Simple Parser Combination

Alessandro Mazzei, Cristina Bosco

Dipartimento di Informatica, Università degli studi di Torino
Corso Svizzera 185, 10149, Torino, Italy
mazzei@di.unito.it, bosco@di.unito.it

Abstract

This paper presents an ensemble system for dependency parsing: three parsers are separately trained and combined by means of a majority vote. The three parsers are (1) the MATE parser [http://code.google.com/p/mate-tools/], (2) the DeSR parser [http://sites.google.com/site/desrparser/], and (3) the MALT parser [http://maltparser.org/]. The MATE, that was never used before on Italian language, drastically outperforms the other parsers in the SPLeT shared task. Nonetheless, a simple voting combination further improves its performances.

Keywords: ensemble parsing, MATE parser, DeSR parser, MALT parser

1. Introduction

In last few years parsing community devoted great attention to dependency formalisms, and today dependency parsing can be seen as the first step in many applicative NLP systems (Kübler et al., 2009). Larger dependency treebanks and more sophisticated parsing algorithms allowed improved performances of dependency parsers for many languages (Nivre et al., 2007; Hajič et al., 2009).

Indeed, dependency parsing performances constantly increased for Italian. As reported in the Evalita evaluation campaigns specific for NLP systems for Italian (EVALITA 2011 Organization Comitee, 2012), the best scores for Italian dependency parsing (expressed in Labelled Attachment Score, LAS) was 86.94% in 2007, 88.73% in 2009, and 91.23% in 2011 (Bosco and Mazzei, 2012). These results were obtained by using the Turin University Treebank, a dependency treebank for Italian (Bosco and Lombardo, 2004) (see the Section 4.). However, statistical dependency parsing seems to be still improved. On the one hand, new promising specific algorithms for learning and classification are emerging; on the other hand researchers are applying universal machine learning techniques to this specific task. Some are trying to use lager sets of syntactic features (e.g. (McDonald and Pereira, 2006; Carreras, 2007)), while others are trying to apply general techniques *to combine* together the results of various parsers (Zeman and Žabokrtskỳ, 2005; Sagae and Lavie, 2006; Hall et al., 2007; Attardi and dell'Orletta, 2009; Surdeanu and Manning, 2010; Lavelli, 2012).

Our system in the SPLeT competition follows both these mentioned directions. We employ three state of art statistical parsers, which use sophisticated parsing algorithms and advanced feature sets. The three parsers are (1) the MATE parser (Bohnet, 2010), (2) the DeSR parser (Attardi, 2006), (3) the MALT parser (Nivre et al., 2006). Moreover, in our system we combine these three parsers by using two very simple voting algorithms (Breiman, 1996; Zeman and Žabokrtskỳ, 2005). We decided to apply an "out of box" approach, i.e. we apply each parser with its standard configurations for learning and classification.

In the next Sections we first give a short description of the three parsers (Section 2.), then we describe our approach for ensemble parsing (Section 3.) and we report the results of our experiments (Section 4.), before to conclude the paper (Section 5.).

2. The three parsers

In this Section we give a brief description of the three parsers applied in our experiments, i.e. MATE, DeSR and MALT parser.

The MATE parser (Bohnet, 2009; Bohnet, 2010) is a development of the algorithms described in (Carreras, 2007; Johansson and Nugues, 2008). It basically adopts the second order maximum spanning tree dependency parsing algorithm. In particular, Bohnet exploits *hash kernel*, a new parallel parsing and feature extraction algorithm that improves the accuracy as well as the parsing speed (Bohnet, 2010). The MATE performances on English and German, which are 90.14% and 87.64% respectively (LAS), posed this parser at the state of art for these languages (Hajič et al., 2009; Bohnet, 2010; Anders et al., 2010).

The DeSR parser (Attardi, 2006) is a transition (shift-reduce) dependency parser similar to (Yamada and Matsumoto, 2003). It builds dependency structures by scanning input sentences in left-to-right and/or right-to-left direction. For each step, the parser learns from the annotated dependencies if to perform a shift or to create a dependency between two adjacent tokens. DeSR can use different set of rules and includes additional rules to handle non-projective dependencies. The parser can choose among several learning algorithms (e.g Multi Layer Perceptron, Simple Vector Machine), providing user-defined feature models. In our experiments we adopted for DeSR the Multi Layer Perceptron algorithm, which is the same configuration that the parser exploited when it won the Evalita 2009 competition.

The MALT parser (Nivre et al., 2006) implements the transition-based approach to dependency parsing too. In particular MALT has two components: (1) a (non-deterministic) transition system that maps sentences to dependency trees; (2) a classifier, that predicts the next transition for every possible system configuration. MALT performs a greedy deterministic search into the transition system guided by the classifier. In this way, it is possible to perform parsing in linear time for projective dependency

trees and quadratic time for arbitrary (non-projective) trees (Nivre, 2008). MALT has several built-in transition systems, but in our experiments we adopted just the standard "Nivre arc-eager" system, that builds structure incrementally from left to right. Moreover, we use the standard classifier provided by MALT, i.e. the SVM (Simple Vector Machine) basic classifier on the standard "NivreEager" feature model.
In our knowledge this is the fist work that experimented the MATE parser on Italian, while DeSR and MALT parsers have been used in many occasions on Italian (e.g. (Lavelli, 2012; Attardi et al., 2012)), reaching the best results in several contests.

3. The combination algorithms

In order to combine the three parsers we used two very simple algorithms, COM1 and COM2, both implemented in PERL programming language. These algorithms have been previously experimented in (Zeman and Žabokrtskỳ, 2005) and in (Surdeanu and Manning, 2010).
The main idea of the COM1 algorithm is to do a democratic voting among the parsers. For each word[1] of the sentence, the dependency (parent and edge label) assigned to the word by each parser is compared: if at least two parsers assign the same dependency, the COM1 algorithm selects that dependency. In the case that each parser assigns a different dependency to the word, the algorithm selects the dependency assigned by the "best parser", that in our experiments on development set was the MATE parser (see below). As noted by (Zeman and Žabokrtskỳ, 2005), that uses the name *voting* for COM1, this is the most logical decision if it is possible to identify a priori the "best parser", in contrast with the more democratic random choice.
The COM2 algorithm is a variation of the COM1. COM1 is a single word combination algorithm that does not consider the whole dependency structure. This means that incorrect dependency trees can be produced by the COM1 algorithm: cycles and several roots can *corrupts* the "treeness" of the structure. The solution that we adopt in the COM2 algorithm is very simple: if the tree produced by the COM1 algorithm for a sentence is corrupted, then it is selected as dependency structure for that sentence the tree produced by the "best parser" . Again, in accord (Zeman and Žabokrtskỳ, 2005), that uses the name *switching* for COM2, this is the most logical decision since MATE is without doubts the best parser on development score.

4. Experimental Results

We used two machines for experiments. A powerful Linux workstation, equipped with 16 cores, processors 2GHz, and 128 GB ram has been used for the MATE parser, so that the average time for learning is 8 hours. Another Linux workstation equipped with a a single processor 1GHz, and 2 GB ram has been used for learning of the DeSR and MALT parsers, that usually required a couple of hours, and for testing that required several minutes for MATE parser and few minutes for MALT and DeSR parsers. MALT and

[1] In this paper we use the term *word* in a general sense, as synonym of *token*.

```
FOREACH sentence
  FOREACH word IN sentence
    IF [L-DeSR(word)==L-MALT(word)]
      L-COM1(word):=L-DeSR(word)
    ELSE
      L-COM1(word):=L-MATE(word)
```

Table 1: The combination algorithm COM1, that correspond to the *voting* algorithm reported in (Zeman and Žabokrtskỳ, 2005)

```
FOREACH sentence
  FOREACH word IN sentence
    IF [L-DeSR(word)==L-MALT(word)]
      L-COM2(word):=L-DeSR(word)
    ELSE
      L-COM2(WORD):=L-MATE(WORD)
  IF [!CORRECT(TREE-COM2)]
    T-COM2(sentence):=T-MATE(sentence)
```

Table 2: The combination algorithm COM2, that correspond to the *switching* algorithm reported in (Zeman and Žabokrtskỳ, 2005)

DeSR parsers accept as input the CONLL-07 format, that is the format provided by the SPLeT organizers. In contrast MATE accept the CONLL-09 format: simple conversions scripts have been implemented to manage this difference.
In the first experiment, in order to evaluate the "best parser" in the COM1 and COM2 algorithms, we used the ISST training (file: $it\_isst\_train.splet$, $71,568$ words, $3,275$ sentences) as learning set and the ISST development (file: $it\_isst\_test.splet$, $5,165$ words, 231 sentences) as development set. The second row in Table 3 shows the results of the three parsers in this first experiment. MATE parser outperforms the DeSR and MALT parsers: in particular, MATE does $\sim 3\%$ better than DeSR and $\sim 5\%$ better than MALT. On the basis of this result, we decided to use MATE as our "best parser" in the combination algorithms (cf. Section 3.). COM1 and COM2 reach the score of 82.54% and 82.36% respectively, and so both combination algorithms improve the performances of the MATE parser close to the 0.5%.
In the second experiment, we use the whole ISST as learning set (files: $it\_isst\_train.splet$ and $it\_isst\_test.splet$, total $76,733$ words, $3,506$ sentences) and we use the blind file provided by the organizers as test set (file: $it\_EULaw\_test\_blind.splet$, $5,662$ words, 240 sentences, European Directives Laws). The first row in Table 3 shows the results of the three parsers in this second experiment: the value 83.08%, produced by the COM2 algorithm, is the final result of our participation to the SPLeT shared task.[2] Note that there is a $\sim 0.1\%$ difference between the COM1 and COM2 results: similar to (Zeman and Žabokrtskỳ,

[2] A previous value of 84.95% was computed on the basis of two misunderstandings: (1) the NatReg set was added to the learning set and (2) the COM1 algorithm was used (instead of the COM2) since it was not assumed the tree-structure constraint.

	MATE	DeSR	MALT	COM1	COM2	Blended$_{W_2}$	Blended$_{W_3}$	Blended$_{W_4}$
TestSet	82.57	78.68	77.98	83.20	**83.08**	82.23	83.15	83.24
DevSet	81.92	78.99	77.04	82.54	82.36	81.45	82.54	82.63
NatReg	75.76	70.66	70.33	76.28	75.88	74.78	76.07	75.97
Evalita11	89.07	86.26	80.76	89.19	89.16	88.03	89.19	89.19

Table 3: The performances (LAS score) of the three parsers, their simple combination (COM1 and COM2), their blended combination (Blended$_{W_2}$, Blended$_{W_3}$, Blended$_{W_4}$) on the SPLeT test set, development set, Regional laws set and on the Evalita test.

2005; Surdeanu and Manning, 2010) we have 10 corrupted trees in the test set, i.e. $\sim 4\%$ of the total (240 sentences). In Table 4 we detailed the results of the three parsers in the second experiment on the basis of their agreement. When the three parsers agree on the same dependency (Table 4, first row), this happens on $\sim 72\%$ of the words, they have a very high LAS score, i.e. 95.6%. Moreover, DeSR and MALT parsers do better of the MATE parser only when they agree on the same dependency (Table 4, second row). The inspection of the other rows in Table 4 shows that COM1 algorithms has the best possible performance w.r.t. the voting strategy. In other words, COM1 selects all the parser combinations that correspond to higher value of LAS score (cf. the discussion on *minority dependencies* in (Surdeanu and Manning, 2010)).

In the third experiment, we again use the whole ISST as learning set (files: $it\_isst\_train.splet$ and $it\_isst\_test.splet$, total $76,733$ words, $3,506$ sentences), but we use the NatReg file provided by the organizers as test set (file: $it\_NatRegLaw\_test\_blind.splet$, $5,194$ words, 119 sentences, Regional Laws of Piedmont Region). The third row in Table 3 shows the results of the three parsers in this third experiment: in this case we have 75.88% for COM2 algorithm. This lower result can be advocated to the different nature of the domain. It is interesting to note that in this experiment MALT and DeSR parsers give similar results ($\sim 70\%$), while the MATE parser still outperforms them by $\sim 5\%$.

Finally, we performed a fourth experiment on totally different learning and test sets, by using a different Italian Treebank with a different PoS tag set and a different dependency format. We used the Evalita 2011 Development Set as learning set (file: $evalita2011\_train.conll$, $93,987$ words, $3,452$ sentences; balanced corpus of newspapers, laws, wikipedia) and we use the Evalita 2011 test as test set (file: $evalita2011\_test.conll$, $7,836$ words, 300 sentences; balanced corpus), that are produced by using the Turin University Treebank (Bosco and Mazzei, 2012). The fourth row in Table 3 shows the results of the three parsers in this third experiment: in this case we have 89.16% for COM2 algorithm[3]. It is interesting to note that the improvement of the COM2 algorithm w.r.t. with respect to the MATE parser is only $\sim 0.1\%$. In Table 5 we detailed the results of the three parsers in this fourth experiment on the basis of their agreement. Again, when the three parsers agree on the same dependency (Table 5, first row), this happens on $\sim 78\%$ of the words, they have a very high LAS score, i.e. 96.6%. In contrast with the second experiment, here we have a not relevant improvement when DeSR and MALT parser do better of the MATE parser, i.e. only when they agree on the same dependency (Table 5, second row). In other words, on the SPLeT test set the COM1 (and COM2[4]) algorithm do much better than MATE since DeSR and MALT parsers have a good performance (45.8% vs. 30.7%) when they do not agree with the MATE parser: this is not true for the Evalita11 experiment, where DeSR and MALT have 38.8% while the MATE has 35.2%.

Scores					Frequency
MATE 95.6	==	DeSR	==	MALT	**71.99**
MATE 30.7	!=	**DeSR 45.8**	==	**MALT**	**4.20**
MATE	==	**DeSR 67.2**	!=	MALT 14.4	**7.70**
MATE	==	**MALT 59.1**	!=	DeSR 20.0	**8.21**
MATE 31.1	!=	DeSR 14.5	!=	MALT 16.3	**7.89**

Table 4: The detailed performances (LAS score) of the three parsers and their simple combination on the SPLeT blind set, i.e. corresponding to the first row of the Table 3.

Combining versus Re-parsing

Since COM1 can produce corrupted dependency trees, as in (Zeman and Žabokrtskỳ, 2005) we used the COM2 algorithm, that checks the correctness of the tree and, in case of tree-corruption, returns the dependency structure produced by the "best parser" of the ensemble. We hypothesize that this strategy can produce good results in our system since one of the parser of the ensemble drastically outperforms the others. However, a general solution to the tree-corruption problem has been proposed: the re–parsing strategy (Sagae and Lavie, 2006; Hall et al., 2007; Attardi and dell'Orletta, 2009). In re–parsing, a new (not corrupted) dependency tree is produced by taking into account the tree produced by each parser of the ensemble: (Attardi and dell'Orletta, 2009) proposed a approximate top-down algorithm that starts by selecting the highest-scoring root

[3]This score is the third w.r.t. to Evalita 2011 dependency parsing shared task, where the Parsit Parser achieved the best score (91.23%) the DeSR parser achieved the second best score (89.88%).

[4]In the fourth experiment there are 8 corrupted trees.

Scores					Frequency
MATE 96.6	==	DeSR	==	MALT	**78.39**
MATE 35.2	!=	**DeSR 38.8**	==	**MALT**	**3.38**
MATE	==	**DeSR 82.0**	!=	MALT 7.2	**9.17**
MATE	==	**MALT 63.3**	!=	DeSR 19.6	**4.27**
MATE 40.7	!=	DeSR 18.4	!=	MALT 7.9	**4.78**

Table 5: The detailed performances (LAS score) of the three parsers and their combination on the Evalita 2011 test set, i.e. corresponding to the fourth row of the Table 3.

node, then the highest-scoring children and so on; (Sagae and Lavie, 2006; Hall et al., 2007) apply a two-steps algorithm: (1) create a graph funding all the structures produced by the parser on the ensemble, and (2) extract the most probable dependency spanning tree from this graph.
(Surdeanu and Manning, 2010) provided experimental evidence that re–parsing algorithms are a better choice for practical ensemble parsing in out-domains: in order to test this hypothesis we performed a number of experiment by using the "MaltBlender" tool (Hall et al., 2007). In Table 3, the columns **Blended**$_{W_2}$, **Blended**$_{W_3}$, **Blended**$_{W_4}$ report the application of the algorithm described in (Hall et al., 2007). There are three weighting strategies: the results of the three parsers are equally weighted (W_2); the three parsers are weighted according to the total labeled accuracy on a held-out development set (W_3); the parsers are weighted according to labeled accuracy per coarse grained PoS tag on a held-out development set (W_4).
For the first, the second and the third experiments (Table 4, first second and third row), the held-out development set is the SPLeT development set; for the fourth experiment (Table 4, fourth row), the held-out development set is the Evalita 2011 test set. Three evidences seems to emerge from this last experiment: (1) the re–parsing strategies always performs slightly better than COM2 algorithms but not always better than COM1 algorithm; (2) there is no winning weighting strategy for re–parsing; (3) it does not seem that blending performs better out-domain than in-domain.

5. Conclusions

In this paper we described our parsing system for the participation to the SPLeT 2012 Shared Task, and two main issues arise by our contribution. The first issue is that the MATE parser has very good performance on Italian ISST treebank, both in domain and out domain, reaching very good scores; similar results have been obtained on the Turin University Treebank. The second issue is that very simple combination algorithms, as well as more complex blending algorithms, can furthermore improve performance also in situations where a parser outperforms the other ones.
In future research we plan to repeat our experiments on larger set of parsers. In particular, on the basis of the consideration that "diversity" is an important value in ensemble parsing, we want to experiment the possibility to combine together statistical parsers with rule based parsers, e.g. (Lesmo, 2012).

Acknowledgements

We want to thank Alessia Visconti and Francesca Cordero for their valuable (human and machine) time. Moreover we like to thank Felice Dell'Orletta for the suggestion to use MaltBlender in the analysis of the results.

6. References

Björkelund Anders, Bohnet Bernd, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstrations*, pages 33–36, Beijing, China, August. Coling 2010 Organizing Committee.

Giuseppe Attardi and Felice dell'Orletta. 2009. Reverse revision and linear tree combination for dependency parsing. In *HLT-NAACL*, pages 261–264.

Giuseppe Attardi, Maria Simi, and Andrea Zanelli. 2012. Tuning DeSR for the Evalita 2011 Dependency Parsing. In *Working Notes of EVALITA 2011*. CELCT a r.l. ISSN 2240-5186.

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.

Bernd Bohnet. 2009. Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 67–72, Stroudsburg, PA, USA. 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 Vincenzo Lombardo. 2004. Dependency and relational structure in treebank annotation. In *Proceedings of the COLING'04 workshop on Recent Advances in Dependency Grammar*, Geneve, Switzerland.

Cristina Bosco and Alessandro Mazzei. 2012. The evalita 2011 parsing task: the dependency track. In *Working Notes of EVALITA 2011*. CELCT a r.l. ISSN 2240-5186.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961.

EVALITA 2011 Organization Comitee. 2012. *Working Notes of EVALITA 2011*. CELCT a r.l.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared

task: syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 1–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryigit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, CoNLL '08, pages 183–187, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sandra Kübler, Ryan T. McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Alberto Lavelli. 2012. An Ensemble Model for the EVALITA 2011 Dependency Parsing Task. In *Working Notes of EVALITA 2011*. CELCT a r.l. ISSN 2240-5186.

Leonardo Lesmo. 2012. The Turin University Parser at Evalita 2011. In *Working Notes of EVALITA 2011*. CELCT a r.l. ISSN 2240-5186.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006))*, volume 6, pages 81–88.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: a data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, volume 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.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computional Linguistics*, 34(4):513–553, December.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In Robert C. Moore, Jeff A. Bilmes, Jennifer Chu-Carroll, and Mark Sanderson, editors, *HLT-NAACL*. The Association for Computational Linguistics.

Mihai Surdeanu and D. Christopher Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *NAACL*. The Association for Computational Linguistics.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3.

D. Zeman and Z. Žabokrtskỳ. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *International Workshop on Parsing Technologies. Vancouver, Canada*, pages 171–178. Association for Computational Linguistics.