Self-Training for Unsupervised Parsing with PRPN

Anhad Mohananey[1*†] **Katharina Kann**[2*] **Samuel R. Bowman**[1]
[1]New York University
[2]University of Colorado Boulder
{anhad,bowman}@nyu.edu
katharina.kann@colorado.edu

Abstract

Neural unsupervised parsing (UP) models learn to parse without access to syntactic annotations, while being optimized for another task like language modeling. In this work, we propose *self-training* for neural UP models: we leverage aggregated annotations predicted by copies of our model as supervision for future copies. To be able to use our model's predictions during training, we extend a recent neural UP architecture, the PRPN (Shen et al., 2018a), such that it can be trained in a semi-supervised fashion. We then add examples with parses predicted by our model to our unlabeled UP training data. Our self-trained model outperforms the PRPN by 8.1% F1 and the previous state of the art by 1.6% F1. In addition, we show that our architecture can also be helpful for semi-supervised parsing in ultra-low-resource settings.

1 Introduction

Unsupervised parsing (UP) models learn to parse sentences into unlabeled constituency trees without the need for annotated treebanks. Self-training (Yarowsky, 1995; Riloff et al., 2003) consists of training a model, using it to label new examples and, based on a confidence metric, adding a subset to the training set, before repeating training. For supervised parsing, results with self-training have been mixed (Charniak, 1997; Steedman et al., 2003; McClosky D, 2006). For unsupervised dependency parsing, Le and Zuidema (2015) obtain strong results by training a supervised parser on outputs of unsupervised parsing. UP models show low self-agreement between training runs (Kim et al., 2019a), while obtaining parsing performances far above chance. Supervising one run with confident parses from the last could combine their individual strengths. Thus, we ask the question: *Can UP benefit from self-training?*

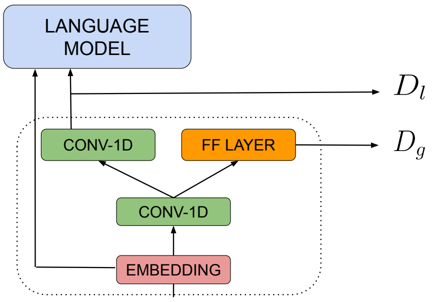

Figure 1: Our parser, represented by the dotted box, outputs syntactic distances D_g and D_l. Both D_g and D_l can be supervised, but D_l can also be learned in a latent manner.

In order to answer this question, we propose SS-PRPN, a semi-supervised extension of the UP architecture PRPN (Shen et al., 2018a), which can be trained jointly on language modeling and supervised parsing. This enables our model to leverage silver-standard annotations obtained via self-training for supervision. Our approach draws on the idea of *syntactic distances*, which can be learned both as latent variables (Shen et al., 2018a) and as explicit supervision targets (Shen et al., 2018b). We use both of these, leveraging annotations obtained via UP to supervise the two different outputs of the parser, in addition to standard UP training.

SS-PRPN, in combination with self-training, improves over its original version by 8.1% F1 and over the previous state of the art (Kim et al., 2019a) by 1.6% F1, when trained and evaluated on the English PTB (Marcus et al., 1999): *UP can indeed benefit from self-training.* We further perform an analysis of our self-training procedure, finding that longer sentences benefit most from self-training.

Although our primary motivation for the de-

[*] Equal contribution.
[†] Now at Electronic Arts.

Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task, pages 104–109
Virtual Meeting, July 9, 2020. ©2020 Association for Computational Linguistics
https://doi.org/10.18653/v1/P17

velopment of a semi-supervised architecture is to enable self-training, we further hypothesize that, since language modeling and parsing annotations seem to provide complementary information, UP should aid low-resource supervised parsing. As a proof of concept, we employ SS-PRPN for semi-supervised training. In extremely-low-data regimes with no more than 250 labeled parses, SS-PRPN outperforms supervised and unsupervised baselines in most settings on unlabeled parsing, and in all settings on labeled constituency parsing.

Related Work Following the line of research on non-neural UP models (Clark, 2001; Klein and Manning, 2002; Bod, 2006), early approaches to neural UP (Yogatama et al., 2017; Choi et al., 2018) obtain improved performance on downstream tasks, yet show highly inconsistent behavior in parsing (Williams et al., 2018).

Recently, Shen et al. (2018a) introduce the first high performing neural UP model (Htut et al., 2018). Dyer et al. (2019) raise concerns that PRPN's parsing methodology is biased towards English trees. Though these concerns are serious, they are largely orthogonal to our research question regarding the helpfulness of self-training for UP.

Several models have been introduced since: Shen et al. (2019) propose an architecture consisting of an LSTM (Hochreiter and Schmidhuber, 1997) with a modified update function for the LSTM cell state, Kim et al. (2019a)—the current state-of-the-art—introduce a model based on a mixture of probabilistic context-free grammars, Kim et al. (2019b) present unsupervised learning of recurrent neural networks grammars, Li et al. (2019) combine PRPN with imitation learning, and Drozdov et al. (2019) employ a recursive autoencoder. Kim et al. (2020) examine tree induction from pretrained models.

2 Model

Syntactic Distances In order to parse a sentence, a computational model needs to output some kind of variables representing a unique tree structure. The variables we use are *syntactic distances* as introduced by Shen et al. (2018a). They represent the syntactic relationships between all successive pairs of words in a sentence. If the distance between two neighboring words is large, they belong to different subtrees, and, thus, their traversal distance in the tree is large. A parse tree can be created by finding the maximum syntactic distance, splitting the sentence into sub-trees there, and repeating this process recursively for each sub-tree until a single token is left.

Algorithm 1: Tree to latent distances D_l

```
D_l ← [1] * leaves_tree        ▷ leaves_tree :leaf count of tree
b ← 0
max ← 100                      ▷ max: max possible depth of tree
Function DISTANCE (tree, b, max)
    DISTANCE (tree_l, b, max-1)
    x ← tree_r                 ▷ tree_r: right child of tree
    while True do
        if x_l is empty then
            D_l[b + leaves_{tree_l}] ← max
            break
        end
        x ← x_l                ▷ x_l: left child of x
    end
```

Two different formulations of syntactic distance have been proposed to realize this basic intuition: The first, which we refer to as D_l, is introduced by Shen et al. (2018a) as a latent variable in their UP model. Since, for self-training, we supervise D_l with values predicted by our model, we introduce Algorithm 1, which is used to convert a tree to distances D_l. The second kind of distance, denoted here as D_g, is introduced by Shen et al. (2018b) as labels for direct supervision. We use their algorithms to map trees to distances D_g and vice versa, and ask readers to refer to Shen et al. (2018b) for details.

We design our parser in such a way that it can predict both. We treat the decision whether D_g or D_l are used at test time as a hyperparameter. The reasons why we employ both types of distances are two-fold: D_g, unlike D_l, cannot be learned in an unsupervised fashion, which is critical for a semi-supervised architecture. Empirically, supervising purely on D_l performs poorly.

The Parser Our parser, cf. Figure 1, consists of an embedding layer and a convolutional layer which are followed by two different components: a linear output layer that predicts D_g and a second convolutional layer that predicts D_l.

Formally, given an input sentence $s = t_0, t_1, \ldots, t_{n-1}$, our parser predicts D_g as:

$$h_i = ReLU(W_c \begin{bmatrix} t_{i-L_1} \\ t_{i-L_1+1} \\ \cdots \\ t_i \end{bmatrix} + b_c) \qquad (1)$$

$$d_i = ReLU(W_d h_i + b_d) \qquad (2)$$

Algorithm 2: Self-training for UP

```
Unlabeled data $X_U$
Training set $X_T \leftarrow \emptyset$
Train $n_c$ UP models on $X_U$
for $s_i \in X_U$ do
    $n_a \leftarrow$ number of models agreeing on parse $p(s_i)$
    if $n_a \geq \mu n_c$ then
        $X_T \leftarrow X_T \cup p(s_i)$    ▷ add confident parse
    end
end
Train model on $X_U$ and $X_T$
```

where W_c are the weights of the first convolutional layer, W_d are the weights of the output layer corresponding to D_g, and b_c and b_d are bias vectors. L_1 is the filter size. D_l involves similar computations, but is the output of the second layer.

Distance Loss When we have silver-standard annotations from self-training available, we compute the loss for both syntactic distances directly. Since the relative ranking between distances—rather than absolute values—defines the tree structure, we train our parser with a hinge ranking loss following Shen et al. (2018b). Our distance loss L_r is the weighted sum of the distance losses corresponding to D_l (L_{sl}) and D_g (L_{sg}):

$$L_r = \alpha L_{sg} + (1 - \alpha) L_{sl} \tag{3}$$

Language Modeling Loss In order to optimize the parameters of our parser without direct supervision, we further feed its output—the predictions for D_l—into a language model, following Shen et al. (2018a).

Multi-Task Training Our parser is trained in a semi-supervised fashion with losses corresponding to (i) learning the distances in a latent manner through *language modeling*, and (ii) supervising directly on *distances*. We sample batches from both objectives at random.

Self-Training For self-training, cf. Algorithm 2, we first train n_c models on the unlabeled PTB training set X_U. We then have them predict parse trees for all sentences in X_U. If more than $\mu * n_c$ models (with μ as a hyperparameters) agree on the same parse, we add it as a silver-standard *labeled* example to the parsing training set X_T. We use Algorithm 1 and the respective algorithm by Shen et al. (2018b) to convert consensus trees into distances D_l and D_g. We then train a new model on both X_U and X_T.

Model	F1(μ)
PRPN	39.8 (5.6)
PRPN (ours)	46.3 (6.3)
C-PCFG	52.8 (3.8)
URNNG	44.8 (4.1)
SS-PRPN	**54.4 (0.6)**
Left Branching (LB)	13.1
Right Branching (RB)	16.5
Random	21.4

Table 1: Results on the English PTB test set, with the model tuned on the dev set. LB, RB and Random baselines are taken as-is from Htut et al. (2018). Since evaluation of C-PCFG, PRPN and URNNG is done against binary gold trees, results might differ from the original papers.

3 Experimental Design

Data and Metrics We experiment on the English Penn Treebank Marcus et al. (PTB; 1999). For evaluation, we compute the F1 score of the output parses against binarized gold parses following Williams et al. (2018). The code for our model is published online[1].

Baselines We compare against an unsupervised recurrent neural network grammar (URNNG; Kim et al., 2019b), a compound probabilistic context free grammar (C-PCFG; Kim et al., 2019a), and Shen et al. (2018a)'s PRPN. We re-implement and tune PRPN in our code base.

Hyperparameters We tune our hyperparameters on the development set. Hidden states and word embeddings have 300 and 100 dimensions, respectively. We set the weight $\alpha = 0.5$. For self-training, we obtain best results with $\mu = 60\%$ and $n_c = 15$. We further experiment with converting either D_l or D_g into final parse trees, and find that D_l works best.

4 Results and Analysis

Unsupervised Parsing Performance Table 1 shows our results. SS-PRPN outperforms all baselines: our model obtains a 1.6% higher F1 score than the strongest baseline. It further improves substantially over comparable non-self-trained baselines: by 14.6% over PRPN and by 8.1% over our reimplementation of it. SS-PRPN also shows a much lower variance. This demonstrates that self-training is indeed a viable approach for UP.

[1] https://github.com/anhad13/SelfTrainingAndLRP

	Av. Length	Av. Depth	Av. F1	#sents
Self-training	7.0	3.3	82.2	1897
PTB gold	20.9	10.6	100.0	39701

Table 2: Statistics of our best self-training annotations compared to PTB.

Length	0 - 10	10 - 20	20 - 30	30 - 40	> 40
Ex.	115	573	613	295	94
% Ex. improved	20%	36.8%	49.7%	52.8%	55.3%

Table 3: Percentage of development examples improved by SS-PRPN in comparison to PRPN, listed by sentence length.

Analysis of Self-Training We interpret agreement rate as our confidence value for self-training, with the hypothesis that, as agreement among models increases, there is a higher likelihood that the parse is correct. In Figure 2, we show that, as expected, the F1 score increases as more models agree, for the best self-training run (15 individual models, or the second last row in Table 1).

Additionally, Figure 2 and Table 2 show that self-training annotations consist of shorter sentences and shallower trees than our dataset's average, i.e., mostly of easier sentences.

Our final hypothesis is that self-training helps mostly for longer sentences, since models often agree on shorter ones anyways and, trivially, longer sentences leave more room for error. Table 3 shows the development set performance and the number of examples for varying sentence lengths. As expected, self-training yields the greatest gains for longer sentences.

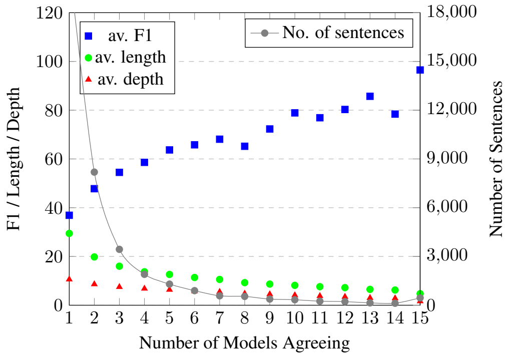

Figure 2: Statistics for self-training ($n_c = 15$): As agreement among UP models goes up, parsing F1 improves, and average depth and length go down.

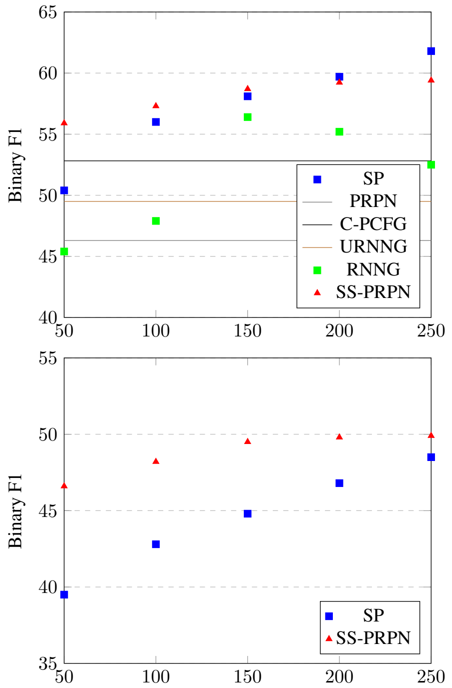

Figure 3: Low-resource parsing on the PTB. The first and second plots show unlabeled and labeled F1 respectively, plotted against the training data size.

Low-Resource Parsing Performance We further investigate how SS-PRPN performs when limited gold parses are available in addition to unlabeled data. To predict constituency labels, we add and train an additional linear output layer after the first convolutional layer. We find that, on the development set, converting D_g into parse trees works better for low-resource parsing than D_l. As supervised baselines, we employ Dyer et al. (2016)'s recurrent neural network grammar (RNNG) and a supervised parser (SP) based on syntactic distances (Shen et al., 2018b). Figure 3 shows results for 50 to 250 annotated examples. The upper part shows the *unlabeled* parsing performance in comparison to the UP baselines. We outperform all baselines for 50 to 150 examples, while SP performs slightly better with more annotations. When looking at *labeled* F1 in the lower part of Figure 3, SS-PRPN clearly outperforms SP, which indicates that unlabeled data can be leveraged in the low-resource setting.

5 Conclusion

We introduce a semi-supervised neural architecture, SS-PRPN, which is capable of UP via self-training. Our self-trained models strongly outperform comparable baselines, and advance the state of the art on PTB by 1.6% F1. Analyses show that our approach yields most gains for longer sentences. Our architecture can also leverage limited amounts of parsing supervision when available. We conclude that it is beneficial to develop better UP models for semi-supervised settings.

6 Acknowledgements

This work has benefited from support of Samsung Research through the project *Improving Deep Learning using Latent Structure* and the donation of Titan V GPU by NVIDIA Corporation.

References

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. *AAAI/IAAI*, 2005.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 workshop on Computational Natural Language Learning*. Association for Computational Linguistics.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Chris Dyer, Gábor Melis, and Phil Blunsom. 2019. A critical analysis of biased parsers in unsupervised parsing. *arXiv:1909.09428*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang-goo Lee. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *ICLR*.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019a. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of Association for Computational Linguistics*. Association for Computational Linguistics.

Yoon Kim, Alexander M Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Dan Klein and Christopher D Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of Association for Computational Linguistics*. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. *arXiv preprint arXiv:1504.04666*.

Bowen Li, Lili Mou, and Frank Keller. 2019. An imitation learning approach to unsupervised parsing. *arXiv:1906.02276*.

Mitchell Marcus et al. 1999. Treebank-3 ldc99t42 web download. *Philidelphia: Linguistic Data Consortium*.

Johnson M McClosky D, Charniak E. 2006. Effective self-training for parsing. *North American Chapter of the Association of Computational Linguistics*.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning Subjective Nouns using Extraction Pattern Bootstrapping. In *Proceedings of CoNLL*.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2018a. Neural language modeling by jointly learning syntax and lexicon. In *ICLR*.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018b. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the Association for Computational Linguistics*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *ICLR*.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Adina Williams, Andrew Drozdov, and Samuel R Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*.

David Yarowsky. 1995. Unsupervised Word-Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of ACL.*

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *ICLR*.