Tree Kernels-based Discriminative Reranker for Italian Constituency Parsers

Antonio Uva[†] and **Alessandro Moschitti**
[†]DISI, University of Trento, 38123 Povo (TN), Italy
Qatar Computing Research Institute, HBKU, 5825 Doha, Qatar
antonio.uva@unitn.it amoschitti@gmail.com

Abstract

English. This paper aims at filling the gap between the accuracy of Italian and English constituency parsing: firstly, we adapt the Bllip parser, i.e., the most accurate constituency parser for English, also known as Charniak parser, for Italian and trained it on the Turin University Treebank (TUT). Secondly, we design a parse reranker based on Support Vector Machines using tree kernels, where the latter can effectively generalize syntactic patterns, requiring little training data for training the model. We show that our approach outperforms the state of the art achieved by the Berkeley parser, improving it from 84.54 to 86.81 in labeled F1.

Italiano. *Questo paper mira a colmare il gap di accuratezza tra il constituency parsing dell'Italiano e quello Inglese: come primo miglioramento, abbiamo adattato il parser a costituenti per l'Inglese, Bllip, anche noto come Charniak parser, per l'Italiano e lo abbiamo addestrato sul Turin University Treebank. In seguito, abbiamo progettato un reranker basato sulle Macchine a Vettori di Supporto che usano kernel arborei, i quali possono efficacemente generalizzare pattern sintattici, richiedendo pochi dati di training per addestrare il modello. Il nostro approccio supera lo stato dell'arte ottenuto con il Berkeley parser, migliorando la labeled F1 da 84.54 a 86.81.*

1 Introduction

Constituency Syntactic parsing is one of the most important research lines in Computational Linguistics. Consequently, a large body of work has been also devoted to its design for Italian language (Bosco et al., 2007; Bosco et al., 2009; Bosco and Mazzei, 2011). However, the accuracy reported for the best parser is still far behind the state of the art of other languages, e.g., English.

One noticeable attempt to fill this technological gap was carried out in the EvalIta challenge, which proposed a parsing track on both dependency and constituency parsing for Italian. Among the several participant systems, the Berkeley parser (Petrov and Klein, 2007) gave the best result (Lavelli and Corazza, 2009; Lavelli, 2011).

At the beginning, the outcome for constituency parsing computed on TUT (Bosco et al., 2009) was much lower than the one obtained for English on the Penn Treebank (Marcus et al., 1993). In the last EvalIta edition, such gap diminished as the Italian parser labeled *F1* increased from 78.73% (EvalIta 2009) to 82.96% (EvalIta 2011). Some years later the parser F1 improved to 83.27% (Bosco et al., 2013). However, the performance of the best English parser (McClosky et al., 2006), i.e., 92.1%, is still far away. The main reason for such gap is the difference in the amount of training data available for Italian compared to English. In fact, while Penn Treebank contains $49,191$ sentences/trees, TUT only contains $3,542$ sentences/trees.

In presence of scarcity of training data, a general solution for increasing the accuracy of a machine learning-based system is the use of more general features. This way, the probability of matching training and testing instance representations is larger, allowing the learning process to find more accurate optima. In case of syntactic parsing, we need to generalize either lexical or syntactic features, or possibly both. However, modeling such generalization in state-of-the-art parser algorithms such as the Bllip[1] (Charniak, 2000; Charniak and Johnson, 2005) is rather challenging. In particular, the space of all possible syntactic patterns is very large and cannot be explicitly coded

[1] https://github.com/BLLIP/bllip-parser

in the model. An easier solution consists in using such features in a simpler model, which can be trained to improve the outcome of the main parser, e.g., selecting one of its best hypotheses. In particular, tree kernels (TKs) by Moschitti (2006) can be used for encoding an exponential number of syntactic patterns in parse rerankers.

In this work, we aim at filling the gap between English and Italian constituency parsing: firstly, we adapted Bllip parser, i.e., the most accurate constituency parser for English, also known as Charniak parser, for Italian and trained it on TUT. We designed various configuration files for defining specific labels for TUT by also defining their type, although we did not encode head-finding rules for Italian, needed to complete the parser adaptation.

Secondly, we apply rerankers based on Support Vector Machines (SVMs) using TKs to the *k*-best parses produced by Bllip, with the aim of selecting its best hypotheses. TKs allow us to represent data using the entire space of subtrees, which correspond to syntactic patterns of different level of generality. This representation enables the training of the reranker with little data. Finally, we tested our models on TUT, following the EvalIta setting and compare with other parsers. For example, we observed an improvement of about 2%, over the Berkeley parser, i.e., 86.81 vs. 84.54.

2 Bllip parser

The Bllip parser is a lexicalized probabilistic constituency parser. It can be considered a smoothed PCFG, whose non-terminals encode a wide variety of manually chosen conditioning information, such as heads, governors, etc. Such information is used to derive probability distributions, which, in turn, are utilized for computing the likelihood of constituency trees being generated. As described by McClosky et al. (2006), Bllip uses five distributions, i.e., the probabilities of the (i) constituent heads, (ii) constituent part-of-speeches (PoS), (iii) head-constituents, (iv) left-of-head and (v) right-of-head constituents. Each probability distribution is conditioned by five or more features and backed-off by the probability of lower-order models in case of rare feature configurations. The variety of information needed by Bllip to work properly makes its configuration much harder than for other parsers, e.g., the Berkeley' one. However, Bllip is faster to train than other off-the-shelf parsers.

2.1 Adapting Bllip to Italian Language

Bllip adaptation required to create various configuration files. For example, PoS and bracket labels observed in training and development sets must be defined in a file named *terms.txt*. As labels present in the TUT are different from those of the Penn Treebank[2], we added them in such file. Then, we specified the type of labels present in the data, i.e., constituent type, open-class PoS, punctuation, etc.

Finally, it should be noted that, since Bllip is lexicalized, head-finding rules for Italian should be specified in the file, *headInfo.txt*. For example, the rule, $ADJP \xrightarrow{r} JJ$, specifies that the head of an adjective phrase (ADJP) is the right-most adjective (JJ). Due to time restriction, we used the default Bllip rules and leave this task as our short-term future work.

3 Tree Kernel-based Reranker

We describe three types of TKs and the Preference Reranker approach using them.

3.1 Tree kernels

TKs can be used for representing arbitrary tree structures in kernel machines, e.g., SVMs. They are a viable alternative to explicit feature design as they implement the scalar products between feature vectors as a similarity between two trees. Such scalar product is computed using efficient algorithms and it is basically equal to the number of the common subparts of the two trees.

Syntactic Tree Kernels (STK) count the number of common tree fragments, where the latter (i) contain more than two nodes and (ii) each node is connected to either all or none of its children. We also used a variant, called STK_b, which adds the number of common leaves of the comparing trees in the final subpart count.

Partial Tree Kernels (PTK) counts a larger class of tree fragments, i.e., any subset of nodes, where the latter are connected in the original trees: clearly, PTK is a generalization of STK.

3.2 Preference Reranker

Preference reranking is cast as a binary classification problem, where each instance is a pair $\langle h_i, h_j \rangle$ of tree hypotheses and the classifier decides if h_i is better than h_j. The positive training examples are the pairs, $\langle h_1, h_i \rangle$, where h_1 has the highest *F1* with respect to the gold standard among the candidate hypotheses. The negative examples are

[2]For example, the PoS-tag NN in Penn Treebank corresponds to tag NOU∼CS in TUT

Models	sentences $\leq$ 40 words				All sentences			
	LR	LP	LF	EMR	LR	LP	LF	EMR
Berkeley (Bosco et al., 2013)	83.45	84.48	83.96	24.91	82.78	83.76	83.27	23.67
Berkeley (our model)	85.31	85.76	85.53	27.76	84.35	84.72	84.54	26.33
Bllip base model	85.90	86.67	86.28	29.54	85.26	85.97	85.61	28.00
STK	86.16	87.02	86.59	30.96	85.73	86.38	86.05	29.33
STK_b	86.36	87.21	86.78	**31.67**	85.89	86.53	86.21	**30.00**
PTK	**86.82**	**87.95**	**87.38**	30.96	**86.33**	**87.29**	**86.81**	29.67

Table 1: Comparative results on the test set. **LR/LP/LF** = labeled recall/precision/$F1$. **EMR** = percentage of sentences where recall and precision are 100%. STK- and STK_b-based rerankers use 20-best hypotheses, while PTK-based reranker use 30-best hypotheses.

obtained inverting the hypotheses in the pairs, i.e., $\langle h_i, h_1 \rangle$. If the hypotheses have the same score, the pair is not included in the training set. At classification time all pairs $\langle h_i, h_j \rangle$ generated from the *k*-best hypotheses are classified. A positive classification is a vote for h_i, whereas a negative classification is a vote for h_j. The hypothesis associated with the highest number of votes (or highest sum of classifier scores) is selected as the best parse.

4 Experiments

In these experiments, we first report on the performance of Bllip for Italian and compare it with the Berkeley parser. Then, we show that our parse reranker can be very effective, even in case of use of small training data.

4.1 Experimental Setup

Parsing data. The data for training and testing the constituency parsers come from the TUT project [3], developed at the University of Turin. There have been several releases of the dataset: we used the latest version from EvalIta 2011. The training set is composed of $3,542$ sentences, while the test set contains 300 sentences. The set of PoS-tags includes 97 tags: 68 encoding morphological features (out of which 19 basic tags) for pre-terminal symbols (e.g., ADJ, ADVB, NOUN, etc.) and 29 non-terminal symbols for phrase constituents (e.g., ADJP, ADVP, NP, etc.).

Reranking Data. To generate the data for training the reranker, we apply 10-fold cross validation to the official TUT training set: we train the based parser on 9 folds and applied it to the remaining fold to generate the n-best trees for each of its sentences. Then, we merge all the 10-labeled folds to produce the training set of the reranker. This way, we avoid the bias a parser would have if applied to the data used for training it. For generating the test data of the reranker, we simply apply the base parser (trained on all TUT training data) to the TUT test set and generate n-hypotheses for each sentence.

SVM Reranker. We train the reranker using SVM-light-TK, which takes both feature vectors and trees as input to learn a classification model. The features used for reranking constituency trees are: (i) the probability and the (inverse) rank of the hypotheses provided by Bllip and (ii) the entire syntactic trees used with two types of kernels, STK and PTK, described in Sec. 3.

Measures. For evaluating the parsers, we used the EVALB scoring program, which reports the Labeled Precision (LP), Labeled Recall (LR), Labeled *F1* (LF) and Exact Match Rate (EMR). According to the official EvalIta procedure for evaluating the participant system output, we did not score the TOP label, ignore all functional labels attached to non-terminals and include punctuation in the scoring procedure.

4.2 Bllip base parser results

We divided the training set in train and validation sets, where the latter is composed of the last 50 sentences of each of the six sections of the former for a total of 300 sentences. We train the models on the training set and tune parameters on the validation set. Then, we applied the learned model to the 300 sentences of the test set. Table 1 shows the results obtained by the Bllip base parser on the TUT test set. Our parser obtained an LF of 86.28% for sentences with less than 40 words and a score of 85.61% for all sentences.

4.3 Comparison with the Berkeley parser

Table 1 also reports the results of the Berkeley parser obtained by Bosco et al. (2013). For comparison purposes, we trained our own version of the Berkeley parser. In particular, we trained the parser for 5 split-merge cycles on the whole training set. We selected such number of cycles applying 10-fold cross validation on the training set. Similarly to Bosco et al. (2013), we specialized

[3] http://www.di.unito.it/~tutreeb/

Models	10-best				20-best				30-best			
	Tree		Tree + feat.		Tree		Tree + feat.		Tree		Tree + feat.	
	len≤40	All	≤40	All	len≤40	All	len≤40	All	len≤40	All	len≤40	All
Bllip base model	**86.28**	**85.61**	86.28	85.61	**86.28**	85.61	86.28	85.61	86.28	85.61	86.28	85.61
STK	84.95	84.49	86.31	85.69	84.70	84.16	86.59	86.05	84.99	84.45	86.55	86.00
STK_b	85.05	84.52	86.31	85.69	84.92	84.35	86.78	86.21	84.92	84.38	86.62	86.06
PTK	86.02	85.46	**87.34**	**86.65**	85.89	**86.41**	**87.37**	**86.79**	**86.42**	**85.92**	**87.38**	**86.81**

Table 2: Reranker performance: the first row reports the number n of the best hypotheses used during training. The second row shows the used group of features: Tree or Tree + feat, while the third row illustrates the parse results (LF) for two sentence groups: sentences with ≤ 40 words and all sentences.

punctuation symbols to more specific tags. However, we used full PoS-tags, as they gave the best results in cross-fold validation. Indeed, the table shows that our own version of the Berkeley parser outperforms the version the one of Bosco et al. (2013) by 1.27 absolute percent points (84.54 vs. 83.27). The table also reports the results of the Bllip parser, which outperforms the best result obtained by the Berkeley parser by 1.07% in LF, i.e., 85.61 vs. 84.54.

4.4 Reranking using different TKs

Table 2 reports the LF obtained by different reranking models, varying: (i) the type of TKs, (ii) the group of features (i.e., either trees or trees + feat.) and (iii) the number, n, of parse trees used to generate the reranker training data. More in particular, we experimented with three values for n, i.e., 10-, 20- and 30-best parse trees. As it can be seen from the table, PTK constantly outperforms STK and STK_b for any number of parse hypotheses. This indicates that the subtree features generated by PTK, which include nodes with any subset of the children in the original tree, are useful for improving the parser accuracy.

Very interestingly, the performance of all models when trained on 30-best trees give either worse results (e.g., STK_b and STK) or very little improvement (e.g., PTK) than training on 20-best parse trees. This may suggest that adding too many negative examples, largely populating the lower part of the n-best list may be detrimental.

The bottom part of Table 1 shows standard parser evaluation metrics for different reranking models using different kernel types: only the kernel models with the highest LF from Table 2 are reported. The method shows an 1.2% absolute improvement in LF (from 85.61% to 86.81%) on all the sentences over the base-parser model (i.e., the baseline) when using the most powerful kernel, PTK, and 30-best hypotheses. STK and STK_b show a lower improvement over the baseline of 0.44% and 0.6%, respectively. One interesting fact is the following: while PTK gives better results in terms of LF, STK and STK_b perform better in terms of EMR, i.e., the percentage of sentence parse completely matching gold trees. This is rather intuitive as the name suggests, Partial Tree Kernel generates partial subtrees, i.e., partial production rules as patterns. On one hand, this can improve the ability of matching syntactic patterns, thus capturing rules partially expressed by more than one support vector. On the other hand, the precision in capturing complete patterns, i.e., regarding a complete tree is intuitively decreased.

5 Related Work and Conclusions

This work was inspired by Collins and Duffy (2002) and Collins and Koo (2005), who explored discriminative approaches for ranking problems. Their studies were limited to WSJ, though, and did not explore the use of max-margin classifiers, i.e., SVMs. The first experiments with SVMs and TKs were conducted by Shen and Joshi (2003), who proposed a new SVM-based voting algorithm making use of preference reranking.

In this paper, we adapted the Charniak parser for Italian gaining an improvement of 1.07% over the Berkeley model (indicated by EvalIta as the state of the art for Italian). Then, our TK-based reranker further improved it up to 2 absolute percent points. It should also be noted that our best reranking result is 3.54 absolute points better than the best outcome reported in (Bosco et al., 2013), i.e., 83.27.

In the future, we would like to integrate (i) the features developed in the reranking software available by Johnson and Ural (2010) in our model for further improving it, (ii) generalizing lexical features (e.g., embeddings, brown clusters) and including similarity measures in PTK, i.e., SPTK (Croce et al., 2011).

Acknowledgments

A special thank is due to Alberto Lavelli and Alessandro Mazzei for enabling us to carry out an exact comparison with their parser.

References

[Bosco and Mazzei2011] Cristina Bosco and Alessandro Mazzei. 2011. The evalita 2011 parsing task: the constituency track. *Working Notes of EVALITA*.

[Bosco et al.2007] Cristina Bosco, Alessandro Mazzei, and Vincenzo Lombardo. 2007. Evalita parsing task: an analysis of the first parsing system contest for italian. *Intelligenza artificiale*, 12:30–33.

[Bosco et al.2009] Cristina Bosco, Alessandro Mazzei, and Vincenzo Lombardo. 2009. Evalita09 parsing task: constituency parsers and the penn format for italian. *Proceedings of EVALITA*, 9:1794–1801.

[Bosco et al.2013] Cristina Bosco, Alessandro Mazzei, and Alberto Lavelli. 2013. Looking back to the evalita constituency parsing task: 2007-2011. In *Evaluation of Natural Language and Speech Tools for Italian*, pages 46–57. Springer.

[Charniak and Johnson2005] Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

[Charniak2000] Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

[Collins and Duffy2002] Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 263–270. Association for Computational Linguistics.

[Collins and Koo2005] Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

[Croce et al.2011] Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046. Association for Computational Linguistics.

[Johnson and Ural2010] Mark Johnson and Ahmet Engin Ural. 2010. Reranking the berkeley and brown parsers. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 665–668. Association for Computational Linguistics.

[Lavelli and Corazza2009] Alberto Lavelli and Anna Corazza. 2009. The berkeley parser at the evalita 2009 constituency parsing task. In *EVALITA 2009 Workshop on Evaluation of NLP Tools for Italian*.

[Lavelli2011] Alberto Lavelli. 2011. The berkeley parser at the evalita 2011 constituency parsing task. In *Working Notes of EVALITA*.

[Marcus et al.1993] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

[McClosky et al.2006] David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics.

[Moschitti2006] Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning*, pages 318–329. Springer.

[Petrov and Klein2007] Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*, volume 7, pages 404–411.

[Shen and Joshi2003] Libin Shen and Aravind K Joshi. 2003. An svm based voting algorithm with application to parse reranking. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 9–16. Association for Computational Linguistics.