Differentia compositionem facit
A Slower-paced and Reliable Parser for Latin

Edoardo Maria Ponti[‡], **Marco Passarotti**[⋆]
[‡]University of Pavia, [⋆]CIRCSE Research Centre – Università Cattolica del Sacro Cuore
Corso Strada Nuova 65 – 27100 Pavia – Italy, Largo A. Gemelli 1 – 20123 Milan – Italy
edoardomaria.ponti01@universitadipavia.it, marco.passarotti@unicatt.it

Abstract

The *Index Thomisticus* Treebank is the largest available treebank for Latin; it contains Medieval Latin texts by Thomas Aquinas. After experimenting on its data with a number of dependency parsers based on different supervised machine learning techniques, we found that DeSR with a multilayer perceptron algorithm, a right-to-left transition, and a tailor-made feature model is the parser providing the highest accuracy rates. We improved the results further by using a technique that combines the output parses of DeSR with those provided by other parsers, outperforming the previous state of the art in parsing the *Index Thomisticus* Treebank. The key idea behind such improvement is to ensure a sufficient diversity and accuracy of the outputs to be combined; for this reason, we performed an in-depth evaluation of the results provided by the different parsers that we combined. Finally, we assessed that, although the general architecture of the parser is portable to Classical Latin, yet the model trained on Medieval Latin is inadequate for such purpose.

Keywords: Combination, Dependency Parsing, Latin

1 Introduction

Latin language displays a rich fusive morphology. Morphemes may carry several grammatical meanings, syntactic constituents are often discontinuous, and word order is quite free, as it can be determined more on pragmatic grounds than by syntax. Such a free word order status for Latin results in a high number of non-projective pairs of nodes in dependency trees, which in turn affects negatively parsing accuracy (Nivre and Nilsson, 2005). This partly explains why parser performances for Latin lag behind other languages.

Furthermore, Latin suffers from lacking sufficiently large language resources, and in particular treebanks. Also, texts written in Latin are scattered across a wide span of periods (covering more than two millennia), territories and styles. Since no selection of texts can be representative enough of Latin as a whole, any experiment run on textual data in this language is only able to deal with its particular variety represented by the specific data in question. In this respect, Passarotti and Ruffolo (2010) showed that, while testing on Classical Latin a model for syntactic parsing trained on Medieval Latin data, the quality of results dramatically drops.

So far, the highest accuracy rates for dependency parsing of Latin are those reported by Passarotti and Dell'Orletta (2010), who trained a parsing model on the *Index Thomisticus* Treebank[1] (IT-TB; Passarotti (2011)). Indeed, the IT-TB is larger than other comparable resources for Latin, like for instance the Latin Dependency Treebank (LDT) for Classical Latin, developed by Bamman and Crane (2007). Moreover, the texts of the IT-TB are written in quite a formal variety of Medieval Latin. In this variety, non-projectivity rate is milder than in Classical Latin: according to Passarotti and Ruffolo (2010), 3.24% of the total pairs of nodes is non-projective in the IT-TB against 6.65% in the LDT.

[1] `http://itreebank.marginalia.it/`

The parser devised by Passarotti and Dell'Orletta (2010) was based on post-processing techniques of combination and revision (Attardi and Dell'Orletta, 2009; Surdeanu and Manning, 2010). These techniques were applied to a number of outputs achieved by using different settings for the stochastic dependency parser DeSR (Attardi, 2006). Stated in the usual evaluation metrics for dependency parsers, the best results reported were 80.02 in Labeled Attachment Score (LAS) and 85.23 in Unlabeled Attachment Score (UAS) (Buchholz and Marsi, 2006).

In recent years, developments in dependency parsing were geared mostly towards improving speed (Bohnet, 2010), for the benefit of real-time tasks, by focusing on one parser at time. Nevertheless, classicists working in the area of Digital Humanities look more for higher accuracy than speed. Indeed, they are more interested in the linguistic features shown by one specific text or author rather than in using textual evidence to draw general linguistic claims on ancient languages. The parser presented in this paper is designed to fit this peculiar application purpose. As a consequence, the trade-off between accuracy and speed is solved at the expense of the latter, if necessary.

The paper is organised as follows. In Section 2, we present the data. As a preliminary step, in Section 3 we describe the candidate parsers and collect the results provided by each of them. In Section 4, we tune their settings and provide a tailor-made feature model. In Section 5, we apply a post-processing technique of combination to the outputs of the parsers, and show the final results, which are in turn evaluated in-depth in Section 6. Then, we experiment and evaluate the same technique on Classical Latin texts in Section 7. Finally, Section 8 presents our conclusions and the future perspectives for this research.

2 Data

The source for the data that we used in the experiments is the most recent release of the IT-TB (September, 2015).

The treebank was built upon the *Index Thomisticus* corpus, which gathers Thomas Aquinas' *opera omnia* (Busa, 1974 1980). The annotation style of the IT-TB resembles that used for the so-called analytical layer of annotation of the Prague Dependency Treebank for Czech. However, also dedicated guidelines for Latin were settled by Bamman et al. (2007).

As it is a still ongoing project, the IT-TB currently includes more data than the versions of it that were used in previous experiments of dependency parsing for Latin. In more detail, the data made available in the most recent release of the IT-TB encompass the entire books 1 and 2, and part of book 3 of *Summa contra Gentiles*, and the concordances of lemma *forma* ('form') excerpted from *Summa theologiae* and *Scriptum super Sententiis Petri Lombardi*. We split the treebank, labelled T2, into a training set and a test set with a ratio of about 9:1. Furthermore, we collected also the data used by Passarotti and Dell'Orletta (2010), labelled T1, keeping their original split into training and test sets.

	Training		Test	
Version	Nodes	Sentences	Nodes	Sentences
T1	61,024	2,820	7,379	329
T2	236,531	13,770	17,353	1,180

Table 1: Size of the subsets of the treebanks

Table 1 shows the number of nodes and sentences for each subset of the treebanks. The purpose of comparing T1 and T2 is ultimately to evaluate the parsing technique that will be presented in the following sections, by assessing to what extent the improvements that we achieve are due to the larger size of the training set and to the technique itself. Where necessary, a development set with the same size of the respective test data was stored away from the training data.

3 Candidate Parsers

As a preliminary step, we surveyed a number of freely available dependency parsers based on supervised machine learning. This technique allows to build predictors characterized by an algorithm, its parameters and a feature set. Hence, its performances are affected by the choice of the main learning procedure, of its fixed specifications and of what is to be observed in the data. We selected the following parsers and applied them to T2. The accuracy rates of the results on T2 by single parser are displayed in Table 2.

- **Malt**.[2] MaltParser 1.8.1 by Nivre and Nilsson (2005), which employs a shift-reduce algorithm. We optimised the parameters of the parser by using the MaltOptimizer tool (Ballesteros and Nivre, 2012): a Covington non-projective algorithm and a tailor-made feature set were selected.
- **MTGB**.[3] A graph-based parser from the MATE-tools collection (Bohnet, 2010), whose latest version was released as *anna-3.61*. A maximum spanning tree algorithm decomposes the tree structure into second-order factors and combines with a non-projective approximation algorithm and a passive-aggressive perceptron.
- **Joint**. A shift-reduce parser developed by Bohnet et al. (2013) and included in the MATE-tools collection. It performs syntactic and morphological disambiguation in a single step rather than in a pipeline.
- **MST**.[4] MSTParser version 0.2, a graph-based parser relying on a maximum spanning tree algorithm (McDonald and Pereira, 2006).
- **LSTMc**.[5] A transition-based parser that uses long short-term memory (LSTM) recurrent neural networks to learn representations of the parser state, where word representations are replaced with character-based ones (Ballesteros et al., 2015). In our experiments, we preserved the default hyper-parameters and stopped training at the 5500th epoch.
- **DeSR**.[6] DeSR version 1.4.3 (Attardi, 2006), a shift-reduce parser whose basic setting is left-to-right transition with a multilayer perceptron. We used the same feature model described by Passarotti and Dell'Orletta (2010).

Parser	LAS	UAS
Malt	72.98	78.69
MTGB	78.31	85.67
Joint	81.51	86.8
MST	69.36	76.98
LSTMc	69.41	76.91
DeSR	**81.55**	**87.1**

Table 2: Results on T2 by single parser

From Table 2, it emerges that DeSR and Joint parsers outperform the other candidates. Apart from the dedicated feature set, we used DeSR in its basic settings. In spite of this, its LAS and UAS already surpass the best results reported by Passarotti and Dell'Orletta (2010), who used the same feature set. This is due both to the larger size of the training set and to the usage of a multilayer perceptron (MLP) instead of a support vector machine algorithm (SVM). Overall, it should be noted from Table 2 that transition-based or graph-based architectures are not discriminative by themselves as for the reliability of the parser.

4 Tuning the Parameters

In order to exploit the full-fledged potential of parsers, we focused on the best-scoring and most customizable one, namely DeSR, and tried to push further its performance. In particular, we experimented new feature models via manual filtering, and tuned the settings of the parser.

[2] http://www.maltparser.org/.

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

[4] https://sourceforge.net/projects/mstparser/.

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

[6] https://sites.google.com/site/desrparser/.

4.1 Feature Model

We found that the feature model shown in Table 3 allows for a slightly higher accuracy rate compared to that presented in Passarotti and Dell'Orletta (2010) and used in Section 3.

Feature	Tokens
LEMMA	-2 -1 0 1 2 3 *prev*(0) *next*(-1) *leftChild*(-1) *leftChild*(0) *rightChild*(-1) *rightChild*(0)
POSTAG	-2 -1 0 1 2 3 *prev*(0) *next*(-1) *leftChild*(-1)
CPOSTAG	-1 0 1 2
FEATS	-1 0 1 2
DEPREL	*rightChild*(-1)
HEAD	-1 0

Table 3: A new feature model for Latin in DeSR

In Table 3, the first column reports the feature attributes[7]; the second column indicates from which token a value is extracted. In the latter, mere numbers are used as a notation for the position of the token with respect to the next one in the input, numbered 0. Thus, positive integers refer to tokens in the input to be processed, whereas negative integers refer to the tokens on the stack. As for the meaning of the special operators, given a token x, *prev*(x) refers to the token preceding x in linear order, whereas *next*(x) to the one following it. Instead, *leftChild*(x) and *rightChild*(x) concern dependency information and refer respectively to the leftmost and rightmost child of x.

This new model is basically the same as that of Passarotti and Dell'Orletta (2010), with the exception of deleting from the feature POSTAG the tokens *leftChild*(0), *rightChild*(-1) and *rightChild*(0). Simplifying a feature model can reduce overfitting. Furthermore, results allow for a linguistic interpretation, as the PoS of a token (and of its adjacent tokens as well) seems not to be relevant to determine the dependency relation of its head. The accuracy improvement is reported in the first row of Table 4, to be compared with the last one of Table 2. LAS has increased by 0.46 and UAS by 0.25.

4.2 Algorithm and Techniques

Since the new feature model improved the accuracy of the parser, we used it in the following two experiments that we run on T2 data.

First, we reversed the transition direction; secondly, we employed a different algorithm to tune the settings of DeSR as finely as possible. Table 4 shows the results achieved on T2 test data. The first line of Table 4 presents the accuracy rates resulting from the application of the new feature model to the default settings (MLP, left-to-right).

By reversing the direction of the transition during parsing (from left-to-right to right-to-left), we achieved an enhancement of the accuracy rate, as it appears from the second row of Table 4. A tentative explanation for this could be that 125,934 tokens of T2 have the head on their left, whereas 96,812 have the head on their right. For some reason, it seems to be easier for the parser to make the correct decision by encountering a dependent before its head rather than the contrary.

[7]LEMMA: lemma. POSTAG: Fine-grained PoS tag. CPOSTAG: Coarse-grained PoS tag. FEATS: morphological features. DEPREL: dependency relation. HEAD: governor in the tree.

Parser	LAS	UAS
DeSR (left-to-right, MLP)	82.01	87.35
DeSR (right-to-left, MLP)	83.14	88.46
DeSR (right-to-left, SVM)	82.35	87.25

Table 4: Results of tuning the settings of DeSR on T2

Then, we replaced MLP with SVM as learning and parsing algorithm. MLP is a feedforward artificial neural network trained with backpropagation, while SVMs are linear classifiers generalized through a kernel trick. In our data, SVM does not reach an accuracy as high as MLP right-to-left does, as proved by the third row of Table 4. In their experiment, Passarotti and Dell'Orletta (2010) had selected a SVM algorithm: the improvement of the accuracy rates that we observed is partly due to the choice of selecting a MLP instead.

5 Combination and Best Results

After maximizing the individual contribution of DeSR, we applied a post-processing technique of combination to the outputs of a number of parsers selected among those described in Sections 3 and 4.

An ensemble usually outperforms a single predictor on condition that a sufficient accuracy and diversity of its members holds. According to Surdeanu and Manning (2010), the simplest algorithm for combination, based on unweighted voting, is as effective as way more complex algorithms. In such a procedure, given an ordered set of outputs, the value selected by the majority prevails. In case of tie, outputs that rank higher in the order have priority. Linear-time re-parsing algorithms can guarantee well-formed (i.e. non-acyclic) dependency trees. Following this, we selected a combination technique led by unweighted voting.

Although Passarotti and Dell'Orletta (2010) already used a linear tree combination based on a voting scheme, all the predictors were stemming from the DeSR suite, thus allowing for a quite narrow range of diversity. This resulted in a limited span between the best LAS / UAS rates achieved before combination (79.27 / 84.63) and those obtained after combination (80.02 / 85.23). Beside using a new feature model and a richer dataset, the key idea in the present paper is to associate DeSR with different parsers: even though they perform worse separately, they contribute to the improvement of the final accuracy by virtue of their mutual differences. The best accuracy rates resulting from the combinations that we experimented are reported in Table 5.

The scores in Table 5 show that there is a significant gap between the best LAS / UAS before combination (83.14 / 88.46) and those after combination (86.5 / 90.97): compare the second row of Table 4 with the last of Table 5. Accuracy rates allowed by different orders and member sets of the selected combinations are very similar and thus could arise by chance: in Table 5 the difference from C4 is significant for C1 ($p < 0.1$) and C2 ($p < 0.5$), but not for C3.

Predictor Outputs	Label	LAS	UAS
DeSR (r, MLP) + DeSR (r, SVM) + DeSR (l, MLP) + MTGB	C1	85.92	90.58
DeSR (r, MLP) + DeSR (r, SVM) + DeSR (l, MLP) + Joint	C2	86.21	90.66
DeSR (r, MLP) + DeSR (l, MLP) + Joint + MTGB	C3	86.37	90.91
DeSR (r, MLP) + DeSR (r, SVM) + DeSR (l, MLP) + Joint + MTGB	**C4**	**86.5**	**90.97**

Table 5: Results by group of outputs combined

Deprel	Frequency	C4	DeSR (r, MLP)	DeSR (r, SVM)	DeSR (l, MLP)	Joint	MTGB
Adv	9.2%	89.2	**84.3**	83.5	82.5	83.6	77.9
Atr	15.61%	89.2	85.8	**86.9**	84.9	85.6	83
Obj	5.96%	85	81.7	80.5	**83.4**	76.7	68.7
Pnom	4.77%	80.7	74.3	76	74.4	**79**	75.6
Pred	3.71%	99.4	**97.7**	96.4	97	93.2	94.1
Sb	9.85%	86.9	82.4	81.8	81.4	**82.6**	78.1

Table 6: LAS of C4 and its members by selected dependency relations

6 In-depth Evaluation

This section presents an in-depth analysis of the results achieved. Indeed, it is still unclear to what extent the new algorithmic architecture and the enlarged dataset contribute to them. Therefore, in Subsection 6.1 we apply the combined parser labelled C4 in Table 5 to the dataset T1, in order to assess the effect of using this new combination of parsers on the same dataset as that used by Passarotti and Dell'Orletta (2010).

In Subsection 6.2 we evaluate the LAS of each parser involved in C4 by a number of selected dependency relations, in order to highlight the parser-specific performance on them.

6.1 Comparison with the Previous Version

In order to show that the technique presented in this paper supplies by itself a noteworthy addition, we trained and tested on T1 each member of C4 (with DeSR provided with the new feature model) and then we combined their outputs. We obtained an accuracy of 82.91 LAS and 88.22 UAS. These results outperform those reported by Passarotti and Dell'Orletta (2010), gaining 2.89 in LAS and 2.99 in UAS. Such an improvement, achieved on the same training and test sets used by the previously best performing parser on the IT-TB data, is due only to the different technique that we used, while the size of the datasets does not play any role here. Indeed, combining different (kinds of) parsers allows for better results than dealing with just one.

6.2 Individual Contribution of Parsers

In order to understand the single contribution given by the parsers of C4 to the combined results, we evaluated the LAS provided by each of them for a set of dependency relations that we selected as the most frequently occurring in T2. Beside the parser-specific LAS by dependency relation, we also calculated that allowed by the C4 combined parser. Table 6 shows the results.

Interestingly enough, no single parser in Table 6 outperforms the others by achieving the highest LAS for all the selected dependency relations. On the contrary, each parser seems to be best performing on specific relations. In more detail, the right-to-left versions of DeSR show the best rates for adverbials (Adv; MLP) and attributes (Atr; SVM). Instead, left-to-right parsers allow for the highest accuracy on subjects (Sb; Joint), nominal predicates (Pnom; Joint) and direct/indirect objects (Obj; MLP). Furthermore, DeSR shows the tendency to guess reliably predicates (Pred), regardless the direction of parsing.

Although they are not among the most frequent dependency relations in T2 (and, thus, not reported in Table 6), it is worth noticing that MTGB outperforms the other parsers for sentence adverbials (dependency relation: AuxY) and verbal attributes not participating in verb government (Atv and AtvV). Another well-known tricky issue in dependency trees is coordination. In this respect, the accuracy rates for coordinated dependency relations provided by the single parsers of C4 reflect those for the corresponding non-coordinated ones. DeSR (r, MLP) outperforms the other parsers for both coordinated adverbials and predicates, just like it does for the non-coordinated ones. DeSR (r, SVM) is the best parser for coordinated attributes; DeSR (l, MLP) achieves the highest rates for coordinated direct/indirect objects, and Joint is best for both coordinated subjects and nominal predicates.

Such a widely distributed contribution carried by the single parsers helps to explain why combination is so effective. Consider for example Figure 1, which we built by using the MaltEval tool (Nilsson and Nivre, 2008). Figure 1 shows the parser-specific dependency analyses for the sentence *Felicitas est proprium hominis bonum* ("Happiness is a proper good of man"; *Summa contra Gentiles*, book 3, chapter 34, number 5). From top to bottom in the figure, the analyses are those predicted by C4, DeSR (r, MLP), DeSR (r, SVM), DeSR (l, MLP), Joint and MTGB respectively. Correct predictions are coloured in green; wrong predictions appear in red.

Although DeSR (r, MLP) is overall the best-performing parser, it is weak in recognizing nominal predicates. Instead, DeSR (r, SVM), DeSR (l, MLP), Joint and MTGB are more reliable than DeSR (r, MLP) for this specific de-

pendency relation. Indeed, their (common) prediction for the Pnom relation in Figure 1 is correct, while the prediction made by DeSR (r, MLP) is wrong. Since the second group makes up the majority of the parsers, the prediction of DeSR (r, MLP) is overridden and C4 displays the correct analysis.

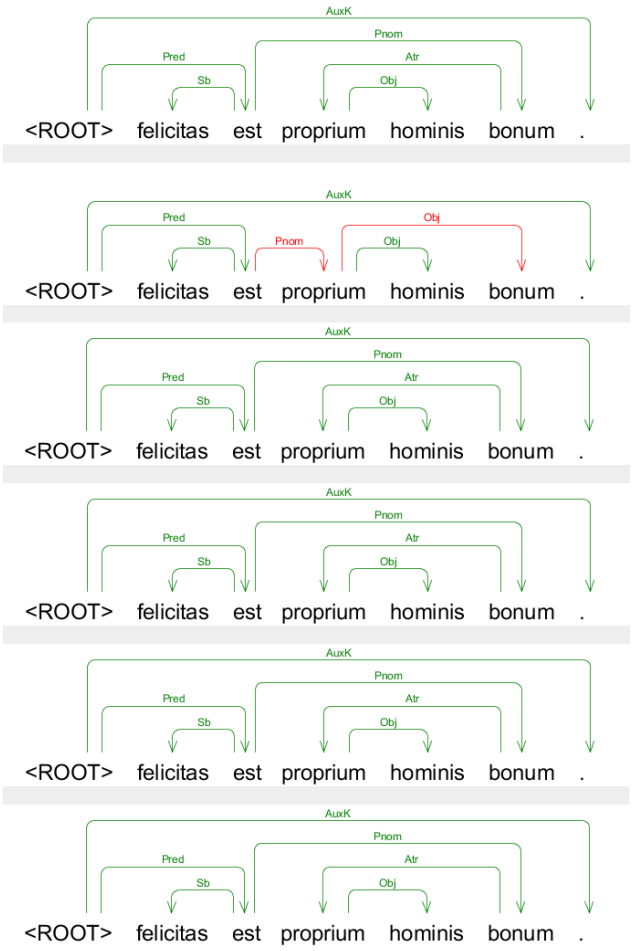

Figure 1: Comparing parser-specific predictions

7 Classical Latin

As mentioned in the Introduction, Latin has several varieties, which can differ even heavily one from the other. Since Classical Latin is considered to be the 'standard' variety of Latin, we applied the C4 combined parser, which was trained on the Medieval Latin data of T2, also on the texts of the LDT. This treebank consists of 53,143 nodes and it features texts ranging from the 1st century B.C. to the 5th century A.D. The LDT texts differ from those of the IT-TB not only on the diachronic axis, but also on the stylistic one. Table 7 displays the results by author and genre.

What stands out clearly from Table 7 is the collapse of the parser performance on all the authors represented in the LDT compared to the results achieved on Thomas Aquinas' texts. This derives from the remarkable incongruity holding between the varieties of Latin represented in the training set (IT-TB) and in the test data (LDT).

We compared our results with those provided by Passarotti and Ruffolo (2010). They report on an experiment where

Author	Genre	LAS	UAS
Caesar	prose	28.2	42.1
Cicero	prose	27.6	41,1
Jerome	prose	35.8	51.2
Ovid	poetry	23.9	37.7
Petronius	prose	35.9	51.3
Propertius	poetry	25.8	40.5
Sallust	prose	27.6	41.3
Vergil	poetry	24.9	39.6

Table 7: Accuracy of C4 on Classical Latin texts

DeSR (provided with the standard feature model for Italian) is trained on a set of 44,195 nodes from the IT-TB and then it is tested on the LDT. Their results are for each author of the LDT more than ten points lower than those shown in Table 7 (both by LAS and UAS). This demonstrates that, although applying to Classical Latin a dependency parser trained on Medieval Latin does not allow for sufficiently reliable accuracy rates, yet enlarging the size of the training set (and changing the parsing technique as well) results in 'less low' performances.

The authors on which C4 performs best are Petronius and Jerome. The former wrote the excerpt known as *Coena Trimachionis* in his *Satyricon* by imitating the spoken non-literary language. The latter translated the *Bible* across the 4th and the 5th century. Instead, the worst results are for poetry texts by Ovid, Propertius and Virgil (respectively *Metamorphoses*, *Elegies*, and *Aeneid*). This trend emerges also from the experiment of Passarotti and Ruffolo (2010), where the best and the worst results are shown by exactly the same two groups of texts.

8 Conclusion and Future Work

In this paper, we presented a number of experiments on dependency parsing of Medieval Latin. We trained and tested six parsers based on different supervised machine learning techniques, using the *Index Thomisticus* Treebank for training.

After finding that DeSR outperforms the other candidate parsers, we provided it with a new feature model for Medieval Latin and tuned its settings. A multilayer perceptron algorithm with a right-to-left direction of transition resulted to be the best configuration. Finally, we experimented several combinations of the outputs of parsers, resulting in a pipeline of analysis that gains 6.48% in LAS and 5.74% in UAS with respect to the state of the art.

An in-depth evaluation of the results demonstrated that, beside the larger size of the traning set, the new parser architecture that we built is indeed responsible for outperforming the best accuracy rates previously reported for parsing Medieval Latin. Moreover, the evaluation showed the parser-specific weaknesses and the strengths in assigning dependency relations. We found that results are grounded on a sufficient range of diversity, which supports our choice of combining the outputs of the parsers. Finally, we applied our ensemble parser to texts written in Classical Latin, showing that the parser is not portable over different varieties of Latin because of their too wide diachronic and

stylistic incongruity. Given the same pipeline architecture that we built, we assume that an ad-hoc parsing model trained on textual data of Classical Latin could provide satisfactory results.

Future improvements are linked both to a further enlargement and wider diachronic coverage of the training set and to testing deep learning techniques (Collobert, 2011). A more thorough investigation of the hyper-parameter setting of the neural network we employed in our paper, as well as the use of other artificial neural network architectures (Dyer et al., 2015; Weiss et al., 2015) might improve the results.

Exploiting various pre- and/or post-processing techniques to hybridize a supervised machine learning approach with a rule-based one is a promising and challenging line of research, which we should pursue in the near future. For example, Jelínek (2014) and Simi et al. (2014) take advantage on automatically simplifying the annotation in the training set before feeding the machine, to restore the original labels before building the output. Finally, other language resources for Medieval Latin could provide additional linguistic knowledge to guide the parser, like for instance the syntactic-based subcategorization lexicon IT-VaLex (McGillivray and Passarotti, 2009).

Acknowledgements

We would like to thank Felice Dell'Orletta and Bernd Bohnet for their suggestions on DeSR and MATE-tools parsers, respectively. Many thanks also to Marco Piastra for providing free access to the facilities of the Artificial Vision Laboratory (University of Pavia, Italy).

9 Bibliographical References

Attardi, G. and Dell'Orletta, F. (2009). Reverse Revision and Linear Tree Combination for Dependency Parsing. In Colorado Boulder, editor, *Proceedings of NAACL 2009*, pages 261–264.

Attardi, G. (2006). Experiments with a Multilanguage Non-Projective Dependency Parser. In *Proceedings of CoNLL-X*, pages 166–170.

Ballesteros, M. and Nivre, J. (2012). Maltoptimizer: A system for maltparser optimization. In *Proceedings of LREC'12*, pages 2757–2763.

Ballesteros, M., Dyer, C., and Smith, N. (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.

Bamman, D. and Crane, G. (2007). The Latin Dependency Treebank in a cultural heritage digital library. In *Proceedings of LaTeCH 2007*, pages 33–40, Prague, Czech Republic.

Bamman, D., Crane, G., Passarotti, M., and Raynaud, S. (2007). Guidelines for the Syntactic Annotation of Latin Treebanks. Technical report, Tufts Digital Library, Boston.

Bohnet, B., Nivre, J., Boguslavsky, I., Farkas, R., Ginter, F., and Hajič, J. (2013). Joint morphological and syntactic analysis for richly inflected languages. *Transactions of ACL*, 1:415–428.

Bohnet, B. (2010). Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 88–97.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*, pages 149–164.

Busa, R. (1974-1980). *Index Thomisticus*. Frommann-Holzboog, Stuttgart-Bad Cannstatt.

Collobert, R. (2011). Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Jelínek, T. (2014). Improvements to Dependency Parsing Using Automatic Simplification of Data. In *Proceedings of LREC 2014*.

McDonald, R. and Pereira, F. (2006). Online Learning of Approximate Dependency Parsing Algorithms. In *Proceedings of EACL 2006*, pages 81–88.

McGillivray, B. and Passarotti, M. (2009). The development of the *Index Thomisticus* Treebank valency lexicon. In *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education*.

Nilsson, J. and Nivre, J. (2008). Malteval: an evaluation and visualization tool for dependency parsing. In *Proceedings of LREC'08*.

Nivre, J. and Nilsson, J. (2005). Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of ACL*, pages 99–106.

Passarotti, M. and Dell'Orletta, F. (2010). Improvements in Parsing the *Index Thomisticus* Treebank. Revision, Combination and a Feature Model for Medieval Latin. In *Proceedings of LREC 2010*, La Valletta, Malta.

Passarotti, M. and Ruffolo, P. (2010). Parsing the *Index Thomisticus* Treebank. Some preliminary results. In *15th International Colloquium on Latin Linguistics*, pages 714–725. Innsbrucker Beiträge zur Sprachwissenschaft.

Passarotti, M. (2011). Language Resources. The State of the Art of Latin and the *Index Thomisticus* Treebank Project. In Ortola Marie-Sol, editor, *Corpus anciens et Bases de données, «ALIENTO. Échanges sapientiels en Méditerranée» n. 2*, pages 301–320. Presses universitaires de Nancy.

Simi, M., Bosco, C., and Montemagni, S. (2014). Less is more? Towards a reduced inventory of categories for training a parser for the Italian Stanford dependencies. In *Proceedings of LREC 2014)*.

Surdeanu, M. and Manning, C. D. (2010). Ensemble models for dependency parsing: cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the NAACL*, pages 649–652.

Weiss, D., Alberti, C., Collins, M., and Petrov, S. (2015). Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158*.