An Improved Oracle for Dependency Parsing with Online Reordering

Joakim Nivre[⋆†] **Marco Kuhlmann**[⋆] **Johan Hall**[⋆]
[⋆]Uppsala University, Department of Linguistics and Philology, SE-75126 Uppsala
[†]Växjö University, School of Mathematics and Systems Engineering, SE-35195 Växjö
E-mail: FIRSTNAME.LASTNAME@lingfil.uu.se

Abstract

We present an improved training strategy for dependency parsers that use online reordering to handle non-projective trees. The new strategy improves both efficiency and accuracy by reducing the number of swap operations performed on non-projective trees by up to 80%. We present state-of-the-art results for five languages with the best ever reported results for Czech.

1 Introduction

Recent work on dependency parsing has resulted in considerable progress with respect to both accuracy and efficiency, not least in the treatment of discontinuous syntactic constructions, usually modeled by *non-projective* dependency trees. While non-projective dependency relations tend to be rare in most languages (Kuhlmann and Nivre, 2006), it is not uncommon that up to 25% of the sentences have a structure that involves at least one non-projective relation, a relation that may be crucial for an adequate analysis of predicate-argument structure. This makes the treatment of non-projectivity central for accurate dependency parsing.

Unfortunately, parsing with unrestricted non-projective structures is a hard problem, for which exact inference is not possible in polynomial time except under drastic independence assumptions (McDonald and Satta, 2007), and most data-driven parsers therefore use approximate methods (Nivre et al., 2006; McDonald et al., 2006). One recently explored approach is to perform online reordering by swapping adjacent words of the input sentence while building the dependency structure. Using this technique, the system of Nivre (2009) processes unrestricted non-projective structures with state-of-the-art accuracy in observed linear time.

The normal procedure for training a transition-based parser is to use an oracle that predicts an optimal transition sequence for every dependency tree in the training set, and then approximate this oracle by a classifier. In this paper, we show that the oracle used for training by Nivre (2009) is sub-optimal because it eagerly swaps words as early as possible and therefore makes a large number of unnecessary transitions, which potentially affects both efficiency and accuracy. We propose an alternative oracle that reduces the number of transitions by building larger structures before swapping, but still handles arbitrary non-projective structures.

2 Background

The fundamental reason why sentences with non-projective dependency trees are hard to parse is that they contain dependencies between non-adjacent substructures. The basic idea in online reordering is to allow the parser to swap input words so that all dependency arcs can be constructed between adjacent subtrees. This idea is implemented in the transition system proposed by Nivre (2009). The first three transitions of this system (LEFT-ARC, RIGHT-ARC, and SHIFT) are familiar from many systems for transition-based dependency parsing (Nivre, 2008). The only novelty is the SWAP transition, which permutes two nodes by moving the second-topmost node from the stack back to the input buffer while leaving the top node on the stack.

To understand how we can parse sentences with non-projective dependency trees, in spite of the fact that dependencies can only be added between nodes that are adjacent on the stack, note that, for any sentence x with dependency tree G, there is always some permutation x' of x such that G is projective with respect to x'. There may be more than one such permutation, but Nivre (2009) defines the canonical *projective* order $<_G$ for x given G as the order given by an inorder traversal of G that respects the order $<$ between a node and its direct dependents. This is illustrated in Figure 1, where the words of a sentence with a non-projective tree

Proceedings of the 11th International Conference on Parsing Technologies (IWPT), pages 73–76,
Paris, October 2009.

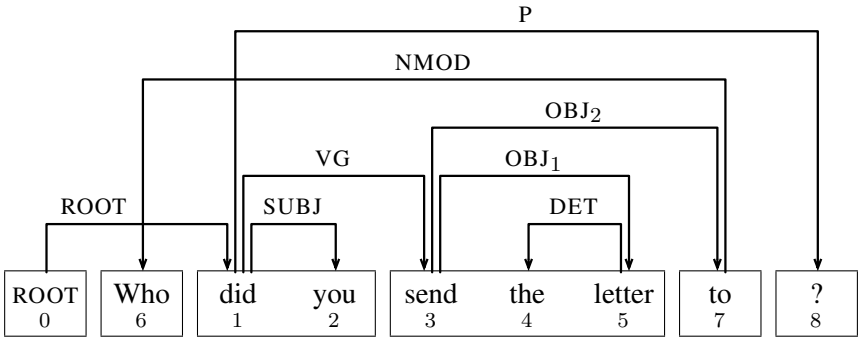

Figure 1: Dependency tree for an English sentence with projective order annotation.

have been annotated with their positions in the projective order; reading the words in this order gives the permuted string *Did you send the letter who to?*

3 Training Oracles

In order to train classifiers for transition-based parsing, we need a *training oracle*, that is, a function that maps every dependency tree T in the training set to a transition sequence that derives T. While every complete transition sequence determines a unique dependency tree, the inverse does not necessarily hold. This also means that it may be possible to construct different training oracles. For simple systems that are restricted to projective dependency trees, such differences are usually trivial, but for a system that allows online reordering there may be genuine differences that can affect both the efficiency and accuracy of the resulting parsers.

3.1 The Old Oracle

Figure 2 defines the original training oracle τ_1 proposed by Nivre (2009). This oracle follows an *eager* reordering strategy; it predicts SWAP in every configuration where this is possible. The basic insight in this paper is that, by postponing swaps and building as much of the tree structure as possible before swapping, we can significantly decrease the length of the transition sequence for a given sentence and tree. This may benefit the *efficiency* of the parser trained using the oracle, as each transition takes a certain time to predict and to execute. Longer transition sequences may also be harder to learn than shorter ones, which potentially affects the *accuracy* of the parser.

3.2 A New Oracle

While it is desirable to delay a SWAP transition for as long as possible, it is not trivial to find the right time point to actually do the swap. To see this, consider the dependency tree in Figure 1. In a parse of this tree, the first configuration in which swapping is possible is when who_6 and did_1 are the two top nodes on the stack. In this configuration we can delay the swap until *did* has combined with its subject *you* by means of a RIGHT-ARC transition, but if we do not swap in the second configuration where this is possible, we eventually end up with the stack $[\text{ROOT}_0, who_6, did_1, send_3, to_7]$. Here we cannot attach *who* to *to* by means of a LEFT-ARC transition and get stuck.

In order to define the new oracle, we introduce an auxiliary concept. Consider a modification of the oracle τ_1 from Figure 2 that cannot predict SWAP transitions. This oracle will be able to produce valid transition sequences only for *projective* target trees; for non-projective trees, it will fail to reconstruct all dependency arcs. More specifically, a parse with this oracle will end up in a configuration in which the set of constructed dependency arcs forms a *set* of projective dependency trees, not necessarily a single such tree. We call the elements of this set the *maximal projective components* of the target tree. To illustrate the notion, we have drawn boxes around nodes in the same component in Figures 1.

Based on the concept of maximal projective components, we define a new training oracle τ_2, which delays swapping as long as the next node in the input is in the same maximal projective component as the top node on the stack. The definition of the new oracle τ_2 is identical to that of τ_1 except that the third line is replaced by "SWAP if $c = ([\sigma|i, j], [k|\beta], A_c)$, $j <_G i$, and $\text{MPC}(j) \neq \text{MPC}(k)$", where $\text{MPC}(i)$ is the maximal projective component of node i. As a special case, τ_2 predicts SWAP if $j <_G i$ and the buffer B is empty.

$$\tau_1(c) = \begin{cases} \text{LEFT-ARC}_l & \text{if } c = ([\sigma|i,j], B, A_c),\ (j,l,i) \in A \text{ and } A^i \subseteq A_c \\ \text{RIGHT-ARC}_l & \text{if } c = ([\sigma|i,j], B, A_c),\ (i,l,j) \in A \text{ and } A^j \subseteq A_c \\ \text{SWAP} & \text{if } c = ([\sigma|i,j], B, A_c) \text{ and } j <_G i \\ \text{SHIFT} & \text{otherwise} \end{cases}$$

Figure 2: Training oracle τ_1 for an arbitrary target tree $G = (V_x, A)$, following the notation of Nivre (2009), where $c = (\Sigma, B, A_c)$ denotes a configuration c with stack Σ, input buffer B and arc set A_c. We write A^i to denote the subset of A that only contains the outgoing arcs of the node i. (Note that A_c is the arc set in configuration c, while A is the arc set in the target tree G.)

For example, in extracting the transition sequence for the target tree in Figure 1, the new oracle will postpone swapping of *did* when *you* is the next node in the input, but not postpone when the next node is *send*. We can show that a parser informed by the new training oracle can always proceed to a terminal configuration, and still derive all (even non-projective) dependency trees.

4 Experiments

We now test the hypothesis that the new training oracle can improve both the accuracy and the efficiency of a transition-based dependency parser. Our experiments are based on the same five data sets as Nivre (2009). The training sets vary in size from 28,750 tokens (1,534 sentences) for Slovene to 1,249,408 tokens (72,703 sentences) for Czech, while the test sets all consist of about 5,000 tokens.

4.1 Number of Transitions

For each language, we first parsed the training set with both the old and the new training oracle. This allowed us to compare the number of SWAP transitions needed to parse all sentences with the two oracles, shown in Table 1. We see that the reduction is very substantial, ranging from 55% (for Czech) to almost 84% (for Arabic). While this difference does not affect the asymptotic complexity of parsing, it may reduce the number of calls to the classifier, which is where transition-based parsers spend most of their time.

4.2 Parsing Accuracy

In order to assess whether the reduced number of SWAP transitions also has a positive effect on parsing accuracy, we trained two parsers for each of the five languages, one for each oracle. All systems use SVM classifiers with a polynomial kernel with features and parameters optimized separately for each language and training oracle. The training data for these classifiers consist only of the sequences derived by the oracles, which means that the parser has no explicit notion of projective order or maximal projective components at parsing time.

Table 2 shows the labeled parsing accuracy of the parsers measured by the overall attachment score (AS), as well as labeled precision, recall and (balanced) F-score for non-projective dependencies.[1] For comparison, we also give results for the two best performing systems in the original CoNLL-X shared task, Malt (Nivre et al., 2006) and MST (McDonald et al., 2006), as well as the combo system MST_{Malt}, (Nivre and McDonald, 2008).

Looking first at the overall labeled attachment score, we see that the new training oracle consistently gives higher accuracy than the old one, with differences of up to 0.5 percentage points (for Arabic and Slovene), which is substantial given that the frequency of non-projective dependencies is only 0.4–1.9%. Because the test sets are so small, none of the differences is statistically significant (McNemar's test, $\alpha = .05$), but the consistent improvement over all languages nevertheless strongly suggests that this is a genuine difference.

In relation to the state of the art, we note that the parsers with online reordering significantly outperform Malt and MST on Czech and Slovene, and MST on Turkish, and have significantly lower scores than the combo system MST_{Malt} only for Arabic and Danish. For Czech, the parser with the new oracle actually has the highest attachment score ever reported, although the difference with respect to MST_{Malt} is not statistically significant.

Turning to the scores for non-projective dependencies, we again see that the new oracle consistently gives higher scores than the old oracle, with

[1] These metrics are not meaningful for Arabic as the test set only contains 11 non-projective dependencies.

	Arabic	Czech	Danish	Slovene	Turkish
Old (τ_1)	1416	57011	8296	2191	2828
New (τ_2)	229	26208	1497	690	1253
Reduction (%)	83.8	55.0	82.0	68.5	55.7

Table 1: Number of SWAP transitions for the old (τ_1) and new (τ_2) training oracle.

	Arabic	Czech				Danish				Slovene				Turkish			
System	**AS**	**AS**	**P**	**R**	**F**	**AS**	**P**	**R**	**F**	**AS**	**P**	**R**	**F**	**AS**	**P**	**R**	**F**
Old (τ_1)	67.2	82.5	74.7	**72.9**	73.8	84.2	30.0	30.0	30.0	75.2	33.3	26.4	29.5	64.7	12.5	11.4	11.9
New (τ_2)	67.5	**82.7**	**79.3**	71.0	**79.3**	84.3	38.2	32.5	35.1	75.7	**60.6**	**27.6**	**37.9**	65.0	14.3	**13.2**	**13.7**
Malt	66.7	78.4	76.3	57.9	65.8	84.8	45.8	27.5	34.4	70.3	45.9	20.7	25.1	65.7	**16.7**	9.2	11.9
MST	66.9	80.2	60.5	61.7	61.1	84.8	54.0	**62.5**	57.9	73.4	33.7	26.4	29.6	63.2	–	11.8	–
MST_{Malt}	**68.6**	82.3	63.9	69.2	66.1	**86.7**	**63.0**	60.0	**61.5**	**75.9**	31.6	**27.6**	29.5	**66.3**	11.1	9.2	10.1

Table 2: Labeled attachment score (AS) overall; precision (P), recall (R) and balanced F-score (F) for non-projective dependencies. Old = τ_1; New = τ_2; Malt = Nivre et al. (2006), MST = McDonald et al. (2006), MST_{Malt} = Nivre and McDonald (2008).

the single exception that the old one has marginally higher recall for Czech. Moreover, the reordering parser with the new oracle has higher F-score than any other system for all languages except Danish. Especially the result for Czech, with 79.3% precision and 71.0% recall, is remarkably good, making the parser almost as accurate for non-projective dependencies as it is for projective dependencies. It seems likely that the good results for Czech are due to the fact that Czech has the highest percentage of non-projective structures in combination with the (by far) largest training set.

5 Conclusion

We have presented a new training oracle for the transition system originally presented in Nivre (2009). This oracle postpones swapping as long as possible but still fulfills the correctness criterion. Our experimental results show that the new training oracle can reduce the necessary number of swaps by more than 80%, and that parsers trained in this way achieve higher precision and recall on non-projective dependency arcs as well as higher attachment score overall. The results are particularly good for languages with a high percentage of non-projective dependencies, with an all-time best over all metrics for Czech.

An interesting theoretical question is whether the new oracle defined in this paper is optimal with respect to minimizing the number of swaps. The answer turns out to be negative, and it is possible to reduce the number of swaps even further by generalizing the notion of maximal projective components to maximal components that may be non-projective. However, the characterization of these generalized maximal components is non-trivial, and is therefore an important problem for future research.

References

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 507–514.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT*, pages 122–131.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL*, pages 216–220.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950–958.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:513–553.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of ACL-IJCNLP*.