Training with Exploration Improves a Greedy Stack LSTM Parser

Miguel Ballesteros[♢] **Yoav Goldberg**[♣] **Chris Dyer**[♠] **Noah A. Smith**[♡]
[♢]NLP Group, Pompeu Fabra University, Barcelona, Spain
[♣]Computer Science Department, Bar-Ilan University, Ramat Gan, Israel
[♠]Google DeepMind, London, UK
[♡]Computer Science & Engineering, University of Washington, Seattle, WA, USA
miguel.ballesteros@upf.edu, yoav.goldberg@gmail.com,
cdyer@google.com, nasmith@cs.washington.edu

Abstract

We adapt the greedy stack LSTM dependency parser of Dyer et al. (2015) to support a training-with-exploration procedure using dynamic oracles (Goldberg and Nivre, 2013) instead of assuming an error-free action history. This form of training, which accounts for model predictions at training time, improves parsing accuracies. We discuss some modifications needed in order to get training with exploration to work well for a probabilistic neural network dependency parser.

1 Introduction

Natural language parsing can be formulated as a series of decisions that read words in sequence and incrementally combine them to form syntactic structures; this formalization is known as transition-based parsing, and is often coupled with a greedy search procedure (Yamada and Matsumoto, 2003; Nivre, 2003; Nivre, 2004; Nivre, 2008). The literature on transition-based parsing is vast, but all works share in common a classification component that takes into account features of the current parser state[1] and predicts the next action to take conditioned on the state. The state is of unbounded size.

Dyer et al. (2015) presented a parser in which the parser's unbounded state is embedded in a fixed-dimensional continuous space using recurrent neural networks. Coupled with a recursive tree composition function, the feature representation is able to capture information from the entirety of the state, without resorting to locality assumptions that were common in most other transition-based parsers. The use of a novel stack LSTM data structure allows the parser to maintain a constant time per-state update, and retain an overall linear parsing time.

The Dyer et al. parser was trained to maximize the likelihood of gold-standard transition sequences, given words. At test time, the parser makes *greedy* decisions according to the learned model. Although this setup obtains very good performance, the training and testing conditions are mismatched in the following way: at training time the historical context of an action is always derived from the gold standard (i.e., perfectly correct past actions), but at test time, it will be a model prediction.

In this work, we adapt the training criterion so as to explore parser states drawn not only from the training data, but also from the model as it is being learned. To do so, we use the method of Goldberg and Nivre (2012; 2013) to dynamically chose an optimal (relative to the final attachment accuracy) action given an imperfect history. By interpolating between algorithm states sampled from the model and those sampled from the training data, more robust predictions at test time can be made. We show that the technique can be used to improve the strong parser of Dyer et al.

2 Parsing Model and Parameter Learning

Our departure point is the parsing model described by Dyer et al. (2015). We do not describe the model in detail, and refer the reader to the original work. At each stage t of the parsing process, the parser state is

[1]The term "state" refers to the collection of previous decisions (sometimes called the history), resulting partial structures, which are typically stored in a stack data structure, and the words remaining to be processed.

encoded into a vector $\mathbf{p}_t$, which is used to compute the probability of the parser action at time t as:

$$p(z_t \mid \mathbf{p}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^\top \mathbf{p}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}(S,B)} \exp\left(\mathbf{g}_{z'}^\top \mathbf{p}_t + q_{z'}\right)}, \quad (1)$$

where $\mathbf{g}_z$ is a column vector representing the (output) embedding of the parser action z, and q_z is a bias term for action z. The set $\mathcal{A}(S, B)$ represents the valid transition actions that may be taken in the current state. Since $\mathbf{p}_t$ encodes information about all previous decisions made by the parser, the chain rule gives the probability of any valid sequence of parse transitions $\boldsymbol{z}$ conditional on the input:

$$p(\boldsymbol{z} \mid \boldsymbol{w}) = \prod_{t=1}^{|\boldsymbol{z}|} p(z_t \mid \mathbf{p}_t). \quad (2)$$

The parser is trained to maximize the conditional probability of taking a "correct" action at each parsing state. The definition of what constitutes a "correct" action is the major difference between a static oracle as used by Dyer et al. (2015) and the dynamic oracle explored here.

Regardless of the oracle, our training implementation constructs a computation graph (nodes that represent values, linked by directed edges from each function's inputs to its outputs) for the negative log probability for the oracle transition sequence as a function of the current model parameters and uses forward- and backpropagation to obtain the gradients respect to the model parameters (Lecun et al., 1998, section 4).

2.1 Training with Static Oracles

With a static oracle, the training procedure computes a canonical reference series of transitions for each gold parse tree. It then runs the parser through this canonical sequence of transitions, while keeping track of the state representation $\mathbf{p}_t$ at each step t, as well as the distribution over transitions $p(z_t \mid \mathbf{p}_t)$ which is predicted by the current classifier for the state representation. Once the end of the sentence is reached, the parameters are updated towards maximizing the likelihood of the reference transition sequence (Equation 2), which equates to maximizing the probability of the correct transition, $p(z_{g_t} \mid \mathbf{p_t})$, at each state along the path.

2.2 Training with Dynamic Oracles

In the static oracle case, the parser is trained to predict the best transition to take at each parsing step, assuming all previous transitions were correct. Since the parser is likely to make mistakes at test time and encounter states it has not seen during training, this training criterion is problematic (Daumé III et al., 2009; Ross et al., 2011; Goldberg and Nivre, 2012; Goldberg and Nivre, 2013, *inter alia*). Instead, we would prefer to train the parser to behave optimally even after making a mistake (under the constraint that it cannot backtrack or fix any previous decision). We thus need to include in the training examples states that result from wrong parsing decisions, together with the optimal transitions to take in these states. To this end we reconsider which training examples to show, and what it means to behave optimally on these training examples. The framework of training with exploration using dynamic oracles suggested by Goldberg and Nivre (2012; 2013) provides answers to these questions. While the application of dynamic oracle training is relatively straightforward, some adaptations were needed to accommodate the probabilistic training objective. These adaptations mostly follow Goldberg (2013).

Dynamic Oracles. A *dynamic oracle* is the component that, given a gold parse tree, provides the optimal set of possible actions to take for any valid parser state. In contrast to static oracles that derive a canonical state sequence for each gold parse tree and say nothing about states that deviate from this canonical path, the dynamic oracle is well defined for states that result from parsing mistakes, and they may produce more than a single gold action for a given state. Under the dynamic oracle framework, an action is said to be optimal for a state if the best tree that can be reached after taking the action is no worse (in terms of accuracy with respect to the gold tree) than the best tree that could be reached prior to taking that action.

Goldberg and Nivre (2013) define the arc-decomposition property of transition systems, and show how to derive efficient dynamic oracles for transition systems that are arc-decomposable.[2] Unfortunately, the arc-standard transition system does

[2]Specifically: for every parser configuration $\mathbf{p}$ and group of

not have this property. While it is possible to compute dynamic oracles for the arc-standard system (Goldberg et al., 2014), the computation relies on a dynamic programming algorithm which is polynomial in the length of the stack. As the dynamic oracle has to be queried for each parser state seen during training, the use of this dynamic oracle will make the training runtime several times longer. We chose instead to switch to the arc-hybrid transition system (Kuhlmann et al., 2011), which is very similar to the arc-standard system but is arc-decomposable and hence admits an efficient $O(1)$ dynamic oracle, resulting in only negligible increase to training runtime. We implemented the dynamic oracle to the arc-hybrid system as described by Goldberg (2013).

Training with Exploration. In order to expose the parser to configurations that are likely to result from incorrect parsing decisions, we make use of the probabilistic nature of the classifier. During training, instead of following the gold action, we sample the next transition according to the output distribution the classifier assigns to the current configuration. Another option, taken by Goldberg and Nivre, is to follow the one-best action predicted by the classifier. However, initial experiments showed that the one-best approach did not work well. Because the neural network classifier becomes accurate early on in the training process, the one-best action is likely to be correct, and the parser is then exposed to very few error states in its training process. By sampling from the predicted distribution, we are effectively increasing the chance of straying from the gold path during training, while still focusing on mistakes that receive relatively high parser scores. We believe further formal analysis of this method will reveal connections to reinforcement learning and, perhaps, other methods for learning complex policies.

Taking this idea further, we could increase the number of error-states observed in the training process by changing the sampling distribution so as to bias it toward more low-probability states. We do this by raising each probability to the power of α ($0 < \alpha \leq 1$) and re-normalizing. This transformation keeps the relative ordering of the events, while shifting probability mass towards less frequent events. As we show below, this turns out to be very beneficial for the configurations that make use of external embeddings. Indeed, these configurations achieve high accuracies and sharp class distributions early on in the training process.

The parser is trained to maximize the likelihood of a correct action z_g at each parsing state $\mathbf{p}_t$ according to Equation 1. When using the dynamic oracle, a state $\mathbf{p}_t$ may admit multiple correct actions $\boldsymbol{z_g} = \{z_{g_i}, \ldots, z_{g_k}\}$. Our objective in such cases is the marginal likelihood of all correct actions,[3]

$$p(\boldsymbol{z_g} \mid \mathbf{p}_t) = \sum_{z_{g_i} \in \boldsymbol{z_g}} p(z_{g_i} \mid \mathbf{p}_t). \qquad (3)$$

3 Experiments

Following the same settings of Chen and Manning (2014) and Dyer et al (2015) we report results[4] in the English PTB and Chinese CTB-5. Table 1 shows the results of the parser in its different configurations. The table also shows the best result obtained with the static oracle (obtained by rerunning Dyer et al. parser) for the sake of comparison between static and dynamic training strategies.

	English		Chinese	
Method	UAS	LAS	UAS	LAS
Arc-standard (Dyer et al.)	92.40	90.04	85.48	83.94
Arc-hybrid (static)	92.08	89.80	85.66	84.03
Arc-hybrid (dynamic)	92.66	90.43	86.07	84.46
Arc-hybrid (dyn., $\alpha = 0.75$)	92.73	90.60	86.13	84.53
+ pre-training:				
Arc-standard (Dyer et al.)	93.04	90.87	86.85	85.36
Arc-hybrid (static)	92.78	90.67	86.94	85.46
Arc-hybrid (dynamic)	93.15	91.05	87.05	85.63
Arc-hybrid (dyn., $\alpha = 0.75$)	**93.56**	**91.42**	**87.65**	**86.21**

Table 1: Dependency parsing: English (SD) and Chinese.

The score achieved by the dynamic oracle for English is 93.56 UAS. This is remarkable given that the parser uses a completely greedy search procedure. Moreover, the Chinese score establishes the state-of-the-art, using the same settings as Chen and Manning (2014).

arcs A, if each arc in A can be derived from $\mathbf{p}$, then a valid tree structure containing *all* of the arcs in A can also be derived from $\mathbf{p}$. This is a sufficient condition, but whether it is necessary is unknown; hence the question of an efficient, $O(1)$ dynamic oracle for the augmented system is open.

[3]A similar objective was used by Riezler et al (2000), Charniak and Johnson (2005) and Goldberg (2013) in the context of log-linear probabilistic models.

[4]The results on the development sets are similar and only used for optimization and validation.

	Catalan		Chinese		Czech		English		German		Japanese		Spanish	
Method	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Arc-standard, static + PP	89.60	85.45	79.68	75.08	77.96	71.06	91.12	88.69	88.09	85.24	93.10	92.28	89.08	85.03
+ pre-training	–	–	82.45	78.55	–	–	91.59	89.15	88.56	86.15	–	–	90.76	87.48
Arc-hybrid, dyn. + PP	90.45	86.38	80.74	76.52	85.68	79.38	91.62	89.23	89.80	87.29	93.47	92.70	89.53	85.69
+ pre-training	–	–	**83.54**	**79.66**	–	–	**92.22**	**89.87**	**90.34**	**88.17**	–	–	**91.09**	87.95
Y'15	–	–	–	–	85.2	77.5	90.75	88.14	89.6	86.0	–	–	88.3	85.4
A'16 + pre-training	**91.24**	**88.21**	81.29	77.29	**85.78**	**80.63**	91.44	89.29	89.12	86.95	**93.71**	**92.85**	91.01	**88.14**
A'16-beam	92.67	89.83	84.72	80.85	88.94	84.56	93.22	91.23	90.91	89.15	93.65	92.84	92.62	89.95

Table 2: Dependency parsing results. The dynamic oracle uses $\alpha = 0.75$ (selected on English; see Table 1). PP refers to pseudo-projective parsing. Y'15 and A'16 are beam = 1 parsers from Yazdani and Henderson (2015) and Andor et al. (2016), respectively. A'16-beam is the parser with beam larger than 1 by Andor et al. (2016). Bold numbers indicate the best results among the greedy parsers.

The error-exploring dynamic-oracle training always improves over static oracle training controlling for the transition system, but the arc-hybrid system slightly under-performs the arc-standard system when trained with static oracle. Flattening the sampling distribution ($\alpha = 0.75$) is especially beneficial when training with pretrained word embeddings.

In order to be able to compare with similar greedy parsers (Yazdani and Henderson, 2015; Andor et al., 2016)[5] we report the performance of the parser on the multilingual treebanks of the CoNLL 2009 shared task (Hajič et al., 2009). Since some of the treebanks contain nonprojective sentences and arc-hybrid does not allow nonprojective trees, we use the pseudo-projective approach (Nivre and Nilsson, 2005). We used predicted part-of-speech tags provided by the CoNLL 2009 shared task organizers. We also include results with pretrained word embeddings for English, Chinese, German, and Spanish following the same training setup as Dyer et al. (2015); for English and Chinese we used the same pretrained word embeddings as in Table 1, for German we used the monolingual training data from the WMT 2015 dataset and for Spanish we used the Spanish Gigaword version 3. See Table 2.

4 Related Work

Training greedy parsers on non-gold outcomes, facilitated by dynamic oracles, has been explored by several researchers in different ways (Goldberg and Nivre, 2012; Goldberg and Nivre, 2013; Goldberg et al., 2014; Honnibal et al., 2013; Honnibal and Johnson, 2014; Gómez-Rodríguez et al., 2014; Björkelund and Nivre, 2015; Tokgöz and Eryiğit, 2015; Gómez-Rodríguez and Fernández-González, 2015; Vaswani and Sagae, 2016). More generally, training greedy search systems by paying attention to the expected classifier behavior during test time has been explored under the imitation learning and learning-to-search frameworks (Abbeel and Ng, 2004; Daumé III and Marcu, 2005; Vlachos, 2012; He et al., 2012; Daumé III et al., 2009; Ross et al., 2011; Chang et al., 2015). Directly modeling the probability of making a mistake has also been explored for parsing (Yazdani and Henderson, 2015). Generally, the use of RNNs to conditionally predict actions in sequence given a history is spurring increased interest in training regimens that make the learned model more robust to test-time prediction errors. Solutions based on curriculum learning (Bengio et al., 2015), expected loss training (Shen et al., 2015), and reinforcement learning have been proposed (Ranzato et al., 2016). Finally, abandoning greedy search in favor of approximate global search offers an alternative solution to the problems with greedy search (Andor et al., 2016), and has been analyzed as well (Kulesza and Pereira, 2007; Finley and Joachims, 2008), including for parsing (Martins et al., 2009).

5 Conclusions

Dyer et al. (2015) presented stack LSTMs and used them to implement a transition-based dependency parser. The parser uses a greedy learning strategy which potentially provides very high parsing speed while still achieving state-of-the-art results. We have demonstrated that improvement by training the greedy parser on non-gold outcomes; dynamic

[5]We report the performance of these parsers in the most comparable setup, that is, with beam size 1 or greedy search.

oracles improve the stack LSTM parser, achieving 93.56 UAS for English, maintaining greedy search.

Acknowledgments

This work was sponsored in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533, and in part by NSF CAREER grant IIS-1054319. Miguel Ballesteros was supported by the European Commission under the contract numbers FP7-ICT-610411 (project MULTISENSOR) and H2020-RIA-645012 (project KRISTINA). Yoav Goldberg is supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI), a Google Research Award and the Israeli Science Foundation (grant number 1555/15).

References

Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proc. of ICML*.

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 *Proc. of ACL*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. arXiv:1506.03099.

Anders Björkelund and Joakim Nivre. 2015. Non-deterministic oracles for unrestricted non-projective transition-based dependency parsing. In *Proc. of IWPT*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. 2015. Learning to search better than your teacher. In *Proc. of ICML*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. of ACL*.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proc. of ICML*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75:297–325.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

T. Finley and T. Joachims. 2008. Training structural SVMs when exact inference is intractable. In *In Proc. of ICML*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proc. of COLING*.

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

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *Transactions of the association for Computational Linguistics*, 2.

Yoav Goldberg. 2013. Dynamic-oracle transition-based parsing with calibrated probabilistic output. In *Proc. of IWPT*.

Carlos Gómez-Rodríguez and Daniel Fernández-González. 2015. An efficient dynamic oracle for unrestricted non-projective parsing. In *Proc. of ACL*.

Carlos Gómez-Rodríguez, Francesco Sartorio, and Giorgio Satta. 2014. A polynomial-time dynamic oracle for non-projective dependency parsing. In *Proc. of EMNLP*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proc. of CoNLL*.

He He, Hal Daumé III, and Jason Eisner. 2012. Imitation learning by coaching. In *NIPS*.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142.

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. A non-monotonic arc-eager transition system for dependency parsing. In *Proc. of CoNLL*.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proc. of ACL*.

A. Kulesza and F. Pereira. 2007. Structured learning with approximate inference. In *NIPS*.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Polyhedral outer approximations with application to natural language parsing. In *Proc. of ICML*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT*.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*.

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

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proc. of ICLR*.

Stefan Riezler, Detlef Prescher, Jonas Kuhn, and Mark Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and em training. In *Proc. of ACL*.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proc. of AISTAT*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. In *Proc. of ACL*.

Alper Tokgöz and Gülşen Eryiğit. 2015. Transition-based dependency DAG parsing using dynamic oracles. In *Proc. of ACL SRW*.

Ashish Vaswani and Kenji Sagae. 2016. Efficient structured inference for transition-based parsing with neural networks and error states. *Transactions of the Association for Computational Linguistics*, 4:183–196.

Andreas Vlachos. 2012. An investigation of imitation learning algorithms for structured prediction. In *Proc. of the European Workshop on Reinforcement Learning*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.

Majid Yazdani and James Henderson. 2015. Incremental recurrent neural network dependency parser with search-based discriminative training. In *Proc. of CoNLL*.