Comparing State-of-the-art Dependency Parsers on the Italian Stanford Dependency Treebank

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

Abstract

English. In the last decade, many accurate dependency parsers have been made publicly available. It can be difficult for non-experts to select a good off-the-shelf parser among those available. This is even more true when working on languages different from English, because parsers have been tested mainly on English treebanks. Our analysis is focused on Italian and relies on the Italian Stanford Dependency Treebank (ISDT). This work is a contribution to help non-experts understand how difficult it is to apply a specific dependency parser to a new language/treebank and choose a parser that meets their needs.

Italiano. *Nell'ultimo decennio sono stati resi disponibili molti analizzatori sintattici a dipendenza. Per i non esperti può essere difficile sceglierne uno pronto all'uso tra quelli disponibili. A maggior ragione se si lavora su lingue diverse dall'inglese, perché gli analizzatori sono stati applicati soprattutto su treebank inglesi. La nostra analisi è dedicata all'italiano e si basa sull'Italian Stanford Dependency Treebank (ISDT). Questo articolo è un contributo per aiutare i non esperti a capire quanto è difficile applicare un analizzatore a una nuova lingua/treebank e a scegliere quello più adatto.*

1 Introduction

In the last decade, there has been an increasing interest in dependency parsing, witnessed by the organisation of various shared tasks, e.g. Buchholz and Marsi (2006), Nivre et al. (2007), Seddah et al. (2013), Seddah et al. (2014). Concerning Italian, there have been tasks on dependency parsing in the first four editions of the EVALITA evaluation campaign (Bosco et al., 2008; Bosco et al., 2009; Bosco and Mazzei, 2011; Bosco et al., 2014). In the 2014 edition, the task on dependency parsing exploited the Italian Stanford Dependency Treebank (ISDT), a treebank featuring an annotation based on Stanford Dependencies (de Marneffe and Manning, 2008).

This paper is a follow-up of Lavelli (2014b) and reports the experience in applying an updated list of state-of-the-art dependency parsers on ISDT. It can be difficult for non-experts to select a good off-the-shelf parser among those available. This is even more true when working on languages different from English, given that parsers have been tested mainly on English treebanks (and in particular on the WSJ portion of the PennTreebank). This work is a contribution to help practitioners understand how difficult it is to apply a specific dependency parser to a new language/treebank and choose a parser to optimize their desired speed/accuracy trade-off.

As in many other NLP fields, there are very few comparative articles where different parsers are directly run by the authors and their performance compared (Daelemans and Hoste, 2002; Hoste et al., 2002; Daelemans et al., 2003). Most of the papers simply present the results of a 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. A notable exception is the study reported in Choi et al. (2015), where the authors present a comparative analysis of ten leading statistical dependency parsers on a multi-genre corpus of English.

It is important to stress that the comparison presented in this paper 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 Parsers

The choice of the parsers used in this study started from the ones already applied in a previous study (Lavelli, 2014b), i.e. MaltParser, the MATE dependency parsers, TurboParser, and ZPar. We then identified a few other freely available dependency parsers that have shown state-of-the-art performance. Some of such parsers are included in the study in Choi et al. (2015) and others have been made publicly available more recently. The additional parsers included in this paper are DeSR, the Stanford Neural Network dependency parser, EmoryNLP, RBG, YARA Parser, and LSTM parser.

Differently from what was done in the previous study, this time we have not included approaches based on combination of parsers' results, such as ensemble or stacking. They usually obtain top performance (see e.g. Attardi and Simi (2014) at EVALITA 2014) but in this case we focus on simplicity and ease of use rather than on absolute performance. Below you may find short descriptions of the parsers reported in the paper.

MaltParser (Nivre et al., 2006) (version 1.8) implements the transition-based approach to dependency parsing, which has two essential components: (i) a nondeterministic transition system for mapping sentences to dependency trees; (ii) a classifier that predicts the next transition for every possible system configuration. MaltParser includes different built-in transition systems, different classifiers and techniques for recovering non-projective dependencies with strictly projective parsers.

The MATE tools[1] 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 (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)[2] (version 2.3) is a C++ package that implements non-projective graph-based dependency parsing exploiting third-order features. The approach uses AD^3, an accelerated dual decomposition algorithm extended to handle specialized head automata and sequential head bigram models.

ZPar (Zhang and Nivre, 2011) (version 0.75) 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 framework.

DeSR (Attardi and Dell'Orletta, 2009) version 1.4.3 is a shift-reduce dependency parser, which uses a variant of the approach of Yamada and Matsumoto (2003). It is capable of dealing directly with non-projective parsing, by means of specific non-projective transition rules (Attardi, 2006). It is highly configurable: one can choose which classifier (e.g. SVM or Multi-Layer Perceptron) and which feature templates to use, and the format of the input, just by editing a configuration file.

EmoryNLP (Choi and McCallum, 2013)[3] (previously ClearNLP) dependency parser (version 1.1.1) uses a transition-based, non-projective parsing algorithm showing a linear-time speed for both projective and non-projective parsing.

The Stanford neural network dependency parser (Chen and Manning, 2014)[4] is a transition-based parser which produces typed dependency parses using a neural network which uses word embeddings as features besides forms and POS tags. It also uses no beam.

RBG (Lei et al., 2014; Zhang et al., 2014b; Zhang et al., 2014a)[5] is based on a low-rank factorization method that enables to map high dimensional feature vectors into low dimensional representations. The method maintains the parameters as a low-rank tensor to obtain low dimensional representations of words in their syntactic roles, and to leverage modularity in the tensor for easy training with online algorithms.

YARA Parser (Rasooli and Tetreault, 2015)[6] is an implementation of the arc-eager dependency model. It uses an average structured perceptron

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

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

[3]http://nlp.mathcs.emory.edu/

[4]http://nlp.stanford.edu/software/nndep.shtml

[5]https://github.com/taolei87/RBGParser

[6]https://github.com/yahoo/YaraParser

as classifier and a beam size of 64. The feature setting is from Zhang and Nivre (2011) with additional Brown cluster features.

LSTM parser (Dyer et al., 2015; Ballesteros et al., 2015)[7] is a transition based dependency parser with state embeddings computed by LSTM RNNs and an alternative char-based model exploiting character embeddings as features. Both the models are applied in the experiments.

The list of parsers is still in progress because the field of dependency parsing is in constant evolution. In mid-May, SyntaxNet, the dependency parser by Google, was made publicly available; a few days later BIST parser (that claims to be "A faster and more accurate parser than Google's McParseface") was announced to become public.

SyntaxNet (Andor et al., 2016)[8], BIST parser (Kiperwasser and Goldberg, 2016)[9], and spaCy[10] are not yet included in our study because we are still trying to make them working in a satisfactory way.

3 Data Set

The experiments reported in the paper are performed on the Italian Stanford Dependency Treebank (ISDT) (Bosco et al., 2013) version 2.0 released in the context of the EVALITA 2014 evaluation campaign on Dependency Parsing for Information Extraction (Bosco et al., 2014)[11]. There are three main novelties with respect to the previously available Italian treebanks: (i) the size of the dataset, much bigger than the resources used in the previous EVALITA campaigns; (ii) the annotation scheme, compliant to *de facto* standards at the level of both representation format (CoNLL) and adopted tagset (Stanford Dependency Scheme); (iii) its being defined with a specific view to supporting information extraction tasks, a feature inherited from the Stanford Dependency scheme.

The training set contains 7,414 sentences (158,561 tokens), the development set 564 sentences (12,014 tokens), and the test set 376 sentences (9,066 tokens).

[7]https://github.com/clab/lstm-parser

[8]https://github.com/tensorflow/models/tree/master/syntaxnet

[9]https://github.com/elikip/bist-parser

[10]https://spacy.io/, https://github.com/spacy-io/spaCy

[11]http://www.evalita.it/2014/tasks/dep_par4IE.

4 Experiments

The level of interaction with the authors of the parsers varied. For MaltParser, MATE parsers, TurboParser, and ZPar we have mainly exploited the experience gained in the context of EVALITA 2014 (Lavelli, 2014a).

Concerning MaltParser, in addition to using the best performing configuration at EVALITA 2014 (Nivre's arc-eager, PP-head), we have used MaltOptimizer[12] (Ballesteros and Nivre, 2014) to identify the best configuration. This was done to be fair to the other parsers, given that MaltParser's best configuration was the result of extensive feature selection at the CoNLL 2006 shared task. According to MaltOptimizer, the best configuration is Nivre's arc-standard.

As for the MATE parsers, we have applied both the graph-based and the transition-based parser.

TurboParser was applied using the three standard configurations (basic, standard, full).

Concerning ZPar, the main difficulty emerged in 2014 (i.e., the fact that sentences with more than 100 tokens needed 70 GB of RAM) is no longer present and so its use is rather straightforward.

As for the new parsers, the only problems during installation concerned an issue with the version of the C++ compiler needed for successfully compiling LSTM parser.

For some of the parsers there is the possibility of exploiting word embeddings (RBG, Stanford parser, LSTM, EmoryNLP) or Brown clustering (YARA parser). As for word embeddings (WEs), we exploited the following (both built using `word2vec`):

- word embeddings of size 300 learned on WackyPedia/itWaC (a corpus of more than 1 billion tokens)[13];
- word embeddings of size 50 produced in the project PAISÀ (Piattaforma per l'Apprendimento dell'Italiano Su corpora Annotati)[14] on a corpus of 250 million tokens.

In general, WEs of size 300 produced an increase in performance, while those of size 50 produced a decrease in performance (with the excep-

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

[13]http://clic.cimec.unitn.it/~georgiana.dinu/down/

[14]http://www.corpusitaliano.it/en/index.html

	LAS	UAS	LA
RBG (full, w/ WEs - size=300)	87.72	90.00	93.03
RBG (standard, w/ WEs - size=300)	87.63	89.91	93.03
RBG (full, w/o WEs)	87.33	89.94	92.41
RBG (standard, w/o WEs)	87.33	89.86	92.43
MATE transition-based	87.07	89.69	92.30
MATE graph-based	86.91	89.53	92.67
TurboParser (model_type=full)	86.53	89.20	92.22
TurboParser (model_type=standard)	86.45	88.96	92.29
ZPar	86.32	88.65	92.40
LSTM (EMNLP 2015, char-based w/ WEs - size=300)	86.07	88.96	92.15
RBG (basic, w/o WEs)	85.99	88.53	91.71
MaltParser (Nivre eager -PP head)	85.82	88.29	91.62
EmoryNLP (w/o WEs)	85.30	87.68	91.51
TurboParser (model_type=basic)	84.90	87.28	91.26
DeSR (MLP)	84.61	87.18	90.79
MaltParser (Nivre standard - MaltOptimizer)	84.44	87.17	90.94
LSTM (ACL 2015, w/ WEs - size=300)	84.20	87.13	90.80
LSTM (EMNLP 2015, char-based w/o WEs)	84.13	87.32	90.75
YARA parser (w/o BCs)	83.87	86.79	90.34
LSTM (ACL 2015, w/o WEs)	83.86	86.95	90.56
Stanford NN dependency parser (w/ WEs - size=50)	83.68	86.50	90.85

Table 1: Results on the EVALITA 2014 test set without considering punctuation, in terms of Labeled Attachment Score (LAS), Unlabeled Attachment Score (UAS) and Label Accuracy (LA).

tion of the Stanford NN dependency parser, which produced results comparable to other parsers with WEs of size 50 and absurdly low results with those of size 300). We were not able to successfully run the EmoryNLP parser with WEs. The use of WEs needs further investigation.

As for the use of Brown clusters (BCs), we are still working to build suitable resources for Italian, so the YARA Parser was used with standard settings and without Brown clusters.

The experiments were performed using the splits provided by the EVALITA 2014 organisers: training on the training set, tuning (if any) using the development set and final test on the test set.

In Table 1 we report the parser results ranked according to decreasing Labeled Accuracy Score (LAS), not considering punctuation. We have grouped together the parsers if the differences between their results (in terms of LAS) are not statistically significant (computation performed using DependAble (Choi et al., 2015)).

The results obtained by the best system submitted to the official evaluation at EVALITA 2014 (Attardi and Simi, 2014) are: 87.89 (LAS), 90.16 (UAS). More details about the task and the results obtained by the participants are available in Bosco et al. (2014).

5 Conclusions

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).

We are already working to widen the scope of the comparison including more parsers and to 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, ...

The next step would be to apply the parsers in a multilingual setting, exploiting the availability of treebanks based on Universal Dependencies in many languages (Nivre et al., 2016)[15].

Acknowledgments

We thank the authors of the parsers for making them freely available, for kindly answering our questions and for providing useful suggestions. We thank the reviewers for valuable suggestions to improve this article.

[15]`http://universaldependencies.org/`

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042.

Giuseppe Attardi and Felice Dell'Orletta. 2009. Reverse revision and linear tree combination for dependency parsing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 261–264, Boulder, Colorado, June. Association for Computational Linguistics.

Giuseppe Attardi and Maria Simi. 2014. Dependency parsing techniques for information extraction. In *Proceedings of EVALITA 2014*.

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 166–170, New York City, June. Association for Computational Linguistics.

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

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. Association for Computational Linguistics.

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, Simonetta Montemagni, and Maria Simi. 2013. Converting Italian treebanks: Towards an Italian Stanford Dependency Treebank. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 61–69, Sofia, Bulgaria, August. Association for Computational Linguistics.

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.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

Jinho D. Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1052–1062, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jinho D. Choi, Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 387–396, Beijing, China, July. Association for Computational Linguistics.

Walter Daelemans and Véronique Hoste. 2002. Evaluation of machine learning methods for natural language processing tasks. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, Las Palmas, Spain.

Walter Daelemans, Véronique Hoste, Fien De Meulder, and Bart Naudts. 2003. Combined optimization of feature selection and algorithm parameters in machine learning of language. In *Proceedings of the 14th European Conference on Machine Learning (ECML 2003)*, Cavtat-Dubronik, Croatia.

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.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July. Association for Computational Linguistics.

Véronique Hoste, Iris Hendrickx, Walter Daelemans, and Antal van den Bosch. 2002. Parameter optimization for machine-learning of word sense disambiguation. *Natural Language Engineering*, 8(4):311–325.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *CoRR*, abs/1603.04351.

Alberto Lavelli. 2014a. Comparing state-of-the-art dependency parsers for the EVALITA 2014 dependency parsing task. In *Proceedings of EVALITA 2014*.

Alberto Lavelli. 2014b. 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)*.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland, June. Association for Computational Linguistics.

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.

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, Marie-Catherine de Marneffe, et al. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *CoRR*, abs/1503.06733.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, et al. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA, October. Association for Computational Linguistics.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109, Dublin, Ireland, August. Dublin City University.

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.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014a. Greed is good if randomized: New inference for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1013–1024, Doha, Qatar, October. Association for Computational Linguistics.

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Globerson. 2014b. Steps to excellence: Simple inference with refined scoring of dependency trees. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 197–207, Baltimore, Maryland, June. Association for Computational Linguistics.