Don't Stop Me Now!
Using Global Dynamic Oracles to Correct Training Biases of Transition-Based Dependency Parsers

Lauriane Aufrant[1,2], **Guillaume Wisniewski**[1] and **François Yvon**[1]

[1]LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91 405 Orsay, France
[2]DGA, 60 boulevard du Général Martial Valin, 75 509 Paris, France
{lauriane.aufrant,guillaume.wisniewski,francois.yvon}@limsi.fr

Abstract

This paper formalizes a sound extension of dynamic oracles to global training, in the frame of transition-based dependency parsers. By dispensing with the precomputation of references, this extension widens the training strategies that can be entertained for such parsers; we show this by revisiting two standard training procedures, *early-update* and *max-violation*, to correct some of their search space sampling biases. Experimentally, on the SPMRL treebanks, this improvement increases the similarity between the train and test distributions and yields performance improvements up to 0.7 UAS, without any computation overhead.

1 Introduction

Transition-based parsers with beam search are among the most widely used models for dependency parsing: they achieve state-of-the-art performance while their training and inference, which rely on approximate search, are very efficient. Training a beam parser faces two difficulties: error propagation and search errors (Huang et al., 2012). Specific learning methods, *early-update* and *max-violation* (presented in §2), have been designed to address them. But they require to update the parameters on partial derivations only, which introduces a discrepancy between the feature distributions seen during training and testing. Notably, derivation endings are under-represented during training, which hurts parsing performance.

In this work, we propose an improved training strategy that corrects such sampling biases for beam parsers (§3). Experiments with the SPMRL treebanks (Seddah et al., 2013), reported in §4, show that the training configurations sampled by this new strategy are closer to the parser configurations seen at test time and result in increases up to 0.7 UAS, with no computation time overhead. These improvements rely on a sound extension of dynamic oracles for global training, the lack of which has repeatedly been pointed out (Goldberg and Nivre, 2012; Sartorio, 2015). These global dynamic oracles have more general benefits than the training strategy proposed here; for instance, they allow to train beam parsers on partially annotated data in a context of active learning or multilingual transfer (Lacroix et al., 2016).

2 Training a Dependency Parser

In a transition-based parser (Nivre, 2008), a parse is computed by performing a sequence of *transitions* building the parse tree in an incremental fashion. In the following, c denotes a *parser configuration* representing a partially built dependency tree. Applying transition t to configuration c results in the parser moving to a *successor* of c, denoted $c \circ t$.

At each step of the parsing process, every possible transition is scored by a classifier, given a feature representation of c and model parameters $\boldsymbol{\theta}$; the score of a *derivation* (a sequence of transitions) generating a given parse tree is the sum of its transition scores. Parsing thus amounts to finding the derivation having the highest score, usually through greedy or beam search.

Parsers using beam search are typically trained with a global criterion, that updates the parameters once for each training sentence. Algorithm 1 summarizes the training for each sentence x (with gold parse y): $\textsc{Initial}(x)$ denotes the initial configuration for x and the procedure $\textsc{Oracle}$ performs decoding to find configurations that play the role of the 'positive' and 'negative' examples (resp. c^+ and c^-) required by the $\textsc{Update}$ operation (typi-

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 318–323,
Valencia, Spain, April 3-7, 2017. ©2017 Association for Computational Linguistics

Algorithm 1: Global training on one sentence.

```
$\theta$: model parameters, initialized to $\theta_0$ before training
Function DpTraining(x,y)
    $c \leftarrow$ Initial($x$)
    $c^+, c^- \leftarrow$ Oracle($c$, y, $\theta$)
    $\theta \leftarrow$ Update($\theta, c^+, c^-$)
```

cally a perceptron update rule (Collins and Roark, 2004) or a gradient computation with the globally normalized loss of Andor et al. (2016)). Several strategies, corresponding to various implementations of the ORACLE function, have been used to find these examples.

In the *early-update* strategy (Collins and Roark, 2004; Zhang and Clark, 2008), a reference derivation is first computed, generally using hand-crafted heuristics. The sentence is then parsed using conventional beam decoding and an update happens as soon as this pre-computed gold derivation falls off the beam, while the rest of the sequence is ignored. The top scoring configuration at this step is penalized and the reference that has just fallen off the beam is reinforced. Another strategy, *max-violation* (Huang et al., 2012), is to continue decoding even though the reference has fallen off the beam, in order to find the configuration having the largest gap between the scores of the (partial) hypothesis and the (partial) gold derivation. Compared to *early-update*, *max-violation* speeds up convergence by covering longer transition sequences and can yield slightly better parsers.

3 Correction of Training Biases

Both standard learning strategies suffer from biases that introduce a discrepancy between the feature distributions seen during training and testing.

First, parameters updates reinforce only gold derivations; at test time, the model might find itself, after an error, in a part of the search space where it was not trained to take good decisions, thus propagating errors (Goldberg and Nivre, 2012).[1]

Second, they both use a *static oracle* that relies on the deterministic pre-computation of a canonical reference. An update occurs as soon as the parser strays from this particular gold derivation, even when the reference tree could still be obtained using an alternative derivation. Updating in such cases raises the risk of lowering parser performance. Indeed, we measured that a beam parser trained with *early-update* and a static oracle counter-intuitively predicts correctly *fewer* heads of the current sentence just after an update than just before, for 15% of the updates (French SPMRL, during 10th epoch).

Third, both the *early-update* and the *max-violation* strategies consider only partial derivations when updating the model parameters. For instance on the French SPMRL, when training with an *early-update* strategy, the end of the derivation is reached for only 41% of the examples at the 10th epoch[2] and, on average, only 57% of a derivation is considered; the *max-violation* strategy, which computes longer partial derivations, partly alleviates this effect: these proportions raise, respectively, to 53% and 81%. While the choice of partial updates has been experimentally proved (Huang et al., 2012) to be critical in achieving good performance, it prevents parsers from visiting configurations corresponding to derivation endings. This explains why configurations and transitions involving final punctuation marks, verbs in SOV languages like Japanese or German subordinate clauses, the ROOT token when placed at the end (Ballesteros and Nivre, 2013), but also stack features involving long distance siblings, are too rarely seen in training, thereby hurting predictions in such configurations.

In the following, we describe improvements addressing those issues.

Dynamic oracles The limits of static oracles have already been highlighted for ARCEAGER greedy parsers: Goldberg and Nivre (2012) show how parsing performance can be significantly improved with a dynamic oracle that computes a reference tailored to the current parser state. Dynamic oracles are at the heart of most state-of-the-art parsers (Ballesteros et al., 2016; Coavoux and Crabbé, 2016; Cross and Huang, 2016; Kiperwasser and Goldberg, 2016). But, to the best of our knowledge, dynamic oracles have only been partially generalized to beam parsers: Björkelund and Nivre (2015)'s oracles address the second but

[1] While beam search already addresses error propagation issues that are due to inexact search, it does not handle this kind of error propagation, which results from training issues.

[2] On the French SPMRL treebank, at the 10th epoch, the parser is close to convergence (see §4).

not the first issue, while the dynamic oracle of the YaraParser (Rasooli and Tetreault, 2015) arbitrarily rules out some configurations that can generate the reference tree.

Algorithm 2 shows how a dynamic oracle can be integrated within the *early-update* learning strategy; this extension can be done in the same way for the *max-violation* strategy but is not detailed here, for space reasons. The specificity of that formalism is to consider that an error occurs only when none of the configurations in the beam can result in the dependency tree that was initially the best reachable one, i.e. when all hypotheses insert new erroneous dependencies.[3]

The Boolean function that tests this condition, denoted $\textsc{Correct}_y(c'|c)$, can be efficiently computed using the $\textsc{Cost}_y(t)$ function, formally defined in Goldberg and Nivre (2013) as the number of dependencies of a gold parse tree y that can no longer be predicted when transition t is applied: a configuration c' is considered as CORRECT in the context of a configuration c, if there exists a sequence of transitions $t_1, \ldots, t_n$ such that $c' = c \circ t_1 \circ \ldots \circ t_n$ and $\textsc{Cost}_y(t_1) = \cdots = \textsc{Cost}_y(t_n) = 0$.

Once an error is detected, the negative example c^- is chosen, as in the 'standard' *early-update* strategy, as the top scoring configuration in the beam. The positive example c^+ is computed in constant time, by choosing the top scoring configuration in the beam (just before k-best truncation) for which CORRECT is true.

Restart Strategy To avoid over-representing the beginning of derivations during training, we propose a new learning strategy: contrary to the baseline training method (Algorithm 1) in which parsing stops as soon as an error is detected and the parameters updated, in our strategy (Algorithm 3) decoding is restarted with a beam containing only the positive configuration c^+ and parsing continues until a new error is detected, triggering new updates. The ORACLE function is then called from several successive configurations, as many times as needed to completely parse the sentence.

This training method ensures that configurations that are close to derivations endings will be seen more often during training.[4]

Algorithm 2: Dynamic oracle for the *early-update* strategy.

```
$c_0$: configuration to start decoding from
$top_\theta(\cdot)$: best scoring element according to $\theta$
$\textsc{Next}(c)$: the set of all successors of $c$
Function EarlyUpdateOracle($c_0$, $y$, $\theta$)
    Beam ← $\{c_0\}$
    while $\exists c \in Beam, \neg\textsc{Final}(c)$ do
        $S \leftarrow \cup_{c \in \text{Beam}} \textsc{Next}(c)$
        Beam ← $k$-best($S$, $\theta$)
        if $\forall c \in Beam, \neg\textsc{Correct}_y(c|c_0)$ then
            gold ←
                $\{c \in S | \textsc{Correct}_y(c|c_0)\}$
            return $top_\theta$(gold), $top_\theta$(Beam)
    gold ← $\{c \in \text{Beam} | \textsc{Correct}_y(c|c_0)\}$
    return $top_\theta$(gold), $top_\theta$(Beam)
```

Algorithm 3: Global training with restart.

```
$\textsc{Final}(\cdot)$: true iff the whole sentence is parsed
Function DpTrainingRestart($x$,$y$)
    $c \leftarrow \textsc{Initial}(x)$
    while $\neg\textsc{Final}(c)$ do
        $c^+, c^- \leftarrow \textsc{Oracle}(c, y, \theta)$
        $\theta \leftarrow \textsc{Update}(\theta, c^+, c^-)$
        $c \leftarrow c^+$
```

Restarting with an oracle tailored to the restart configuration is made possible by our global dynamic oracle. In this frame, the strategy can even be further improved: similarly to their greedy counterpart, global dynamic oracles enable to augment training with an error exploration component by restarting from c^- instead of c^+ after an error, thus addressing the first issue mentioned.

4 Experiments

Experimental Setup The validity of our approach is evaluated on the SPMRL treebank (Seddah et al., 2013). We consider, as baselines, a greedy parser trained with a dynamic oracle (GREEDY DYN) and beam parsers trained with the *early-update* and *max-violation* strategies and a static oracle (resp. EARLY and MAXV). The im-

[3] While fairly simple, this formalism is a major change from the traditional paradigm where references are explicitly computed for each action.

[4] Standard training with full update also ensures this, but with the risk of divergence (Huang et al., 2012). Restarting in c^+ with a new beam has the same convergence guarantee as standard *early-update* and *max-violation*.

	ar	de	eu	fr	he	hu	ko	pl	sv	average
GREEDY DYN	83.98	90.73	84.00	84.23	83.78	84.33	82.79	87.66	86.35	85.32
EARLY	85.03	92.74	84.42	86.02	85.39	85.63	82.73	89.60	87.00	86.51
IMP-EARLY	**85.27**	92.89	84.59	86.26	**85.84**	**85.74**	**82.98**	89.55	**87.37**	86.72
MAXV	85.06	92.77	84.59	86.10	85.53	85.57	82.68	89.42	87.16	86.54
IMP-MAXV	85.04	**92.90**	**84.68**	**86.26**	85.83	85.55	82.94	**90.12**	87.31	**86.74**

Table 1: Performance (UAS) of the various training strategies on the SPMRL datasets.

provements of §3 are applied to these two strategies (resp. IMP-EARLY and IMP-MAXV).

In all our experiments, we use our in-house, open source implementation of a beam ARCEAGER parser in the PanParser framework (Aufrant and Wisniewski, 2016),[5] with the averaged structured perceptron (Collins, 2002), a beam size of 8 and the ROOT placed at the end. We use coarse gold PoS tags and the extended features set of Zhang and Nivre (2011), without label information. These features, designed for English, have not been adapted to the specificities of the languages. All models are trained up to convergence on a validation set. As a point of comparison, on average over the treebank, our GREEDY DYN baseline is 2.7 UAS higher than a MaltParser trained with ARCEAGER and the same kind of information (coarse tags, no label).

Results Table 1 reports the performance of all training strategies evaluated by the traditional UAS on the projective test sets, ignoring punctuation tokens. All reported scores are averaged over 5 runs. Results show that our learning strategy consistently outperforms the corresponding baseline, with average increases of 0.2 UAS, up to 0.7 UAS.

Discussion Table 2 shows the performance imbalance between various positions in the sentence and confirms that our improvements partly alleviate this phenomenon: the scores on the first half of the sentence are mostly unchanged, while large gains are reported on the second half.

To assess that these UAS gains result from a better matching of training and test configurations, we compute the Kullback-Leibler divergence between the probability distribution (estimated with frequency counts and 0.1 Laplace smoothing) of the features of all configurations in beam scored during the 10th training epoch and the feature distribution seen at test time.

Quarter	1st	2nd	3rd	4th
EARLY	90.0	85.4	83.1	84.7
IMP-EARLY	90.0	85.3	84.2	85.1

Table 2: Performance (UAS) of the standard and improved *early-update* strategies, depending on the position in the sentence (French SPMRL dataset, with similar results in other languages). The first quarter corresponds to the attachment of tokens in the first 25% of the sentence length.

	Baseline	Improved
EARLY	0.350	0.280
MAXV	0.357	0.277

Table 3: Effect of our improvements on the Kullback-Leibler divergence between the train and test feature distributions (French SPMRL dataset, with similar results in other languages).

Table 3 reports the Kullback-Leibler divergences induced by our refinements with respect to the corresponding baselines. It clearly shows that our 'improved' learning strategy considers training examples that are closer to test configurations. Similar experiments on greedy parsers show that their train-test divergence is reduced from 0.320 to 0.219 by the dynamic oracle and exploration strategy of Goldberg and Nivre (2012). In these two experiments, feature similarity correlates with UAS improvements and can therefore provide a new way to interpret oracle influence.

Finally, regarding efficiency, we observe (Figure 1) that IMP-EARLY converges in a number

[5] The oracle for beam parsers described in this work can be used with any scoring function and learning method, such as Andor et al. (2016). But its implementation may require to change the whole code architecture as reference derivations must be computed on the fly.

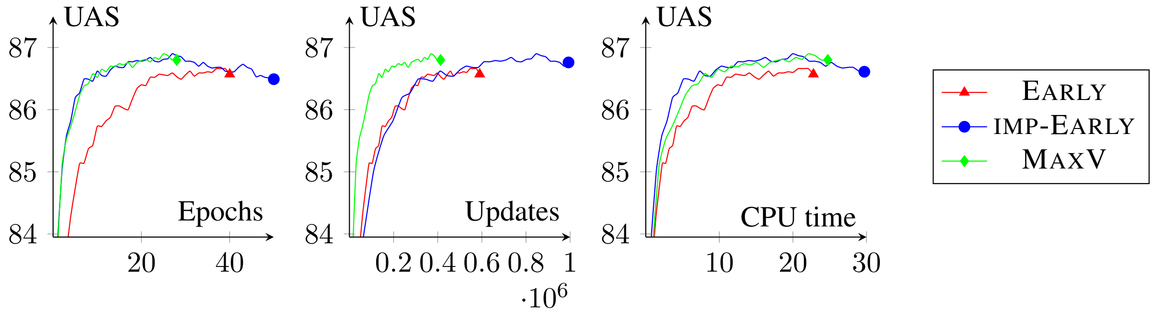

Figure 1: Learning curves on the validation set (SPMRL, fr). IMP-EARLY has the same update efficiency as EARLY, but with the epoch and computation time convergence of MAXV.

of epochs similar to that of standard MAXV. Despite an increased number of updates, it is however slightly faster (in CPU time) because it avoids the extra reference pre-computation.

5 Conclusion

In this paper, we have extended the dynamic oracle framework to global training, for transition-based dependency parsers. This innovation lets us propose an alternative training strategy, that reduces the discrepancy between the feature distributions seen at train and test time that exists in state-of-the-art methods. Experiments on the 9 SPMRL treebanks show that our restart strategy improves both parsing accuracy and model convergence. We intend for future work to investigate other ways to reduce the train-test distribution discrepancy in structured prediction, using the new possibilities offered by this extended framework.

Acknowledgments

This work has been partly funded by the French *Direction générale de l'armement*.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany, August. Association for Computational Linguistics.

Lauriane Aufrant and Guillaume Wisniewski. 2016. PanParser: a Modular Implementation for Efficient Transition-Based Dependency Parsing. Technical report, LIMSI-CNRS, March.

Miguel Ballesteros and Joakim Nivre. 2013. Going to the roots of dependency parsing. *Computational Linguistics*, 39(1):5–13.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack lstm parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2005–2010, Austin, Texas, November. Association for Computational Linguistics.

Anders Björkelund and Joakim Nivre. 2015. Non-Deterministic Oracles for Unrestricted Non-Projective Transition-Based Dependency Parsing. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 76–86.

Maximin Coavoux and Benoit Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 172–182, Berlin, Germany, August. Association for Computational Linguistics.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas, November. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India, December. The COLING 2012 Organizing Committee.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151, Montréal, Canada, June. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California, June. Association for Computational Linguistics.

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

Mohammad Sadegh Rasooli and Joel Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *arXiv preprint arXiv:1503.06733*.

Francesco Sartorio. 2015. *Improvements in Transition Based Systems for Dependency Parsing*. Ph.D. thesis, Universit degli studi di Padova.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA, October. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii, October. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.