treebanks (the ClearNLP dependency parser), they have limitations in the encoding of texts (input texts only in ASCII and not in UTF-8; the Redshift dependency parser), ...

MaltParser (Nivre et al., 2006) (version 1.8) 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 (Nivre, 2008). MaltParser includes different built-in transition systems, different classifiers and techniques for recovering non-projective dependencies with strictly projective parsers.

The ensemble model made available by Mihai Surdeanu (Surdeanu and Manning, 2010)[1] implements a linear interpolation of several linear-time parsing models (all based on MaltParser). In particular, it combines five different variants of MaltParser (Nivre's arc-standard left-to-right, Nivre's arc-eager left-to-right, Covington's non projective left-to-right, Nivre's arc-standard right-to-left, Covington's non projective right-to-left) as base parsers. Each individual parser runs in its own thread, which means that, if a sufficient number of cores are available, the overall runtime is essentially similar to a single MaltParser. The resulting parser has state-of-the-art performance yet it remains very fast.

The MATE tools[2] include both a graph-based parser (Bohnet, 2010) and a transition-based parser (Bohnet and Nivre, 2012; Bohnet and Kuhn, 2012). For the languages of the 2009 CoNLL Shared Task, the graph-based MATE parser reached accuracy scores similar or above the top performing systems with fast processing. The speed improvement is obtained with the use of Hash Kernels and parallel algorithms. The transition-based MATE parser is a model that takes into account complete structures as they become available to rescore the elements of a beam, combining the advantages of transition-based and graph-based approaches.

TurboParser (Martins et al., 2013)[3] (version 2.1) is a C++ package that implements graph-based dependency parsing exploiting third-order features.

ZPar (Zhang and Nivre, 2011) is a transition-based parser implemented in C++. ZPar supports multiple languages and multiple grammar formalisms. ZPar has been most heavily developed for Chinese and English, while it provides generic support for other languages. It leverages a global discriminative training and beam-search

[1] http://www.surdeanu.info/mihai/ensemble/

[2] https://code.google.com/p/mate-tools/

[3] http://www.ark.cs.cmu.edu/TurboParser/

			collapsed and propagated		
	LAS		P	R	F_1
MATE stacking (TurboParser)	89.72		82.90	90.58	86.57
Ensemble (5 parsers)	89.72		82.64	90.34	86.32
ZPar	**89.53**		84.65	92.11	88.22
MATE stacking (transition-based)	89.02		82.09	89.77	85.76
TurboParser (model_type=full)	88.76		83.32	90.71	86.86
TurboParser (model_type=standard)	88.68		83.07	90.55	86.65
MATE graph-based	88.51		81.72	89.42	85.39
MATE transition-based	88.32		80.70	89.40	84.82
Ensemble (MaltParser v.1.8)	88.15		80.69	88.34	84.34
MaltParser (Covington non proj)	87.79		81.50	87.39	84.34
MaltParser (Nivre eager -PP head)	**87.53**		81.30	88.78	84.88
MaltParser (Nivre standard - MaltOptimizer)	86.35		81.17	89.04	84.92
Ensemble (MaltParser v.1.3)	86.27		78.57	86.28	82.24

Table 1: Results on the EVALITA 2014 development set without considering punctuation. The second column reports the results in term of Labeled Attachment Score (LAS). The score is in bold if the difference with the following line is statistically significant. The three columns on the right show the results in terms of Precision, Recall and F_1 for the collapsed and propagated relations.

			collapsed and propagated		
	LAS		P	R	F_1
MATE stacking (transition-based)	87.67		79.14	88.14	83.40
Ensemble (5 parsers)	87.53		78.28	88.09	82.90
MATE stacking (TurboParser)	87.37		79.13	87.97	83.31
MATE transition-based	87.07		78.72	87.16	82.73
MATE graph-based	86.91		78.74	87.97	83.10
ZPar	86.79		80.30	88.93	84.39
TurboParser (model_type=full)	86.53		79.43	89.42	84.13
TurboParser (model_type=standard)	86.45		79.65	89.32	84.21
Ensemble (MaltParser v.1.8)	85.94		76.30	86.38	81.03
MaltParser (Nivre eager -PP head)	**85.82**		78.47	86.06	82.09
Ensemble (MaltParser v.1.3)	85.06		76.36	84.74	80.33
MaltParser (Covington non proj)	84.94		77.24	82.97	80.00
MaltParser (Nivre standard - MaltOptimizer)	84.44		76.53	86.99	81.43

Table 2: Results on the EVALITA 2014 test set without considering punctuation. The second column reports the results in term of Labeled Attachment Score (LAS). The score is in bold if the difference with the following line is statistically significant. The three columns on the right show the results in terms of Precision, Recall and F_1 for the collapsed and propagated relations.

framework.

2.1 Experimental Settings

The level of interaction with the authors of the parsers varied. In two cases (ensemble, MaltParser), we have mainly exploited the experience gained in previous editions of EVALITA. In the case of the MATE parsers, we have had a few interactions with the author who suggested the use of some undocumented options. In the case of TurboParser, we have simply used the parser as it is after reading the available documentation. Concerning ZPar, we have had a few interactions with the authors who helped solving some issues.

As for the ensemble, at the beginning we repeated what we had already done at EVALITA 2011 (Lavelli, 2011), i.e. using the ensemble as it is, simply exploiting the more accurate extended models for the base parsers. The results were unsatisfactory, because the ensemble is based on an old version of MaltParser (v.1.3) that performs worse than the current version (v.1.8). So we decided to apply the ensemble model both to the output produced by the current version of MaltParser and to the output produced by some of the parsers used in this study. In the latter case, we have used the output of the following 5 parsers: graph-based MATE parser, transition-based MATE parser, TurboParser (full model),

			collapsed and propagated		
	LAS		P	R	F_1
Ensemble (5 parsers)	87.22		78.21	87.92	82.78
MATE stacking (transition-based)	**86.99**		78.42	87.70	82.80
MATE transition-based	86.47		78.08	87.11	82.35
ZPar	86.40		79.84	88.27	83.84
TurboParser (model_type=full)	86.35		79.77	89.12	84.19
MATE graph-based	86.34		77.94	87.02	82.23
TurboParser (model_type=standard)	86.32		79.50	89.39	84.16
MATE stacking (TurboParser)	85.87		76.79	86.43	81.32
Ensemble (MaltParser v.1.8)	85.87		76.59	86.58	81.28
MaltParser (Nivre eager -PP head)	**85.66**		78.28	86.89	82.36
MaltParser (Covington non proj)	84.98		77.24	83.24	80.13
Ensemble (MaltParser v.1.3)	84.75		75.52	83.98	79.52
MaltParser (Nivre standard - MaltOptimizer)	84.25		76.29	86.77	81.19

Table 3: Results on the EVALITA 2014 test set after training on the training set only (NO development set) without considering punctuation. The second column reports the results in term of Labeled Attachment Score (LAS). The score is in bold if the difference with the following line is statistically significant. The three columns on the right show the results in terms of Precision, Recall and F_1 for the collapsed and propagated relations.

MaltParser (Nivre's arc-eager, PP-head, left-to-right), and MaltParser (Nivre's arc-eager, PP-head, right-to-left).

Concerning MaltParser, in addition to using the best performing configurations at EVALITA 2011[4], we have used MaltOptimizer[5] (Ballesteros and Nivre, 2014) to identify the best configuration. According to MaltOptimizer, the best configuration is Nivre's arc-standard. However, we have obtained better results using the configurations used in EVALITA 2011. We are currently investigating this issue.

As for the MATE parsers, we have applied both the graph-based parser and the transition-based parser. Moreover, we have combined the graph-based parser with the output of another parser (both the transition-based parser and TurboParser) using stacking. Stacking is a technique of integrating two parsers at learning time[6], where one of the parser generates features for the other.

Concerning ZPar, the main difficulty was the fact that a lot of RAM is needed for processing long sentences (i.e., sentences with more than 100 tokens need 70 GB of RAM). After some interactions with the authors, we were able to understand and fix this issue.

During the preparation of the participation in the task, the experiments were performed using the split provided by the organisers, i.e. training on the training set and testing using the development set.

When applying stacking, we have performed 10-fold cross validation of the first parser on the training set, using the resulting output to provide to the second parser the predictions used during learning. During parsing the output of the first parser (trained on the whole training set and applied to the development set) has been provided to the second parser.

3 Results

In Table 1 we report the parser results on the development set ranked according to decreasing Labeled Accuracy Score (LAS), considering punctuation. The score is in bold if the difference with the following line is statistically significant[7] (the difference is significant only if p-value is less than 0.05). In the three columns on the right of the table the results for the collapsed and propagated relations are shown (both the conversion and the evaluation are performed using scripts provided by the organisers).

In Table 1 we have grouped together the parsers if the differences between their results (in terms of

[4]Nivre's arc-eager, PP-head, and Covington non projective.

[5]http://nil.fdi.ucm.es/maltoptimizer/

[6]Differently from what is done by the ensemble method described above where the combination takes place only at parsing time.

[7]To compute the statistical significance of the differences between results, we have used MaltEval (Nilsson and Nivre, 2008).

LAS) are not statistically significant. As it can be seen, five clusters can be identified.

Note that the computation of the statistical significance of the results was possible only for the standard evaluation (LAS) but not for the evaluation of the recognition of Stanford Dependencies. This is obviously a strong limitation in the possibility of analysing the results. We plan to investigate if it is possible to perform such computation.

An obvious remark is that the ranking of the results according to LAS and according to the recognition of Stanford Dependencies is different. This made the choice of the parsers for the participation difficult, given that the participants would have been ranked based on both measures.

According to the results on the development set, we decided to submit for the official evaluation three models: ZPar, MATE stacking (TurboParser), and the ensemble combining 5 of the best parsers. For the official evaluation, the training was performed using both the training and the development set. In Table 2. you may find the results of all the parsers used in this study (in italics those submitted to the official evaluation). Comparing Table 1 and Table 2, it emerges that some of the parsers show different behaviours between the development and the test set. This calls for an analysis to understand the reasons of such difference. The results of a preliminary analysis are reported in Section 4.

The results obtained by the best system submitted to the official evaluation are: 87.89 (LAS), 81.89/90.45/85.95 (P/R/F_1). According to LAS, our systems were ranked fourth (the ensemble combining 5 of the best parsers), fifth (MATE parser stacking based on TurboParser) and eighth (ZPar). Evaluating using Stanford Dependencies was different. The same systems were ranked ninth, seventh, and fifth respectively. More details about the task and the results obtained by the participants are available in Bosco et al. (2014).

4 Discussion

We are currently analysing the results shown above to understand how to further proceed in our investigation. A general preliminary consideration is that, as expected, approaches that combine the results of different parsers perform better than those based on a single parser model, usually with the drawback of a bigger complexity.

The results shown in Tables 1 and 2 raise a few questions.

The first question concern the fact that some of the parsers (e.g., ZPar) show different behaviours between the development and the test set. This is still true even if we consider the clusters of where the results are not statistically different. To investigate this issue we performed some experiments training on the training set only (not using the development set) and analysing the test set. These results are reported in Table 3. The results show that some parsers have different behaviours on the development set and on the test set, even when considering only the clustering performed taking into account the statistical significance of the difference between different parsers' performance. This issue needs to be further investigated.

The second question concern the discrepancy between the standard evaluation in terms of LAS and the recognition of the Stanford dependencies in terms of Precision, Recall and F_1. For example, the ensemble is our best scoring system according to the standard evaluation, while is our worst system when evaluated on the Stanford dependencies. A crucial element to investigate this issue is the possibility of computing the statistical significance of the difference between the results of the recognition of Stanford Dependencies.

5 Conclusions and Future Work

In the paper we have reported on work in progress on the comparison between several state-of-the-art dependency parsers on the Italian Stanford Dependency Treebank (ISDT) in the context of the EVALITA 2014 dependency parsing task.

In the near future, we plan to widen the scope of the comparison including more parsers and analysing some unexpected behaviours emerged from our experiments.

Finally, we will perform an analysis of the results obtained by the different parsers considering not only their performance but also their behaviour in terms of speed, CPU load at training and parsing time, ease of use, licence agreement, ...

Acknowledgments

This work was partially supported by the EC-funded project EXCITEMENT (FP7ICT-287923). We wish to thank the authors of the parsers for making them freely available. In particular, we would like to thank Bernd Bohnet, Joakim Nivre, Mihai Surdeanu, Yue Zhang, and Yijia Liu for

kindly answering our questions on the practical application of their parsers and for providing useful suggestions.

References

Miguel Ballesteros and Joakim Nivre. 2014. MaltOptimizer: Fast and effective parser optimization. *Natural Language Engineering*, FirstView:1–27, 10.

Bernd Bohnet and Jonas Kuhn. 2012. The best of both worlds – a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 77–87, Avignon, France, April. Association for Computational Linguistics.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea, July. Association for Computational Linguistics.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August. Coling 2010 Organizing Committee.

Cristina Bosco and Alessandro Mazzei. 2011. The EVALITA 2011 parsing task: the dependency track. In *Working Notes of EVALITA 2011*, pages 24–25.

Cristina Bosco, Alessandro Mazzei, Vincenzo Lombardo, Giuseppe Attardi, Anna Corazza, Alberto Lavelli, Leonardo Lesmo, Giorgio Satta, and Maria Simi. 2008. Comparing Italian parsers on a common treebank: the EVALITA experience. In *Proceedings of LREC 2008*.

Cristina Bosco, Simonetta Montemagni, Alessandro Mazzei, Vincenzo Lombardo, Felice DellOrletta, and Alessandro Lenci. 2009. Evalita09 parsing task: comparing dependency parsers and treebanks. In *Proceedings of EVALITA 2009*.

Cristina Bosco, Felice Dell'Orletta, Simonetta Montemagni, Manuela Sanguinetti, and Maria Simi. 2014. The EVALITA 2014 dependency parsing task. In *Proceedings of EVALITA 2014*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. Association for Computational Linguistics.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Manchester, UK, August. Coling 2008 Organizing Committee.

Alberto Lavelli. 2011. An ensemble model for the EVALITA 2011 dependency parsing task. In *Working Notes of EVALITA 2011*.

Alberto Lavelli. 2014. A preliminary comparison of state-of-the-art dependency parsers on the italian stanford dependency treebank. In *Proceedings of the first Italian Computational Linguistics Conference (CLiC-it 2014)*.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jens Nilsson and Joakim Nivre. 2008. MaltEval: an evaluation and visualization tool for dependency parsing. In *Proceedings of LREC 2008*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, June. Association for Computational Linguistics.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 649–652, Los Angeles, California, June. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.