10.12871/clicit201424

Testing parsing improvements with combination and translation in Evalita 2014

Alessandro Mazzei
Dipartimento di Informatica
Università degli Studi di Torino
Corso Svizzera 185, 10149 Torino
mazzei@di.unito.it

Abstract

English. We present the two systems used by the UniTo group to participate to the Evalita 2014 parsing tasks. In particular, we describe the ensemble parser system used for DPIE task and the parsing-by-translation system used for the CLaP task.

Italiano. *Presentiamo i due sistemi utilizzati dal gruppo UniTo per partecipare alla competizione sul parsing di Evalita 2014. Descriviamo il sistema di ensamble parsing usato nel DPIE task e il sistema basato su traduzione usato per partecipare al CLaP task.*

1 Introduction

In the last years a great attention has been devoted to the dependency formalisms and parsers (Kübler et al., 2009). As a consequence many research lines follow new techniques in order to improve the parsing performances, e.g. (Carreras, 2007; Surdeanu and Manning, 2010). However, the specific applicative scenario can draw a clear playground where improvements can be effectively measured. The Evalita 2014 competition on parsing set up two distinct parsing tasks: (1) the Dependency Parsing for Information Extraction (DPIE) task, and (2) the Cross-language Dependency Parsing (CLaP) task.

The DPIE task is the "classical" dependency parsing task for the evaluation of the parsing systems on the Italian language (Bosco and Mazzei, 2012). However, in contrast with the previous editions of the task, the DPIE task adopts the new ISDT treebank (Bosco et al., 2013), which is based on the stanford dependency annotation (de Marneffe and Manning, 2008b), and uses two distinct evaluation measures: the first is the traditional LAS (Labeled Attachment Score), the second is related to the Information Extraction process and is based on a subset of the dependency relations inventory.

The CLaP task wants to test the utility of a standard cross-lingual annotation schema in order to parse foreign languages. By using an universal variant (McDonald et al., 2013) of the Italian ISDT treebank (U-ISDT) as learnin set, one has to parse sentences of several foreign languages.

In order to participate to both the tasks we devised two distinct parsing systems. We participate to the DPIE task by reusing a very simple ensamble parsing system (Mazzei and Bosco, 2012) (Section 2), and we participate to the CLaP task by designing a new cross-language parsing system that uses an on-line translator as external knowledge source (Section 3).

2 The DPIE task

The Dependency Parsing for Information Extraction (DPIE) is the main task of EVALITA 2014 competition on parsing. The focus is on standard dependency parsing of Italian texts. The evaluation is performed on two directions: the LAS (Labelled Attachment Score) as well as a measure on the *collapsed propagated dependencies*, i.e. on simple transformations of a subset of the whole dependency set, which usually are expressed in form of triples (de Marneffe and Manning, 2008a). In particular, the measure based on collapsed propagated dependencies is designed to test the utility of the dependency parsing with respect to the general process of Information Extraction.

In order to participate to this task we decided to reuse the system described in (Mazzei and Bosco, 2012), which follows two promising directions towards the improvement of the performance of the statistical dependency parsers. Indeed, some new promising parsing algorithms use larger sets of syntactic features, e.g. (McDonald and Pereira, 2006; Carreras, 2007), while others apply gen-

eral 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). We explored both these directions in our participation to the DPIE task by combining three state of the art statistical parsers. The three parsers are the MATE[1] parser (Bohnet, 2010) (version 3.61), the DeSR[2] parser (Attardi, 2006) (version 1.4.3), the MALT[3] parser (Nivre et al., 2006) (version 1.7.2). We combined these three parsers by using two very simple voting algorithms (Breiman, 1996; Zeman and Žabokrtskỳ, 2005), on the standard configurations for learning and classification.

The MATE parser (Bohnet, 2009; Bohnet, 2010) is a development of the algorithms described in (Carreras, 2007), and 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 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.

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.

[1] `http://code.google.com/p/mate-tools/`
[2] `http://sites.google.com/site/desrparser/`
[3] `http://maltparser.org/`

2.1 The combination algorithms

We combine the three parsers by using two very simple algorithms: COM1 (Algorithm 1) and COM2 (Algorithm 2), both implemented in the 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

```
foreach sentence do
    foreach word W in the sentence S do
        if DepP2(W) == DepP3(W) then
            Dep-COM1(W) := DepP2(W)
        else
            Dep-COM1(W) := DepP1(W)
        end
    end
end
```

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

is to do a democratic voting among the parsers. For each word in the sentence, the dependency (the parent and the 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*. As noted by (Zeman and Žabokrtskỳ, 2005), who use the name *voting* for COM1, this is the most logical decision if it is possible to identify a priori the best parser, in contrast to the more democratic random choice.

```
foreach sentence do
    foreach word W in the sentence S do
        if DepP2(W) == DepP3( W) then
            Dep-COM2(W) := DepP2(W)
        else
            Dep-COM2(W) := DepP1(W)
        end
    end
    if TREE-COM2(S) is corrupted then
        TREE-COM2(S) := TREE-P1(S)
    end
end
```

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

	MATE	DeSR	MALT	COM1	COM2
DevSet	89.65	86.19	86.26	89.60	89.65
TestSet	87.05	84.15	84.61	87.21	87.05

Table 1: The LAS score for the MATE, DeSR and MALT parsers, their simple combinations COM1 and COM2 on the development and test sets.

The COM2 algorithm is a simple 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 multiple roots can destroy the *treeness* of the structure. The solution that we adopt in the COM2 algorithm is quite naive: if the tree produced by the COM1 algorithm for a sentence is corrupted, then the COM2 returns the tree produced by the best parser. Again, similarly to (Zeman and Žabokrtskỳ, 2005), who use the name *switching* for COM2, this is the most logical decision when there is an emerging best parser from a development data set.

2.2 Experimental Results

We applied our approach for parsing combination in two stages. In the first stage we use the development set to evaluate the best parser and in the second stage we use the COM1 and COM2 algorithms to parse the test set. For all the experiments we used two machines. A powerful Linux workstation, equipped with 16 cores, processors 2GHz, and 128 GB ram has been used for the training of the MATE and Malt parsers. Morever, we have not been able to install DeSR on this machine, so we use a virtual Linux workstation equipped with a single processor 1GHz, and 2 GB ram has been used DeSR. The MALT and DeSR parsers accept as input the CONLL-07 format, that is the format provided by the task organizers. In contrast, MATE accepts the CONLL-09 format: simple conversions scripts have been implemented to manage this difference.

A first run was performed in order to evaluate the best parser in the COM1 and COM2 algorithms with respect to the LAS. We used the ISDT training ($file\ isdt\_train.conll$, $165,975$ words) as training set and the ISDT development ($file$: $isdt\_devel.conll$, $12,578$ words) as development set. The first row in Table 1 shows the results of the three parsers in this first experiment. MATE parser outperforms the DeSR and MALT parsers of $\sim 3\%$ better. On the basis of this result, we used MATE as our best parser in the combination algorithms (cf. Section 2.1).

COM1 and COM2 reach the score of 89.60% and 89.65% respectively. So, on the development set there is no improvement on the performance of the best parser. The reason of this is evident from table 2, that details the results of the three parsers on the development set on the basis of their agreements. The second row of this table show that when $DeSR == MALT! = MATE$, the combination algorithm gives the *wrong* selection preferring the majority.

In a second run, we used the union of the training and development set as a whole training set ($files$: $isdt\_train.conll, isdt\_devel.conll$) and we used the blind file provided by the organizers as test set ($file$: $DPIE\_Test\_DS\_blind.conll$, $9,442$ words). The second row in Table 1 shows the results of the three parsers in this second experiment: the LAS values 87.21% and 87.05%, produced by COM1 and COM2, are the official results for of our participation to the DPIE task.

There is a $\sim 0.15\%$ difference between the COM1 and COM2 results and in Table 3 we detailed the results of the three parsers on the test set. When the three parsers agree on the same dependency (Table 3, first row), this happens on $\sim 80.27\%$ of the words, they have a very high LAS score, i.e. 94.03%. In contrast to the development set, DeSR and MALT parsers do better than the MATE parser only when they agree on the same dependency (Table 3, second row). The inspection of the other rows in Table 3 shows that COM1 algorithms has the best possible performance w.r.t. the voting strategy. Finally, the fact that COM2 produces the same result of MATE shows that the LAS improvement produces always a non-correct tree in the final output.

In Table 4 we report the results of the system with respect to the measure defined on the propagated and collapsed dependencies. In contrast to the LAS measure, here COM1 produces a worse result than COM2. So, improvements in the LAS

	MATE	DeSR	MALT	COM1	COM2
DevSet	84.8/92.0/88.2	80.7/89.2/84.7	81.0/89/0/84.8	85.2/91.2/88.1	84.8/92.0/88.2
TestSet	80.5/90.0/85.0	76.9/86.7/81.5	76.8/86.6/81.4	80.9/88.0/**84.3**	80.5/90.0/**85.0**

Table 4: The collapsed and propagated dependency sore in terms of precison/recall/F-score for the collapsed dependencies for the three parsers, their simple combinations (COM1 and COM2) on the development and test sets.

					%
MATE 95.4	==	DeSR	==	MALT	**81.8**
MATE 43.5	!=	**DeSR 39.8**	==	**MALT**	**4.9**
MATE	==	**DeSR 70.9**	!=	MALT 13.1	**4.8**
MATE	==	**MALT 70.0**	!=	DeSR 15.6	**5.0**
MATE 46.6	!=	DeSR 10.9	!=	MALT 15.5	**3.6**

Table 2: The detailed performances on the LAS score of the three parsers and their simple combination on the ISDT development set. Note that we are computing the scores with punctuation.

					%
MATE 94.03	==	DeSR	==	MALT	**80.28**
MATE 40.7	!=	**DeSR 41.9**	==	**MALT**	**5.34**
MATE	==	**DeSR 62.2**	!=	MALT 19.4	**5.11**
MATE	==	**MALT 67.4**	!=	DeSR 17.6	**5.25**
MATE 35.9	!=	DeSR 15.9	!=	MALT 17.8	**4.03**

Table 3: The detailed performances on the LAS score of the three parsers and their simple combination on the ISDT test set. Note that we are computing the scores with punctuation.

produces as drawback a decline with respect to this measure.

3 The CLaP task

The Cross-language Dependency Parsing (CLaP) is a pilot task focusing on cross-lingual transfer parsing. In this subtask it is asked to learn from the Italian Stanford Dependency Treebank annotated in with the universal dependencies ($file$: $isdt\_udl.conll$), and to test on sentences of other languages (McDonald et al., 2013). In particular, we decided to participate to the task on four specific languages: German (DE), Espanol (ES), French (FR) and Brazilian Portuguese (PT-BR). For each language, the organizers provided a development file.

In CLaP task we used only one parser, i.e. the MALT parser. We decided to use this parser since there is a related system, called MaltOptimizer (Ballesteros and Nivre, 2012) (version 1.0.3), that allows for a straight optimization of the various parameters of the MALT parser. Indeed, our strategy was to train the MALT parser on the universal isdt by using the specific algorithm and features which optimize the learning on the development set of the target language. Moreover, in order to supply lexical information to the parsing algorithm, we used *Google_translate* (`https://translate.google.com`) to translate foreign words in Italian. In Figure 1 we reported the workflow adopted in this task for learning and parsing of the French language (it is analogous for the other languages). The learning stage is composed by five steps:

1. A script extracts the foreign words from the development set

2. Google_translate translates the foreign words, contained in one single file, into Italian.

3. A script recomposes the development set with Italian words

4. MaltOptimizer uses the recomposed development set in order to produce a configuration file (algorithm and features).

5. The MALT parser uses the configuration file to produce a parsing model file.

In a similar way, the parsing stage is composed by five steps:

1. A script extracts the foreign words from the test set.

2. Google_translate translates the foreign words,

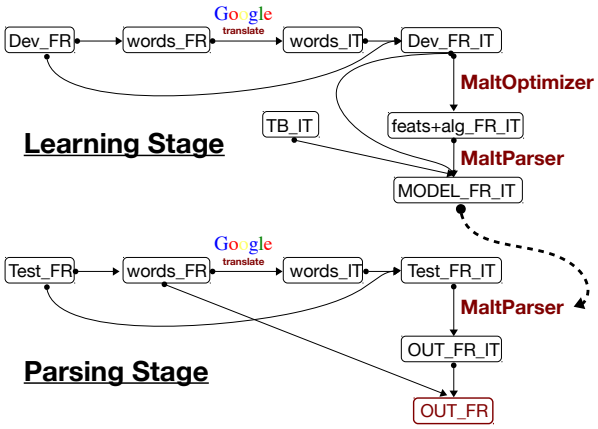

Figure 1: The workflow adopted fot the CLaP task for the French language: the schema is identical for the Spanish, German, Brazilian-Portoguese.

	DE	ES	FR	PT-BR
Baseline 1	60.23	67.72	66.74	66.12
Baseline 2	66.51	71.69	71.60	71.70
System	66.51	72.39	71.53	71.70

Table 5: The LAS score for CLaP task on the test sets for German (DE), Espanol (ES), French (FR), Brazilian-Portoguese (PT-BR) languages.

contained in one single file, into Italian.

3. A script recomposes the test set with Italian words.

4. The MALT parser uses the parsing model to parse the recomposed test set.

5. A script recomposes the parsing test set with the foreign words.

In Table 5 we reported the results in terms of LAS measure of the system together with two baselines. The baseline 1 it has been produced by training the MALT parser with the standard configuration on the learning set obtained by the union of the u-ISDT with the original development set of the foreign language. The baseline 2 it has been produced by training the MALT parser with the standard configuration on the learning set obtained by the union of the u-ISDT with the translated development set of the foreign language. The results proves that our workflow produces an improvement on the LAS measure of $5 - 6\%$ for each language. Comparing the baselines, we can say that the improvements are essentially by the translation process rather than the optimization process.

4 Conclusions

In this paper we described the two systems used by the UniTo group to participate to EVALITA 2014 parsing competition. The first, used in the DPIE task, is a very simple ensamble parsing algorithm; the second is a cross-language parsing algorithm that uses an on-line translator as external knowledge source.

In the DPIE task, we can see that the performance of the ensamble system with respect to the bast parser is quite neglectable, in contrast to the results obtained in other competition (Mazzei and Bosco, 2012). This result suggests that the performance of the simple ensamble algorithms adopted are highly sensitive from the leaning set adopted.

In the CLaP task, we can see that the performance of the developed system outperforms the baseline for all the four languages. This result confirms the possibility to improve parsing performances by using data developed for other languages.

References

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

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. 2012. Maltoptimizer: A system for maltparser optimization. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

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 Alessandro Mazzei. 2012. The evalita dependency parsing task: from 2007 to 2011. In *Evaluation of Natural Language and Speech Tools for Italian - Proceedings of Evalita 2011*, volume 7689, pages 1–12. Springer-Verlag, Heidelberg. ISBN: 978-3-642-35827-2.

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.

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.

Marie-Catherine de Marneffe and Christopher D. Manning, 2008a. *Stanford typed dependencies manual*, September. http://nlp.stanford.edu/software/dependencies_manual.pdf.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008b. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8, 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.

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.

Alessandro Mazzei and Cristina Bosco. 2012. Simple Parser Combination. In *SPLeT 2012 – Fourth Workshop on Semantic Processing of Legal Texts (SPLeT 2012) – First Shared Task on Dependency Parsing of Legal Texts*, pages 57–61.

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.

Ryan T. McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B. Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*, pages 92–97. The Association for Computer Linguistics.

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.

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 Christopher D. 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.

Daniel Zeman and Zdenek 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.