On the Role of Supervision in Unsupervised Constituency Parsing

Haoyue Shi **Karen Livescu** **Kevin Gimpel**
Toyota Technological Institute at Chicago, IL, USA, 60637
{freda,klivescu,kgimpel}@ttic.edu

Abstract

We analyze several recent unsupervised constituency parsing models, which are tuned with respect to the parsing F_1 score on the *Wall Street Journal* (WSJ) development set (1,700 sentences). We introduce strong baselines for them, by training an existing supervised parsing model (Kitaev and Klein, 2018) on the same labeled examples they access. When training on the 1,700 examples, or even when using only 50 examples for training and 5 for development, such a few-shot parsing approach can outperform all the unsupervised parsing methods by a significant margin. Few-shot parsing can be further improved by a simple data augmentation method and self-training. This suggests that, in order to arrive at fair conclusions, we should carefully consider the amount of labeled data used for model development. We propose two protocols for future work on unsupervised parsing: (i) use fully unsupervised criteria for hyperparameter tuning and model selection; (ii) use as few labeled examples as possible for model development, and compare to few-shot parsing trained on the same labeled examples.[1]

1 Introduction

Recent work has considered neural unsupervised constituency parsing (Shen et al., 2018a; Drozdov et al., 2019; Kim et al., 2019b, *inter alia*), showing that it can achieve much better performance than trivial baselines. However, many of these approaches use the gold parse trees of all sentences in a development set for either early stopping (Shen et al., 2018a, 2019; Drozdov et al., 2019, *inter alia*) or hyperparameter tuning (Kim et al., 2019a). In contrast, models trained and tuned without any labeled data (Kim et al., 2019b; Peng et al., 2019) are much less competitive.

Are the labeled examples important in order to obtain decent unsupervised parsing performance? How well can we do if we *train* on these labeled examples rather than merely using them for tuning? In this work, we consider training a supervised constituency parsing model (Kitaev and Klein, 2018) with very few examples as a strong baseline for unsupervised parsing tuned on labeled examples.

We empirically characterize unsupervised and few-shot parsing across the spectrum of labeled data availability, finding that (i) tuning based on a few (as few as 15) labeled examples is sufficient to improve unsupervised parsers over fully unsupervised criteria by a significant margin; (ii) unsupervised parsing with supervised tuning does outperform few-shot parsing with fewer than 15 labeled examples, but few-shot parsing quickly dominates once there are more than 55 examples; and (iii) when few-shot parsing is combined with a simple data augmentation method and self-training (Steedman et al., 2003; Reichart and Rappoport, 2007; McClosky et al., 2006, *inter alia*), only 15 examples are needed for few-shot parsing to begin to dominate.

Based on these results, we propose the following two protocols for future work on unsupervised parsing:

1. Derive and use fully unsupervised criteria for hyperparameter tuning and model selection.
2. Use as few labeled examples as possible for model development and tuning, and compare to few-shot parsing models trained on the used examples as a strong baseline.

We suggest future work to tune and compare models under each protocol separately.

In addition, we present two side findings on unsupervised parsing: (i) the vocabulary size in unsupervised parsing, which has not been widely considered as a hyperparameter and varies across

[1] Project page: https://ttic.uchicago.edu/~freda/project/rsucp/

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 7611–7621,
November 16–20, 2020.

prior work, greatly affects the performance of all unsupervised parsing models tested; and (ii) self-training can help improve all investigated unsupervised parsing (Shen et al., 2018a, 2019; Drozdov et al., 2019; Kim et al., 2019a) and few-shot parsing models, and thus can be considered as a post-processing step in future work.

2 Related Work

Unsupervised parsing. During the past two decades, there has been a lot of work on both unsupervised constituency parsing (Klein and Manning, 2002, 2004; Bod, 2006a,b; Seginer, 2007; Snyder et al., 2009, *inter alia*) and unsupervised dependency parsing (Klein and Manning, 2004; Smith and Eisner, 2006; Spitkovsky et al., 2011, 2013, *inter alia*). Recent work has proposed several effective models for unsupervised or distantly supervised constituency parsing, optimizing either a language modeling objective (Shen et al., 2018a, 2019; Kim et al., 2019b,a, *inter alia*) or other downstream semantic objectives (Li et al., 2019; Shi et al., 2019). Some of them are tuned with labeled examples in the WSJ development set (Shen et al., 2018a, 2019; Htut et al., 2018; Drozdov et al., 2019; Kim et al., 2019a; Wang et al., 2019) or other labeled examples (Jin et al., 2018, 2019).

Data augmentation. Data augmentation is a strategy for automatically increasing the amount and variety of data for training models, without actually collecting any new data. Such methods have been found helpful on many NLP tasks, including text classification (Kobayashi, 2018; Samanta et al., 2019), relation classification (Xu et al., 2016), and part-of-speech tagging (Şahin and Steedman, 2018). Part of our approach also falls into the category of data augmentation, applied specifically to constituency parsing from very few examples.

Few-shot parsing. Sagae et al. (2008) show that a supervised dependency parsing model trained on 100 examples can work surprisingly well. Recent work has demonstrated the potential of few-shot dependency parsing on multiple languages (Aufrant et al., 2018; Meechan-Maddon and Nivre, 2019; Vania et al., 2019, *inter alia*). Our approach (§3) can be viewed as few-shot constituency parsing.

3 Few-Shot Constituency Parsing

We apply Benepar (§3.1; Kitaev and Klein, 2018) as the base model for few-shot parsing. We present a simple data augmentation method (§3.2) and an iterative self-training strategy (§3.3) to further improve the performance. We suggest that such an approach should serve as a strong baseline for unsupervised parsing with supervised tuning.

3.1 Parsing Model

The Benepar parsing model consists of (i) word embeddings, (ii) transformer–based (Vaswani et al., 2017) word-span embeddings, and (iii) a multi-layer perceptron to compute a score for each labeled span.[2] The score of an arbitrary tree is defined as the sum of all of its internal span scores. Given a sentence and its ground-truth parse tree T^*, the model is trained to satisfy $\mathit{score}(T^*) \geq \mathit{score}(T) + \Delta(T^*, T)$ for any tree T $(T \neq T^*)$, where Δ denotes the Hamming loss on labeled spans. The label-aware CKY algorithm is used to obtain the tree with the highest score. More details can be found in Kitaev and Klein (2018).

3.2 Data Augmentation

We introduce a data augmentation method, subtree substitution (SUB; Figure 1), to automatically improve the diversity of data in the few-shot setting.

We start with a set of sentences with N unlabeled parse trees $\mathcal{S} = \{\langle s_i, \mathcal{T}_i \rangle\}_{i=1}^{N}$; $s_i = \langle w_{i1}, w_{i2}, \ldots, w_{iL_i} \rangle$ denotes a sentence with L_i words, where w_{ik} denotes a word; $\mathcal{T}_i = \{\langle b_{ij}, e_{ij} \rangle\}_{j=1}^{C_i}$ denotes the unlabeled parse tree of s_i with C_i nonterminal nodes; b_{ij} and e_{ij} denotes the beginning and ending index of a constituent.

The augmented dataset $\mathcal{S}'$ is initialized to $\mathcal{S}$. At each step, we draw a sentence s_i and its parse tree $\mathcal{T}_i$ uniformly from $\mathcal{S}'$, and draw a constituent $\langle b_{ij}, e_{ij} \rangle \in \mathcal{T}_i$ uniformly from $\mathcal{T}_i$. After that, we replace $\langle b_{ij}, e_{ij} \rangle$ with a random $\langle b_{kh}, e_{kh} \rangle \in t_k$; that is, we replace a constituent with another one from the training set. We let s'_i and $\mathcal{T}'_i$ denote modified sentence and its parse tree, assign $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(s'_i, \mathcal{T}'_i)\}$, and repeat the above procedure until $\mathcal{S}'$ reaches the desired size.

3.3 Self-Training

Steedman et al. (2003), Reichart and Rappoport (2007) and McClosky et al. (2006) have shown that

[2] In this work, there are only two labels: (i) NT denotes a constituent and (ii) $\emptyset$ denotes non-constituent. The label $\emptyset$ enables the parser to output non-binary trees; details can be found in Kitaev and Klein (2018). Almost all existing unsupervised parsing models do not use the nonterminal categories in the development set, so we propose to train such unlabeled constituency parsing models as their baselines.

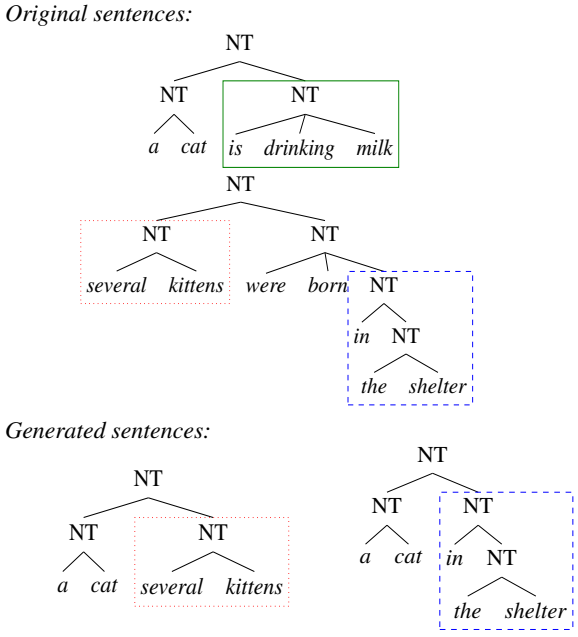

Figure 1: Illustration of the proposed data augmentation approach for improving few-shot parsing: we create new sentences by subtree substitution (e.g., substituting the subtree in the solid box by the ones in the dotted or dashed box), whether the created sentences are grammatical or not. NT denotes nonterminal nodes.

self-training (ST) on unseen sentences can improve a parsing model. Inspired by this, we apply an iterative self-training strategy after obtaining each supervised or unsupervised parsing model.

Concretely, we start with an arbitrary parsing model $\mathcal{M}_0$. At the i^{th} step of self-training, we (i) use the trained model from the previous step (i.e., $\mathcal{M}_{i-1}$) to predict parse trees for sentences in the WSJ training set and those in the WSJ development set, and (ii) train a supervised parsing model $\mathcal{M}_i$ (Kitaev and Klein, 2018) to fit the prediction of $\mathcal{M}_{i-1}$. No gold labels are used in self-training.

4 Experiments

4.1 Dataset and Training Details

We use the WSJ portion of the Penn Treebank corpus (Marcus et al., 1993) to train and evaluate the models, replace all number tokens with a special token, and split standard train/dev/test sets following Kim et al. (2019b).[3] For each criterion, we tune the hyperparameters of each model with respect to its performance on the development set. To solve the problem of vocabulary sparsity in the few-shot parsing setting (§3), we initialize the word embeddings of Benepar (Kitaev and Klein, 2018) with the word embeddings from an LSTM–based (Hochreiter and Schmidhuber, 1997) language model trained on the WSJ training set. During training, models are able to access all sentences (without parse trees) in the WSJ training set; for few-shot parsing or unsupervised parsing with supervised tuning, some unlabeled parse trees in the WSJ development set are available as well. We augment the training set to 10,000 examples for few-shot parsing with SUB, and apply 5-step self-training when applicable.

We evaluate the unlabeled F_1 score of all models using `evalb`,[4] discarding punctuation. More details can be found in the supplementary material.

4.2 Models and Tuning Criteria

We investigate four recently proposed models: PRPN (Shen et al., 2018a), ON-LSTM (Shen et al., 2019), DIORA (Drozdov et al., 2019), and Compound PCFG (Kim et al., 2019a).

PRPN and ON-LSTM are left-to-right neural language models, where syntactic distance (Shen et al., 2018b) between consecutive words is computed from the model output and used to infer the constituency parse tree. DIORA learns text-span representations and span-level scores by optimizing a masked language modeling objective. The compound PCFG uses a neural parameterization of a PCFG, as well as a per-sentence latent vector which introduces context sensitivity. Both DIORA and the Compound PCFG use the CKY algorithm to infer the parse tree of a given sentence.

As fully unsupervised tuning criteria, we use perplexity on the development set for PRPN and ON-LSTM, and the upper bound of perplexity for the Compound PCFG, following Shen et al. (2018a, 2019) and Kim et al. (2019a) respectively. For DIORA, we use its reconstruction loss on the development set.[5]

4.3 Comparison between Unsupervised Parsing and Few-Shot Parsing

We compare unsupervised parsing against few-shot parsing (Table 1 and Figure 2): when there are 55 or more labeled examples available, few-shot

[3] For analysis purposes (§4.5 and Figure 2), we use WSJ Section 24, instead of the standard development set (Section 22) as we train few-shot parsing on part of it. We do not use the standard test split (Section 23) to avoid tuning on the test set, hence our analysis numbers are not directly comparable with those reported in original papers.

[4] https://nlp.cs.nyu.edu/evalb/

[5] Drozdov et al. (2019) did not evaluate any unsupervised tuning criteria for DIORA. We choose reconstruction loss because it is what DIORA minimizes during training.

Model $\backslash \lvert\mathcal{D}_{\textit{label}}\rvert$	0	15	55	1,700
PRPN	42.4	44.6	44.7	44.9
DIORA	**46.6**	47.7	47.6	48.0
Compound PCFG	39.2	39.2	39.2	39.2
ON-LSTM	39.0	**51.5**	**51.1**	**52.0**
Few-Shot	N/A	42.1	55.5	81.2
Few-Shot + SUB	N/A	52.5	58.5	82.6
Few-Shot + SUB + ST	N/A	**53.4**	**61.2**	**85.1**

Table 1: Unlabeled F_1 scores on the standard WSJ test set (Section 23). We keep all tokens, resulting in a vocabulary size $|V| = 35K$. $|\mathcal{D}_{\textit{label}}| = 0$ means using fully unsupervised criteria (§4.2); otherwise we use the first $|\mathcal{D}_{\textit{label}}|$ labeled examples in WSJ Section 22. For few-shot parsing (§3), we divide the available labeled examples into 10/5, 50/5, and 1,600/100 respectively for training and development. We use boldface for the best unsupervised parsing result and the best few-shot parsing result in each column.

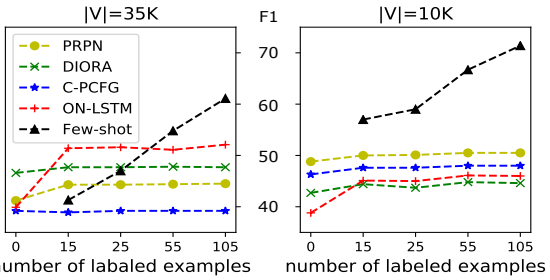

Figure 2: Performance of models with vocabulary size 35K (left) and 10K (right) on WSJ Section 24. C-PCFG denotes the Compound PCFG. The F_1 scores are averaged over 5 runs with the same hyperparameters, different random seeds, and different sets of labeled examples when applicable.

Model	$\text{WSJ}_{\textit{train}}$	+ $\text{WSJ}_{\textit{dev}}$ SUB
PRPN	44.9	**46.1**
DIORA	48.0	**48.2**
Compound PCFG	39.2	**42.2**
ON-LSTM	**52.0**	48.2

Table 2: Unlabeled F_1 scores on the standard WSJ test set. $\text{WSJ}_{\textit{train}}$ denotes models trained with the 40K sentences in the WSJ training set, and + $\text{WSJ}_{\textit{dev}}$ SUB denotes models trained with the union of WSJ training sentences and 10K sentences augmented from 1,700 WSJ development sentences. The best number in each row is bolded.

parsing (§3) consistently outperforms all unsupervised parsing models; with SUB and self-training or a smaller vocabulary size ($|V| = 10K$), few-shot parsing begins to dominate even when only 15 labeled examples are available.

On the other hand, we find that a few labeled examples are consistently helpful for most models to achieve better results than fully unsupervised parsing. In addition, models tuned on a very small number (e.g., 15) of labeled examples can achieve similar performance to those tuned on 1,700 labeled examples; that is, we need far fewer labeled examples than existing unsupervised parsing approaches have used to obtain very similar results.

To test if SUB can also help improve unsupervised parsing models, we generate 10K sentences from the 1,700 sentences in the WSJ development set with SUB (Figure 1), and add them to the 40K-sentence WSJ training set. We compare unsupervised parsing models trained on the original WSJ training set and the augmented one (Table 2). We find that SUB can sometimes help, but not by a large margin, and all numbers in Table 2 are far below the performance of few-shot parsing with the same data availability (82.6; Table 1). Few-shot parsing with data augmentation is a strong baseline for unsupervised parsing with data augmentation.

4.4 The Importance of Vocabulary Size

We notice that the result of the Compound PCFG in Table 1 is much worse than that reported by Kim et al. (2019a).[6] The only major difference between their approach and ours is the vocabulary size: instead of keeping all words, they keep the most frequent 10K words in the WSJ corpus and replace others with a special token. To analyze the importance of this choice, we compare the performance of the models with vocabulary size 35K vs. 10K (Figure 2), tuning models separately in the two settings. We find that the vocabulary size, which has not been widely considered a hyperparameter and varies across prior work, greatly affects the performance of all models tested. One possible reason is that a large portion (79.9%) of the low-frequency (i.e., outside the 10K vocabulary) word tokens are nouns or adjectives – some models (e.g., PRPN and Compound PCFG) may benefit from collapsing these tokens to a single form, as it may be a beneficial kind of word clustering. This suggests that we should consider tuning the vocabulary size

[6] Our DIORA result also differs from that reported by Drozdov et al. (2019); however, our number is not directly comparable to theirs due to different data settings—they use a different training set and apply ELMo (Peters et al., 2018) for model initialization.

Model	#ST-steps		
	0	1	5
PRPN	44.7	44.7	45.1
DIORA	46.7	48.7	49.1
Compound PCFG	41.1	41.8	42.2
ON-LSTM	50.2	51.3	52.1
Few-Shot	44.3	44.5	45.0
Few-Shot + SUB	53.3	55.5	56.6

Table 3: F_1 score on WSJ Section 24 of different models, where the base models are those used to report results in Table 1 with $|\mathcal{D}_{label}| = 15$.

as a hyperparameter, or fix the vocabulary size for fair comparison in future work.

4.5 Self-Training Improves all Models

Inspired by the fact that self-training boosts the performance of few-shot parsing (Table 1), we apply iterative self-training to the unsupervised parsing models as well, and find that it improves all models (Table 3).[7] It is worth noting that 5-step self-training is better than 1-step self-training for all base models we experimented with. Our results suggest that iterative (e.g., 5-step) self-training may be considered as a standard post-hoc processing step for unsupervised parsing.

5 Discussion

While many state-of-the-art unsupervised parsing models are tuned on all labeled examples in a development set (Drozdov et al., 2019; Kim et al., 2019b; Wang et al., 2019, *inter alia*), we have demonstrated that, given the same data, few-shot parsing with simple data augmentation and self-training can consistently outperform all of these models by a large margin. We suggest that one possibility for future work is to focus on fully unsupervised criteria, such as language model perplexity (Shen et al., 2018a, 2019; Kim et al., 2019b; Peng et al., 2019; Li et al., 2020) and model stability across different random seeds (Shi et al., 2019), for model selection, as discussed in unsupervised learning work (Smith and Eisner, 2005, 2006; Spitkovsky et al., 2010a,b, *inter alia*). An alternative is to use as few labeled examples in the development set as possible, and compare to few-shot parsing trained on the used examples as a strong baseline. In addition, we find that self-training is a useful post-processing step for unsupervised parsing.

[7] A similar idea and similar results have been presented by Kim et al. (2019a), where they train an RNNG (Dyer et al., 2016) to fit the prediction of unsupervised parsing models.

Our work does not necessarily imply that unsupervised parsers produce poor parses; they may be producing good parses that clash with the conventions of treebanks (Klein, 2005). If this is the case, then extrinsic evaluation of parsers in downstream tasks (Shi et al., 2018), e.g., machine translation (DeNero and Uszkoreit, 2011; Neubig et al., 2012; Gimpel and Smith, 2014), may better show the potential of unsupervised methods.

Acknowledgments

We thank Allyson Ettinger, Yoon Kim, Jiayuan Mao, Shane Settle, and Shubham Toshniwal for helpful discussions, as well as all the knowledgeable anonymous reviewers for their valuable and insightful feedback.

References

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2018. Quantifying training challenges of dependency parsers. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3191–3202, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Rens Bod. 2006a. 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*, pages 865–872, Sydney, Australia. Association for Computational Linguistics.

Rens Bod. 2006b. Unsupervised parsing with U-DOP. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 85–92, New York City. Association for Computational Linguistics.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Edinburgh, Scotland, UK. 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 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1129–1141, Minneapolis, Minnesota. Association for Computational Linguistics.

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

for Computational Linguistics: Human Language Technologies, pages 199–209, San Diego, California. Association for Computational Linguistics.

Kevin Gimpel and Noah A. Smith. 2014. Phrase dependency machine translation with quasi-synchronous tree-to-tree features. *Computational Linguistics*, 40(2):349–401.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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*, pages 4998–5003, Brussels, Belgium. Association for Computational Linguistics.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2018. Unsupervised grammar induction with depth-bounded PCFG. *Transactions of the Association for Computational Linguistics*, 6:211–224.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, Lane Schwartz, and William Schuler. 2019. Unsupervised learning of PCFGs with normalizing flow. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2442–2452, Florence, Italy. Association for Computational Linguistics.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019a. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy. Association for Computational Linguistics.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1105–1117, Minneapolis, Minnesota. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Dan Klein. 2005. *The unsupervised learning of natural language structure*. Ph.D. thesis, Stanford University.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 478–485, Barcelona, Spain.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Bowen Li, Lili Mou, and Frank Keller. 2019. An imitation learning approach to unsupervised parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3485–3492, Florence, Italy. Association for Computational Linguistics.

Jun Li, Yifan Cao, Jiong Cai, Yong Jiang, and Kewei Tu. 2020. An empirical comparison of unsupervised constituency parsing methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3278–3283, Online. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

Ailsa Meechan-Maddon and Joakim Nivre. 2019. How to parse low-resource languages: Cross-lingual parsing, target language annotation, or both? In *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pages 112–120, Paris, France. Association for Computational Linguistics.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 843–853, Jeju Island, Korea. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Hao Peng, Roy Schwartz, and Noah A. Smith. 2019. PaLM: A hybrid parser and language model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3642–3649, Hong Kong, China. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 616–623, Prague, Czech Republic. Association for Computational Linguistics.

Kenji Sagae, Yusuke Miyao, Rune Saetre, and Jun'ichi Tsujii. 2008. Evaluating the effects of treebank size in a practical application for parsing. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 14–20, Columbus, Ohio. Association for Computational Linguistics.

Gözde Gül Şahin and Mark Steedman. 2018. Data augmentation via dependency tree morphing for low-resource languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5004–5009, Brussels, Belgium. Association for Computational Linguistics.

Bidisha Samanta, Niloy Ganguly, and Soumen Chakrabarti. 2019. Improved sentiment detection via label transfer from monolingual to synthetic code-switched text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3528–3537, Florence, Italy. Association for Computational Linguistics.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391, Prague, Czech Republic. Association for Computational Linguistics.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2018a. Neural language modeling by jointly learning syntax and lexicon. In *Proceedings of the International Conference on Learning Representations*.

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 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180, Melbourne, Australia. Association for Computational Linguistics.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *Proceedings of the International Conference on Learning Representations*.

Haoyue Shi, Jiayuan Mao, Kevin Gimpel, and Karen Livescu. 2019. Visually grounded neural syntax acquisition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1842–1861, Florence, Italy. Association for Computational Linguistics.

Haoyue Shi, Hao Zhou, Jiaze Chen, and Lei Li. 2018. On tree-based neural sentence modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4631–4641, Brussels, Belgium. Association for Computational Linguistics.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, Ann Arbor, Michigan. Association for Computational Linguistics.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 569–576, Sydney, Australia. Association for Computational Linguistics.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 73–81, Suntec, Singapore. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1281–1290, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010a. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter*

of the Association for Computational Linguistics, pages 751–759, Los Angeles, California. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1983–1995, Seattle, Washington, USA. Association for Computational Linguistics.

Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17, Uppsala, Sweden. Association for Computational Linguistics.

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 *10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Clara Vania, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1105–1116, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1061–1070, Hong Kong, China. Association for Computational Linguistics.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1461–1470, Osaka, Japan. The COLING 2016 Organizing Committee.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612.

Appendices

A Training Details

A.1 Datasets and Standard Splits

We summarize the details for the dataset, i.e., the Penn Treebank (Marcus et al., 1993) in Table 4.[8] We follow Kim et al. (2019b) to preprocess the original dataset, removing all noterminals indicating syntactic movement (e.g., the -NONE- category) as well as all sub-categories of nonterminal nodes (e.g., NP-SBJ $\rightarrow$ NP).

A.2 Hyperparameter Tuning

We modify the original code to fit our experiments. All models requires PyTorch (Paszke et al., 2017) for automatic differentiation. We train all of our models except ON-LSTM with an Adam optimizer (Kingma and Ba, 2015), and use stochastic gradient descent (SGD) for ON-LSTM.[9] We make the batch size as large as possible to make efficient use of GPU memory. We use grid search (i.e., enumerate all possible combinations of hyperparameters) to tune models, where the considered hyperparameters of models and values are as follows. We evaluate on the development set after every epoch. For hyperparameter tuning, we let each model run for 10 epochs (i.e., see the training set for 10 times) and select the best group of hyperparameters with respect to each criterion (see paper for details). We tune the hyperparameters with the vocabulary size $|V| = 35K$, and we use the selected best performing hyperparameters to train models with $|V| = 10K$.

[8] https://catalog.ldc.upenn.edu/LDC99T42

[9] Shen et al. (2019) and previous work have shown that SGD leads to much better performance than Adam, in terms of language model perplexity.

Split	Sections	# Examples
dev	Section 22	39,832
test	Section 23	1,700
train	Sections 02-21	2,416
rest	Sections 00, 01, 24	1,346 (Sec. 24)

Table 4: Details of standard split for the WSJ portion of the Penn Treebank (Marcus et al., 1993). All sentences are in English. The *rest* split is intended left not to use in the standard supervised parsing process, and may be used for other purposes. We use Section 24 for our analysis.

Benepar (Kitaev and Klein, 2018).[10]

Hyperparameter	Considered Values
learning rate	$1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-5}$
hidden layer size	256, 512, 1024
number of hidden layers	2, 4, 8

hyperparameter search trials $= 4 \times 3 \times 3 = 36$.

PRPN (Shen et al., 2018a).[11]

Hyperparameter	Considered Values
learning rate	$1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}$
hidden layer size	100, 200, 400

hyperparameter search trials $= 3 \times 3 = 9$.

ON-LSTM (Shen et al., 2019).[12]

Hyperparameter	Considered Values
learning rate	1, 10, 30
hidden layer size	100, 200, 400

hyperparameter search trials $= 3 \times 3 = 9$.

We follow Shen et al. (2019) to use a 3-layer ON-LSTM, and used the master gates in the second layer to compute the syntactic distance (Shen et al., 2018b).

DIORA (Drozdov et al., 2019).[13]

Hyperparameter	Considered Values
Architecture	TreeLSTM (Tai et al., 2015; Zhu et al., 2015) MLP, MLP-shared
learning rate	$1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}$
negative ex.	5, 10, 20
hidden layer size	100, 200, 400

hyperparameter search trials $= 3 \times 3 \times 3 \times 3 = 81$.

Compound PCFG (Kim et al., 2019a).[14]

Hyperparameter	Considered Values
learning rate	$1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}$
#nonterminal	20, 30, 40
#preterminal	50, 60

hyperparameter search trials $= 3 \times 3 \times 2 = 18$.

A.3 Model Selection

All experiments are done on a Titan X GPU or a Titan RTX GPU when large GPU memory is required. After selecting the best hyperparameter set for each model under each criterion, we let the model run for at most 100 epochs and at most 96 hours.[15] We evaluate on the development set after every epoch, with respect to each (fully unsupervised or few labeled examples based) criterion. The best performing hyperparameter set is summarized in Table 5.

A.4 Run Time

We report the running time and number of epochs within the allowed running time (i.e., 96h) of each best-performing model in Table 6.

A.5 Hyperparameters for Self-Training

We use the following hyperparameters for all self-training experiments. Other possible hyperparameters are identical to the default ones. Please see the code attached for details.

Hyperparameter	Value
learning rate	5×10^{-5}
hidden layer size	1024
number of hidden layers	2

B Evaluation parameters

We report the used `evalb` parameters in Table 7. We modify the original COLLINS.prm to let it evaluate unlabeled F_1 score,[16] ignoring punctuation when testing.[17]

C Standard Deviation w.r.t. Different Small Development Sets

In the main content of the paper (Figure 2), We show the average WSJ Section 24 performance (in terms of F_1 score) of models with different random seeds and different development set (if applicable),

[10] https://github.com/nikitakit/self-attentive-parser

[11] https://github.com/yikangshen/PRPN

[12] https://github.com/yikangshen/Ordered-Neurons

[13] https://github.com/iesl/diora

[14] https://github.com/harvardnlp/compound-pcfg

[15] Practically, some settings share the same best hyperparameters, so we can save all the intermediate checkpoints and do post-hoc model selection efficiently.

[16] https://nlp.cs.nyu.edu/evalb/EVALB.tgz

[17] We keep the punctuation during training.

Model	# Available Ex.	Hyperparameters	# Params	Dev. Performance
Benepar	15	1×10^{-4}, 512, 4	6.6M	54.7
	25	1×10^{-4}, 1024, 2	6.9M	56.6
	55	5×10^{-5}, 256, 8	6.4M	65.4
	105	5×10^{-5}, 256, 4	3.3M	66.1
	1,700	5×10^{-5}, 1024, 2	6.9M	84.1
Benepar+SUB	15	5×10^{-5}, 1024, 2	6.9M	62.8
	25	5×10^{-5}, 1024, 2	6.9M	57.8
	55	5×10^{-5}, 1024, 2	6.9M	64.4
	105	5×10^{-5}, 1024, 2	6.9M	66.4
	1,700	5×10^{-5}, 1024, 2	6.9M	84.7
PRPN	0	1×10^{-3}, 400	10.8M	97.5 (ppl.)
	15	1×10^{-3}, 200	8.4M	43.5
	25	1×10^{-3}, 400	10.8M	41.9
	55	1×10^{-3}, 200	8.4M	43.4
	105	1×10^{-3}, 400	10.8M	41.7
	1,700	1×10^{-3}, 400	10.8M	44.7
ON-LSTM	0	20, 400	18.4M	69.2 (ppl.)
	15	20, 200	16.2M	52.1
	25	20, 200	16.2M	49.8
	55	30, 400	18.4M	50.1
	105	30, 400	18.4M	49.4
	1,700	30, 400	18.4M	51.9
DIORA	0	TreeLSTM, 1×10^{-3}, 3, 200	1.1M	0.15 (recons. loss.)
	15	MLP, 1×10^{-3}, 10, 200	0.5M	47.0
	25	MLP, 1×10^{-3}, 10, 200	0.5M	47.1
	55	MLP, 1×10^{-3}, 10, 200	0.5M	47.3
	105	MLP, 1×10^{-3}, 10, 200	0.5M	45.8
	1,700	MLP, 1×10^{-3}, 10, 200	0.5M	46.7
Compound PCFG	0	1×10^{-3}, 40, 60	34.7M	258.4 (ppl. upper bound)
	15	1×10^{-3}, 30, 60	34.1M	43.1
	25	1×10^{-3}, 30, 60	34.1M	45.1
	55	1×10^{-3}, 30, 60	34.1M	39.9
	105	1×10^{-3}, 30, 60	34.1M	41.2
	1,700	1×10^{-3}, 30, 60	34.1M	40.4

Table 5: The best performing sets of hyperparameters with respect to each investigated criterion, as well as corresponding validation performance (on the corresponding development set, i.e., the first several or no labeled examples in WSJ Section 22). The hyperparameter values are given by the order mentioned in A.2. # Available Ex. denotes the number of available (labeled) examples. # Params denotes number of (trainable) model parameters, estimated in the setting that the vocabulary size $|V| = 35K$. The performance without parenthesis is in terms of F_1 score; ppl. denotes language model perplexity; recons. loss denotes the reconstruction loss.

Model	# Epoch	Time
Benepar	100	0.2h
Benepar + SUB	100	4h
PRPN	100	30h
ON-LSTM	100	28h
DIORA	100	18h
Compound PCFG	12	96h

Table 6: The number of epoch and estimated run time of each model.

across 5 runs. Due to space limitation, we do not include the standard deviation plot. We show the full plot in Figure 3. All the standard deviations are less than 3.

```
DEBUG 0
MAX_ERROR 0
CUTOFF_LEN 10000
LABELED 0
DELETE_LABEL TOP
DELETE_LABEL -NONE-
DELETE_LABEL ,
DELETE_LABEL :
DELETE_LABEL ``
DELETE_LABEL ''
DELETE_LABEL .
DELETE_LABEL -LRB-
DELETE_LABEL -RRB-
```

Table 7: The hyperparameters used for `evalb`.

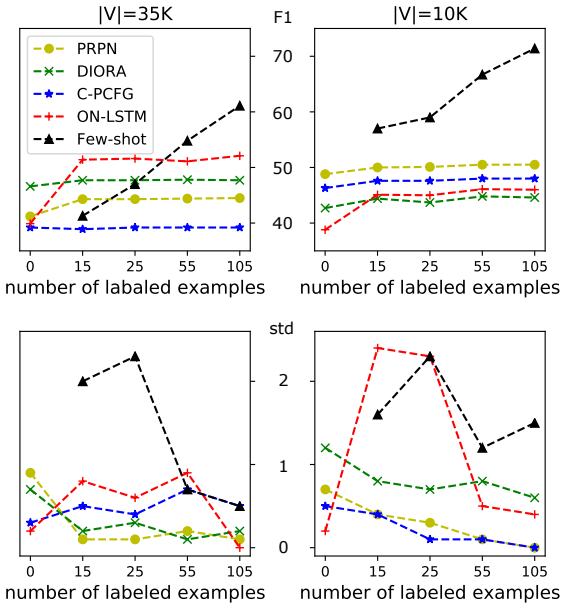

Figure 3: The average F_1 score and standard deviation, across 5 runs with the same hyperparameters (Table 5) different random seeds and different labeled development examples when applicable. The top part (F_1 score) is the same as Figure 2 in our paper.