Neural Bi-Lexicalized PCFG Induction

Songlin Yang[♣], **Yanpeng Zhao**[♢], **Kewei Tu**[♣*]
[♣]School of Information Science and Technology, ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences
[♢]ILCC, University of Edinburgh
{yangsl,tukw}@shanghaitech.edu.cn
yannzhao.ed@gmail.com

Abstract

Neural lexicalized PCFGs (L-PCFGs) (Zhu et al., 2020) have been shown effective in grammar induction. However, to reduce computational complexity, they make a strong independence assumption on the generation of the child word and thus bilexical dependencies are ignored. In this paper, we propose an approach to parameterize L-PCFGs without making implausible independence assumptions. Our approach directly models bilexical dependencies and meanwhile reduces both learning and representation complexities of L-PCFGs. Experimental results on the English WSJ dataset confirm the effectiveness of our approach in improving both running speed and unsupervised parsing performance.

1 Introduction

Probabilistic context-free grammars (PCFGs) has been an important probabilistic approach to syntactic analysis (Lari and Young, 1990; Jelinek et al., 1992). They assign a probability to each of the parses admitted by CFGs and rank them by the plausibility in such a way that the ambiguity of CFGs can be ameliorated. Still, due to the strong independence assumption of CFGs, vanilla PCFGs (Charniak, 1996) are far from adequate for highly ambiguous text.

A common premise for tackling the issue is to incorporate lexical information and weaken the independence assumption. There have been many approaches proposed under the premise (Magerman, 1995; Collins, 1997; Johnson, 1998; Klein and Manning, 2003). Among them lexicalized PCFGs (L-PCFGs) are a relatively straightforward formalism (Collins, 2003). L-PCFGs extend PCFGs by associating a word, i.e., the lexical head, with each grammar symbol. They can thus exploit lexical information to disambiguate parsing decisions and are much more expressive than vanilla PCFGs. However, they suffer from representation and inference complexities. For representation, the addition of lexical information greatly increases the number of parameters to be estimated and exacerbates the data sparsity problem during learning, so the expectation-maximisation (EM) based estimation of L-PCFGs has to rely on sophisticated smoothing techniques and factorizations (Collins, 2003). As for inference, the CYK algorithm for L-PCFGs has a $O(l^5|G|)$ complexity, where l is the sentence length and $|G|$ is the grammar constant. Although Eisner and Satta (1999) manage to reduce the complexity to $O(l^4|G|)$, inference with L-PCFGs is still relatively slow, making them less popular nowadays.

Recently, Zhu et al. (2020) combine the ideas of factorizing the binary rule probabilities (Collins, 2003) and neural parameterization (Kim et al., 2019) and propose neural L-PCFGs (NL-PCFGs), achieving good results in both unsupervised dependency and constituency parsing. Neural parameterization is the key to success, which facilitates informed smoothing (Kim et al., 2019), reduces the number of learnable parameters for large grammars (Chiu and Rush, 2020; Yang et al., 2021) and facilitates advanced gradient-based optimization techniques instead of using the traditional EM algorithm (Eisner, 2016). However, Zhu et al. (2020) oversimplify the binary rules to decrease the complexity of the inside/CYK algorithm in learning (i.e., estimating the marginal sentence log-likelihood) and inference. Specifically, they make a strong independence assumption on the generation of the child word such that it is only dependent on the nonterminal symbol. Bilexical dependencies, which have been shown useful in unsupervised dependency parsing (Han et al., 2017; Yang et al., 2020), are thus ignored.

*Corresponding Author

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 2688–2699
August 1–6, 2021.

To model bilexical dependencies and meanwhile reduce complexities, we draw inspiration from the canonical polyadic decomposition (CPD) (Kolda and Bader, 2009) and propose a latent-variable based neural parameterization of L-PCFGs. Cohen et al. (2013); Yang et al. (2021) have used CPD to decrease the complexities of PCFGs, and our work can be seen as an extension of their work to L-PCFGs. We further adopt the *unfold-refold* transformation technique (Eisner and Blatz, 2007) to decrease complexities. By using this technique, we show that the time complexity of the inside algorithm implemented by Zhu et al. (2020) can be improved from cubic to quadratic in the number of nonterminals m. The inside algorithm of our proposed method has a linear complexity in m after combining CPD and unfold-refold.

We evaluate our model on the benchmarking Wall Street Journey (WSJ) dataset. Our model surpasses the strong baseline NL-PCFG (Zhu et al., 2020) by 2.9% mean F1 and 1.3% mean UUAS under CYK decoding. When using the Minimal Bayes-Risk (MBR) decoding, our model performs even better. We provide an efficient implementation of our proposed model at `https://github.com/sustcsonglin/TN-PCFG`.

2 Background

2.1 Lexicalized CFGs

We first introduce the formalization of CFGs. A CFG is defined as a 5-tuple $\mathcal{G} = (\mathcal{S}, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R})$ where $\mathcal{S}$ is the start symbol, $\mathcal{N}$ is a finite set of nonterminal symbols, $\mathcal{P}$ is a finite set of preterminal symbols,[1] Σ is a finite set of terminal symbols, and $\mathcal{R}$ is a set of rules in the following form:

$$\begin{aligned} &S \rightarrow A && && A \in \mathcal{N} \\ &A \rightarrow BC, && A \in \mathcal{N}, && B, C \in \mathcal{N} \cup \mathcal{P} \\ &T \rightarrow w, && && T \in \mathcal{P}, w \in \Sigma \end{aligned}$$

$\mathcal{N}, \mathcal{P}$ and Σ are mutually disjoint. We will use 'nonterminals' to indicate $\mathcal{N} \cup \mathcal{P}$ when it is clear from the context.

Lexicalized CFGs (L-CFGs) (Collins, 2003) extend CFGs by associating a word with each of the nonterminals:

$$\begin{aligned} &S \rightarrow A[w_p] && A \in \mathcal{N} \\ &A[w_p] \rightarrow B[w_p]C[w_q], && A \in \mathcal{N}; B, C \in \mathcal{N} \cup \mathcal{P} \\ &A[w_p] \rightarrow C[w_q]B[w_p], && A \in \mathcal{N}; B, C \in \mathcal{N} \cup \mathcal{P} \\ &T[w_p] \rightarrow w_p, && T \in \mathcal{P} \end{aligned}$$

where $w_p, w_q \in \Sigma$ are the headwords of the constituents spanned by the associated grammar symbols, and p, q are the word positions in the sentence. We refer to A, a parent nonterminal annotated by the headword w_p, as *head-parent*. In binary rules, we refer to a child nonterminal as *head-child* if it inherits the headword of the head-parent (e.g., $B[w_p]$) and as *non-head-child* otherwise (e.g., $C[w_q]$). A head-child appears as either the left child or the right child. We denote the head direction by $D \in \{\leftharpoonup, \rightharpoonup\}$, where $\leftharpoonup$ means head-child appears as the left child.

2.2 Grammar induction with lexicalized probabilistic CFGs

Lexicalized probabilistic CFGs (L-PCFGs) extend L-CFGs by assigning each production rule $r = A \rightarrow \gamma$ a scalar π_r such that it forms a valid categorical probability distribution given the left hand side A. Note that preterminal rules always have a probability of 1 because they define a deterministic generating process.

Grammar induction with L-PCFGs follows the same way of grammar induction with PCFGs. As with PCFGs, we maximize the log-likelihood of each observed sentence $\boldsymbol{w} = w_1, \ldots, w_l$:

$$\log p(\boldsymbol{w}) = \log \sum_{t \in T_{\mathcal{G}_L}(\boldsymbol{w})} p(t)\,, \qquad (1)$$

where $p(t) = \prod_{r \in t} \pi_r$ and $T_{\mathcal{G}_L}(\boldsymbol{w})$ consists of all possible *lexicalized* parse trees of the sentence $\boldsymbol{w}$ under an L-PCFG $\mathcal{G}_L$. We can compute the marginal $p(\mathbf{w})$ of the sentence by using the inside algorithm in polynomial time. The core recursion of the inside algorithm is formalized in Equation 3. It recursively computes the probability $s_{i,j}^{A,p}$ of a head-parent $A[w_p]$ spanning the substring $w_i, \ldots, w_{j-1}$ ($p \in [i, j-1]$). Term A1 and A2 in Equation 3 cover the cases of the head-child as the left child and the right child respectively.

2.3 Challenges of L-PCFG induction

The major difference between L-PCFGs from vanilla PCFGs is that they use word-annotated nonterminals, so the nonterminal number of L-PCFGs

[1] An alternative definition of CFGs does not distinguish nonterminals $\mathcal{N}$ (constituent labels) from preterminals $\mathcal{P}$ (part-of-speech tags) and treats both as nonterminals.

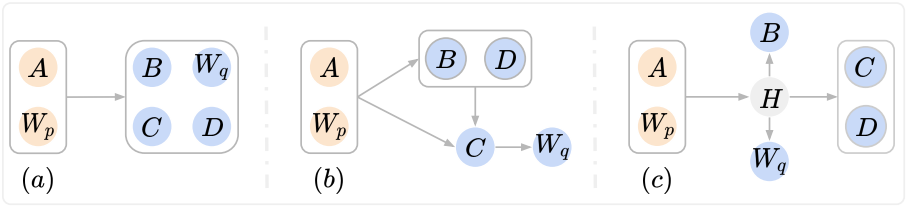


Figure 1: (a) The original parameterization of L-PCFGs. (b) The parameterization of Zhu et al. (2020): W_q is independent with B, D, A, W_p given C. (c) Our proposed parameterization. We slightly abuse the Bayesian network notation by grouping variables. In the standard notation, there would be arcs from the parent variables to each grouped variable as well as arcs between the grouped variables.

is up to $|\Sigma|$ times the number of nonterminals in PCFGs. As the grammar size is largely determined by the number of binary rules and increases approximately in cubic of the nonterminal number, representing L-PCFGs has a high space complexity $\mathcal{O}(m^3|\Sigma|^2)$ (m is the nonterminal number). Specifically, it requires an order-6 probability tensor for binary rules with each dimension representing A, B, C, w_p, w_q, and head direction D, respectively. With so many rules, L-PCFGs are very prone to the data sparsity problem in rule probability estimation. Collins (2003) suggests factorizing the binary rule probabilities according to specific independence assumptions, but his approach still relies on complicated smoothing techniques to be effective.

The addition of lexical heads also scales up the computational complexity of the inside algorithm by a factor $\mathcal{O}(l^2)$ and brings it up to $\mathcal{O}(l^5m^3)$. Eisner and Satta (1999) point out that, by changing the order of summations in Term A1 (A2) of Equation 3, one can cache and reuse Term B1 (B2) in Equation 4 and reduce the computational complexity to $\mathcal{O}(l^4m^2 + l^3m^3)$. This is an example application of unfold-refold as noted by Eisner and Blatz (2007). However, the complexity is still cubic in m, making it expensive to increase the total number of nonterminals.

2.4 Neural L-PCFGs

Zhu et al. (2020) apply neural parameterization to tackle the data sparsity issue and to reduce the total learnable parameters of L-PCFGs. Considering the head-child as the left child (similarly for the other case), they further factorize the binary rule probability as:

$$p(A[w_p] \rightarrow B[w_p]C[w_q]) = p(B, \backsim, C|A, w_p)p(w_q|C)\,. \quad (2)$$

Bayesian networks representing the original probability and the factorization are illustrated in Figure 1 (a) and (b). With the factorized binary rule probability in Equation 2, Term A1 in Equation 3 can be rewritten as Equation 5. Zhu et al. (2020) implement the inside algorithm by caching Term C1-1 in Equation 6, resulting in a time complexity $\mathcal{O}(l^4m^3 + l^3m)$, which is cubic in m. We note that, we can use unfold-refold to further cache Term C1-2 in Equation 6 and reduce the time complexity of the inside algorithm to $\mathcal{O}(l^4m^2 + l^3m + l^2m^2)$, which is quadratic in m.

Although the factorization of Equation 2 reduces the space and time complexity of the inside algorithm of L-PCFG, it is based on the independence assumption that the generation of w_q is independent of A, B, D and w_p given the non-head-child C. This assumption can be violated in many scenarios and hence reduces the expressiveness of the grammar. For example, suppose C is Noun, then even if we know B is Verb, we still need to know D to determine if w_q is an object or a subject of the verb, and then need to know the actual verb w_p to pick a likely noun as w_q.

3 Factorization with latent variable

Our main goal is to find a parameterization that removes the implausible independence assumptions of Zhu et al. (2020) while decreases the complexities of the original L-PCFGs.

To reduce the representation complexity, we draw inspiration from the canonical polyadic decomposition (CPD). CPD factorizes an n-th order tensor into n two-dimensional matrices. Each matrix consists of two dimensions: one dimension comes from the original n-th order tensor and the other dimension is shared by all the n matrices. The shared dimension can be marginalized to recover the original n-th order tensor. From a probabilistic perspective, the shared dimension can be regarded as a latent-variable. In the spirit of CPD, we introduce a latent-variable H to decompose the order-6

$$s_{i,j}^{A,p} = \underbrace{\sum_{k=p+1}^{j-1}\sum_{q=k}^{j-1}\sum_{B,C} s_{i,k}^{B,p}\cdot s_{k,j}^{C,q}\cdot p(A[w_p]\rightarrow B[w_p]C[w_q])}_{\text{Term A1}} + \underbrace{\sum_{k=i+1}^{p}\sum_{q=i}^{k-1}\sum_{B,C} s_{i,k}^{B,q}\cdot s_{k,j}^{C,p}\cdot p(A[w_p]\rightarrow B[w_q]C[w_p])}_{\text{Term A2}} \quad (3)$$

$$= \sum_{k=p+1}^{j-1}\sum_{B} s_{i,k}^{B,p}\underbrace{\sum_{q=k}^{j-1}\sum_{C} s_{k,j}^{C,q}\cdot p(A[w_p]\rightarrow B[w_p]C[w_q])}_{\text{Term B1}} + \sum_{k=i+1}^{p}\sum_{C} s_{k_2,j}^{C,p}\underbrace{\sum_{q=i}^{k-1}\sum_{B} s_{i,k}^{B,q}\cdot p(A[w_p]\rightarrow B[w_q]C[w_p])}_{\text{Term B2}} \quad (4)$$

$$\text{Term A1} = \sum_{k=p+1}^{j-1}\sum_{q=k}^{j-1}\sum_{B,C} s_{i,k}^{B,p}\cdot s_{k,j}^{C,q}\cdot \underbrace{p(B,\leftharpoonup,C|A,w_p)\cdot p(w_q|C)}_{\text{factorization of } p(A[w_p]\rightarrow B[w_p]C[w_q])} \quad (5)$$

$$= \sum_{k=p+1}^{j-1}\sum_{B} s_{i,k}^{B,p}\underbrace{\sum_{C} p(B,\leftharpoonup,C|A,w_p)\underbrace{\sum_{q=k}^{j-1} s_{k,j}^{C,q}\cdot p(w_q|C)}_{\text{Term C1-1}}}_{\text{Term C1-2}} \quad (6)$$

$$\text{Term A1} = \sum_{k=p+1}^{j-1}\sum_{q=k}^{j-1}\sum_{B,C} s_{i,k}^{B,p}\cdot s_{k,j}^{C,q}\cdot \underbrace{\sum_{H} p(H|A,w_p)p(B|H)p(C,\leftharpoonup|H)p(w_q|H)}_{\text{factorization of } p(A[w_p]\rightarrow B[w_p]C[w_q])} \quad (7)$$

$$= \sum_{H} p(H|A,w_p)\sum_{k=p+1}^{j-1}\underbrace{\sum_{B} s_{i,k}^{B,p}p(B|H)}_{\text{Term D1-1}}\underbrace{\sum_{q=k}^{j-1}\sum_{C} s_{k,j}^{C,q}p(C\leftharpoonup|H)p(w_q|H)}_{\text{Term D1-2}} \quad (8)$$

Table 1: Recursive formulas of the inside algorithm for Eisner and Satta (1999) (Equation 4), Zhu et al. (2020) (Equation 5- 6), and our formalism (Equation 7- 8), respectively. $s_{i,j}^{A,p}$ indicates the probability of a head nonterminal $A[w_p]$ spanning the substring $w_i, \ldots, w_{j-1}$, where p is the position of the headword in the sentence.

probability tensor $p(B, C, D, w_q|A, w_p)$. Instead of fully decomposing the tensor, we empirically find that binding some of the variables leads to better results. Our best factorization is as follows (also illustrated by a Bayesian network in Figure 1 (c)):

$$p(B, C, W_q, D|A, W_p) = \sum_{H} p(H|A, W_p)p(B|H)p(C, D|H)p(W_q|H)\,. \quad (9)$$

According to d-separation (Pearl, 1988), when A and w_p are given, B, C, w_q, and D are interdependent due to the existence of H. In other words, our factorization does not make any independence assumption beyond the original binary rule. The domain size of H is analogous to the tensor rank in CPD and thus influences the expressiveness of our proposed model.

Based on our factorization approach, the binary rule probability is factorized as

$$p(A[w_p]\rightarrow B[w_p]C[w_q]) = \sum_{H} p(H|A,w_p)p(B|H)p(C\leftharpoonup|H)p(w_q|H)\,, \quad (10)$$

and

$$p(A[w_p]\rightarrow B[w_q]C[w_p]) = \sum_{H} p(H|A,w_p)p(C|H)p(B\rightharpoonup|H)p(w_q|H)\,. \quad (11)$$

We also follow Zhu et al. (2020) and factorize the start rule as follows.

$$p(S\rightarrow A[w_p]) = p(A|S)p(w_p|A)\,. \quad (12)$$

Computational complexity: Considering the head-child as the left child (similarly for the other case), we apply Equation 10 in Term A1 of Equation 3 and obtain Equation 7. Rearranging the summations in Equation 7 gives Equation 8, where Term D1-1 and D1-2 can be cached and reused, which also uses the unfold-refold technique. The final time complexity of the inside computation with

our factorization approach is $\mathcal{O}(l^4 d_H + l^2 m d_H)$ (d_H is the domain size of the latent variable H), which is linear in m.

Choices of factorization: If we follow the intuition of CPD, then we shall assume that B, C, D, and w_q are all independent conditioned on H. However, properly relaxing this strong assumption by binding some variables could benefit our model. Though there are many different choices of binding the variables, some bindings can be easily ruled out. For instance, binding B and C inhibits us from caching Term D1-1 and Term D1-2 in Equation 7 and thus we cannot implement the inside algorithm efficiently; binding C and w_q leads to a high computational complexity because we will have to compute a high-dimensional ($m|\Sigma|$) categorical distribution. In Section 6.3, we make an ablation study on the impact of different choices of factorizations.

Neural parameterizations: We follow Kim et al. (2019) and Zhu et al. (2020) and define the following neural parameterization:

$$p(A|S) = \frac{\exp(\mathbf{u}_S^\top f_1(\mathbf{w}_A))}{\sum_{A' \in \mathcal{N}} \exp(\mathbf{u}_S^\top f_1(\mathbf{w}_{A'}))},$$

$$p(w|A) = \frac{\exp(\mathbf{u}_A^\top f_2(\mathbf{w}_w))}{\sum_{w' \in \Sigma} \exp(\mathbf{u}_A^\top f_2(\mathbf{w}_{w'}))},$$

$$p(B|H) = \frac{\exp(\mathbf{u}_H^\top w_B)}{\sum_{B' \in \mathcal{N} \cup \mathcal{P}} \exp(\mathbf{u}_H^\top \mathbf{w}_{B'})},$$

$$p(w|H) = \frac{\exp(\mathbf{u}_H^\top f_2(\mathbf{w}_w))}{\sum_{w' \in \Sigma} \exp(\mathbf{u}_H^\top f_2(\mathbf{w}_{w'}))},$$

$$p(C \leftarrow |H) = \frac{\exp(\mathbf{u}_H^\top \mathbf{w}_{C\leftarrow})}{\sum_{C' \in \mathcal{M}} \exp(\mathbf{u}_H^\top \mathbf{w}_{C'})},$$

$$p(C \rightarrow |H) = \frac{\exp(\mathbf{u}_H^\top \mathbf{w}_{C\rightarrow})}{\sum_{C' \in \mathcal{M}} \exp(\mathbf{u}_H^\top \mathbf{w}_{C'})},$$

$$p(H|A, w) = \frac{\exp(\mathbf{u}_H^\top f_4([\mathbf{w}_A; \mathbf{w}_w]))}{\sum_{H' \in \mathcal{H}} \exp(\mathbf{u}_{H'}^\top f_4([\mathbf{w}_A; \mathbf{w}_w]))},$$

where $\mathcal{H} = \{H_1, \ldots, H_{d_H}\}$, $\mathcal{M} = (\mathcal{N} \cup \mathcal{P}) \times \{\leftarrow, \rightarrow\}$, $\mathbf{u}$ and $\mathbf{w}$ are nonterminal embeddings and word embeddings respectively, and $f_1(\cdot)$, $f_2(\cdot)$, $f_3(\cdot)$, $f_4(\cdot)$ are neural networks with residual layers (He et al., 2016) (Full parameterization is shown in Appendix.).

4 Experimental setup

4.1 Dataset

We conduct experiments on the Wall Street Journal (WSJ) corpus of the Penn Treebank (Marcus et al., 1994). We use the same preprocessing pipeline as in Kim et al. (2019). Specifically, punctuation is removed from all data splits and the top 10,000 frequent words in the training data are used as the vocabulary. For dependency grammar induction, we follow (Zhu et al., 2020) to use the Stanford typed dependency representation (de Marneffe and Manning, 2008).

4.2 Hyperparameters

We optimize our model using the Adam optimizer with $\beta_1 = 0.75, \beta_2 = 0.999$, and learning rate 0.001. All parameters are initialized with Xavier uniform initialization. We set the dimension of all embeddings to 256 and the ratio of the nonterminal number to the preterminal number to 1:2. Our best model uses 15 nonterminals, 30 preterminals, and $d_H = 300$. We use grid search to tune the nonterminal number (from 5 to 30) and domain size d_H of the latent H (from 50 to 500).

4.3 Evaluation

We run each model four times with different random seeds and for ten epochs. We train our models on training sentences of length $\leqslant 40$ with batch size 8 and test them on the whole testing set. For each run, we perform early stopping and select the best model according to the perplexity of the development set. We use two different parsing methods: the variant of CYK algorithm (Eisner and Satta, 1999) and Minimum Bayes-Risk (MBR) decoding (Smith and Eisner, 2006). [2] For constituent grammar induction, we report the means and standard deviations of sentence-level F1 scores.[3] For dependency grammar induction, we report unlabeled directed attachment score (UDAS) and unlabeled undirected attachment score (UUAS).

5 Main result

We present our main results in Table 2. Our model is referred to as **N**eural **Bi**-**L**exicalized **PCFGs** (NBL-PCFGs). We mainly compare our approach against recent PCFG-based models: neural PCFG (N-PCFG) and compound PCFG (C-PCFG) (Kim et al., 2019), tensor decomposition based neural

[2] In MBR decoding, we use automatic differentiation (Eisner, 2016; Rush, 2020) to estimate the marginals of spans and arcs, and then use the CYK and Eisner algorithms for constituency and dependency parsing, respectively.

[3] Following Kim et al. (2019), we remove all trivial spans (single-word spans and sentence-level spans). Sentence-level means that we compute F1 for each sentence and then average over all sentences.

PCFG (TN-PCFG) (Yang et al., 2021) and neural L-PCFG (NL-PCFG) (Zhu et al., 2020). We report both official result of Zhu et al. (2020) and our reimplementation.

We do not use the compound trick (Kim et al., 2019) in our implementations of lexicalized PCFGs because we empirically find that using it results in unstable training and does not necessarily bring performance improvements.

We draw three key observations: (1) Our model achieves the best F1 and UUAS scores under both CYK and MBR decoding. It is also comparable to the official NL-PCFG in the UDAS score. (2) When we remove the compound parameterization from NL-PCFG, its F1 score drops slightly while its UDAS and UUAS scores drop dramatically. It implies that compound parameterization is the key to achieve excellent dependency grammar induction performance in NL-PCFG. (3) The MBR decoding outperforms CYK decoding.

Regarding UDAS, our model significantly outperforms NL-PCFGs in UDASs if compound parameterization is not used (37.1 vs. 23.8 with CYK decoding), showing that explicitly modeling bilexical relationship is helpful in dependency grammar induction. However, when compound parameterization is used, the UDAS of NL-PCFGs is greatly improved, slightly surpassing that of our model. We believe this is because compound parameterization greatly weakens the independence assumption of NL-PCFGs (i.e., the child word is dependent on C only) by leaking bilexical information via the global sentence embedding. On the other hand, NBL-PCFGs are already expressive enough and thus compound parameterization brings no further increase of their expressiveness but makes learning more difficult.

Model	WSJ		
	F1	UDAS	UUAS
Official results			
N-PCFG*	50.8		
C-PCFG*	55.2		
NL-PCFG*	55.3	**39.7**	53.3
TN-PCFG†	57.7		
Our results			
NL-PCFG*	$53.3_{\pm 2.1}$	$23.8_{\pm 1.1}$	$47.4_{\pm 1.0}$
NL-PCFG†	$57.4_{\pm 1.4}$	$25.3_{\pm 1.3}$	$47.2_{\pm 0.7}$
NBL-PCFG*	$58.2_{\pm 1.5}$	$37.1_{\pm 2.8}$	$54.6_{\pm 1.3}$
NBL-PCFG†	$\mathbf{60.4}_{\pm 1.6}$	$39.1_{\pm 2.8}$	$\mathbf{56.1}_{\pm 1.3}$
For reference			
S-DIORA	57.6		
StructFormer	54.0	46.2	61.6
Oracle Trees	84.3		

Table 2: Unlabeled sentence-level F1 scores, unlabeled directed attachment scores and unlabeled undirected attachment scores on the WSJ test data. † indicates using MBR decoding. * indicates using CYK decoding. Recall that the official result of Zhu et al. (2020) uses compound parameterization while our reimplementation removes the compound parameterization. S-DIORA: Drozdov et al. (2020). StructFormer: Shen et al. (2020).

6 Analysis

In the following experiments, we report results using MBR decoding by default. We also use $d_H = 300$ by default unless otherwise specified.

6.1 Influence of the domain size of H

d_H (the domain size of H) influences the expressiveness of our model. Figure 2a illustrates perplexities and F1 scores with the increase of d_H and a fixed nonterminal number of 10 (plots of UDAS and UUAS can be found in Appendix). We can see that when d_H is small, the model has a high perplexity and a low F1 score, indicating the limited expressiveness of NBL-PCFGs. When d_H is larger than 300, the perplexity becomes plateaued and the F1 score starts to decrease possibly because of overfitting.

6.2 Influence of nonterminal number

Figure 2b illustrates perplexities and F1 scores with the increase of the nonterminal number and fixed $d_H = 300$ (plots of UDAS and UUAS can be found in Appendix). We observe that increasing the nonterminal number has only a minor influence on NBL-PCFGs. We speculate that it is because the number of word-annotated nonterminals ($m|\Sigma|$) is already sufficiently large even if m is small. On the other hand, the nonterminal number has a big influence on NL-PCFGs. This is most likely because NL-PCFGs make the independence assumption that the generation of w_q is solely determined by the non-head-child C and thus require more nonterminals so that C has the capacity of conveying information from A, B, D and w_p. Using more nonterminals (> 30) seems to be helpful for NL-

	F1	UDAS	UUAS	Perplexity
D-C	**60.4**	39.1	56.1	**161.9**
D-alone	57.2	32.8	54.1	164.8
D-w_q	47.7	**45.7**	**58.6**	176.8
D-B	47.8	36.9	54.0	169.6

Table 3: Binding the head direction D with different variables.

PCFGs, but would be computationally too expensive due to the quadratically increased complexity in the number of nonterminals.

6.3 Influence of different variable bindings

Table 3 presents the results of our models with the following bindings:

- D-alone: D is generated alone.
- D-w_q: D is generated with w_q.
- D-B: D is generated with head-child B.
- D-C: D is generated with non-head-child C.

Clearly, binding D and C (the default setting for NBL-PCFG) results in the lowest perplexity and the highest F1 score. Binding D and w_q has a surprisingly good performance in unsupervised dependency parsing.

We find that how to bind the head direction has a huge impact on the unsupervised parsing performance and we give the following intuition. Usually given a headword and its type, the children generated in each direction would be different. So, D is intuitively more related to w_q and C than to B. On the other hand, B is dependent more on the headword instead. In Table 3 we can see that (D-B) has a lower UDAS score than (D-C) and (D-w_q), which is consistent with this intuition. Notably, in Zhu et al. (2020), their Factorization III has a significantly lower UDAS than the default model (35.5 vs. 25.9), and the only difference is whether the generation of C is dependent on the head direction. This is also consistent with our intuition.

6.4 Qualitative analysis

We analyze the parsing performance of different PCFG extensions by breaking down their recall numbers by constituent labels (see Table 4). NPs and VPs cover most of the gold constituents in WSJ test set. TN-PCFGs have the best performance in predicting NPs and NBL-PCFGs have better performance in predicting other labels on average.

We further analyze the quality of our induced trees. Our model prefers to predict left-headed constituents (i.e., constituents headed by the leftmost word). VPs are usually left-headed in English, so our model has a much higher recall on VPs and correctly predicts their headwords. SBARs often start with *which* and *that* and PPs often start with prepositions such as *of* and *for*. Our model often relies on these words to predict the correct constituents and hence erroneously predicts these words as the headwords, which hurts the dependency accuracy. For NPs, we find our model often makes mistakes in predicting adjective-noun phrases. For example, the correct parse of *a rough market* is *(a (rough market))*, but our model predicts *((a rough) market)* instead.

7 Discussion on dependency annotation schemes

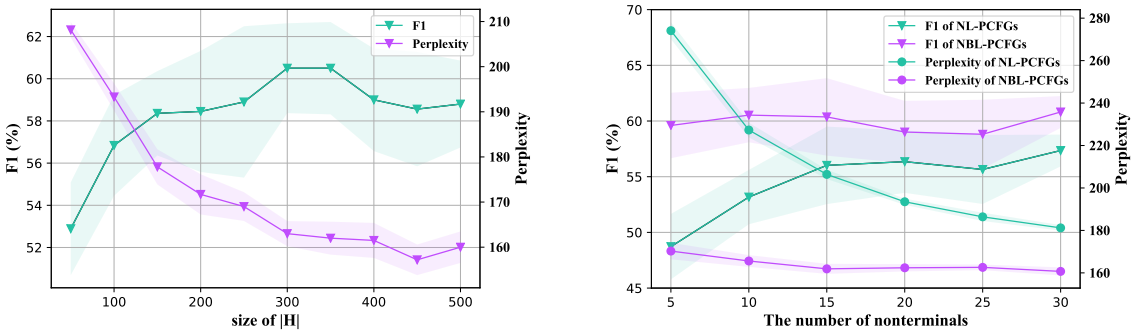
What should be regarded as the headwords is still debatable in linguistics, especially for those around function words (Zwicky, 1993). For example, in phrase *the company*, some linguists argue that *the* should be the headword (Abney, 1972). These disagreements are reflected in the dependency annotation schemes. Researchers have found that different dependency annotation schemes result in very different evaluation scores of unsupervised dependency parsing (Noji, 2016; Shen et al., 2020).

In our experiments, we use the Stanford Dependencies annotation scheme in order to compare with NL-PCFGs. Stanford Dependencies prefers to select content words as headwords. However, as we discussed in previous sections, our model prefers to select function words (e.g., *of*, *which*, *for*) as headwords for SBARs or PPs.This explains why our model can outperform all the baselines on constituency parsing but not on dependency parsing (as judged by Stanford Dependencies) at the same time. Table 3 shows that there is a trade-off between the F1 score and UDAS, which suggests that adapting our model to Stanford Dependencies would hurt its ability to identify constituents.

8 Speed comparison

In practice, the forward and backward pass of the inside algorithm consumes the majority of the running time in training a N(B)L-PCFG. The existing implementation by Zhu et al. (2020)[4] does not employ efficient parallization and has a cubic time

[4]https://github.com/neulab/neural-lpcfg



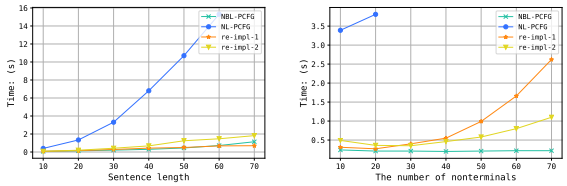
(a) (b)

Figure 2: The change of F1 scores, perplexities with the change of $|H|$ and nonterminal number.

	N-PCFG†	C-PCFG†	TN-PCFG†	NL-PCFG	NBL-PCFG
NP	72.3%	73.6%	**75.4%**	74.0%	66.2%
VP	28.1%	45.0%	48.4%	44.3%	**61.1%**
PP	73.0%	71.4%	67.0%	68.4%	**77.7%**
SBAR	53.6%	54.8%	50.3%	49.4%	**63.8%**
ADJP	40.8%	44.3%	53.6%	55.5%	**59.7%**
ADVP	43.8%	**61.6%**	59.5%	57.1%	59.1%
Perplexity	254.3	196.3	207.3	181.2	**161.9**

Table 4: Recall on six frequent constituent labels and perplexities of the WSJ test data. † means that the results are reported by Yang et al. (2021)

complexity in the number of nonterminals. We provide an efficient reimplementation (we follow Zhang et al. (2020) to batchify) of the inside algorithm based on Equation 6. We refer to an implementation which caches Term C1-1 as *re-impl-1* and refer to an implementation which caches Term C1-2 as *re-impl-2*.



(a) (b)

Figure 3: Total time in performing the inside algorithm and automatic differentiation with different sentence lengths and nonterminal numbers.

We measure the time based on a single forward and backward pass of the inside algorithm with batch size 1 on a single Titan V GPU. Figure 3a illustrates the time with the increase of the sentence length and a fixed nonterminal number of 10. The original implementation of NL-PCFG by Zhu et al. (2020) takes much more time when sentences are long. For example, when sentence length is 40, it needs 6.80s, while our fast implementation takes 0.43s and our NBL-PCFG takes only 0.30s. Figure 3b illustrates the time with the increase of the nonterminal number m and a fixed sentence length of 30. The original implementation runs out of 12GB memory when $m = 30$. *re-impl-2* is faster than *re-impl-1* when increasing m as it has a better time complexity in m (quadratic for *re-impl-2*, cubic for *re-impl-1*). Our NBL-PCFGs have a linear complexity in m, and as we can see in the figure, our NBL-PCFGs are much faster when m is large.

9 Related Work

Unsupervised parsing has a long history but has regained great attention in recent years. In unsupervised dependency parsing, most methods are based on Dependency Model with Valence (DMV) (Klein and Manning, 2004). Neurally parameterized DMVs have obtained state-of-the-art performance (Jiang et al., 2016; Han et al., 2017, 2019; Yang et al., 2020). However, they rely on gold POS tags and sophisticated initializations (e.g. K&M initialization or initialization with the parsing result of another unsupervised model). Noji et al. (2016) propose a left-corner parsing-based DMV model to limit the stack depth of center-embedding, which is insensitive to initialization but needs gold POS tags. He et al. (2018) propose a latent-variable based DMV model, which does not need gold POS tags but requires good initialization and high-quality induced POS tags. See Han et al. (2020) for a survey of unsupervised dependency parsing. Compared to these methods, our method does not require gold/induced POS tags or sophisticated initializations, though its performance lags behind some of these previous methods.

Recent unsupervised constituency parsers can be roughly categorized into the following groups: (1) PCFG-based methods. Depth-bounded PCFGs (Jin et al., 2018a,b) limit the stack depth of center-embedding. Neurally parameterized PCFGs (Jin

et al., 2019; Kim et al., 2019; Zhu et al., 2020; Yang et al., 2021) use neural networks to produce grammar rule probabilities. (2) Deep Inside-Outside Recursive Auto-encoder (DIORA) based methods (Drozdov et al., 2019a,b, 2020; Hong et al., 2020; Sahay et al., 2021). They use neural networks to mimic the inside-outside algorithm and they are trained with masked language model objectives. (3) Syntactic distance-based methods (Shen et al., 2018, 2019, 2020). They encode hidden syntactic trees into syntactic distances and inject them into language models. (4) Probing based methods (Kim et al., 2020; Li et al., 2020). They extract phrase-structure trees based on the attention distributions of large pre-trained language models. In addition to these methods, Cao et al. (2020) use constituency tests and Shi et al. (2021) make use of naturally-occurring bracketings such as hyperlinks on webpages to train parsers. Multimodal information such as images (Shi et al., 2019; Zhao and Titov, 2020; Jin and Schuler, 2020) and videos (Zhang et al., 2021) have also been exploited for unsupervised constituency parsing.

We are only aware of a few previous studies in unsupervised joint dependency and constituency parsing. Klein and Manning (2004) propose a joint DMV and CCM (Klein and Manning, 2002) model. Shen et al. (2020) propose a transformer-based method, in which they define syntactic distances to guild attentions of transformers. Zhu et al. (2020) propose neural L-PCFGs for unsupervised joint parsing.

10 Conclusion

We have presented a new formalism of lexicalized PCFGs. Our formalism relies on the canonical polyadic decomposition to factorize the probability tensor of binary rules. The factorization reduces the space and time complexity of lexicalized PCFGs while keeping the independence assumptions encoded in the original binary rules intact. We further parameterize our model by using neural networks and present an efficient implementation of our model. On the English WSJ test data, our model achieves the lowest perplexity, outperforms all the existing extensions of PCFGs in constituency grammar induction, and is comparable to strong baselines in dependency grammar induction.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was supported by the National Natural Science Foundation of China (61976139).

References

Steven P. Abney. 1972. The english noun phrase in its sentential aspect.

Steven Cao, Nikita Kitaev, and Dan Klein. 2020. Unsupervised parsing via constituency tests. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4798–4808, Online. Association for Computational Linguistics.

Eugene Charniak. 1996. Tree-bank grammars. Technical report, USA.

Justin Chiu and Alexander Rush. 2020. Scaling hidden Markov language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1341–1349, Online. Association for Computational Linguistics.

Shay B. Cohen, Giorgio Satta, and Michael Collins. 2013. Approximate PCFG parsing using tensor decomposition. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 487–496, Atlanta, Georgia. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Andrew Drozdov, Subendhu Rongali, Yi-Pei Chen, Tim O'Gorman, Mohit Iyyer, and Andrew McCallum. 2020. Unsupervised parsing with S-DIORA: Single tree encoding for deep inside-outside recursive autoencoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4832–4845, Online. Association for Computational Linguistics.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019a. 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.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019b. 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.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, Austin, TX. Association for Computational Linguistics.

Jason Eisner and John Blatz. 2007. Program transformations for optimization of parsing algorithms and other weighted logic programs. In *Proceedings of FG 2006: The 11th Conference on Formal Grammar*, pages 45–85. CSLI Publications.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 457–464, College Park, Maryland, USA. Association for Computational Linguistics.

Wenjuan Han, Yong Jiang, Hwee Tou Ng, and Kewei Tu. 2020. A survey of unsupervised dependency parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2522–2533, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Wenjuan Han, Yong Jiang, and Kewei Tu. 2017. Dependency grammar induction with neural lexicalization and big training data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1683–1688, Copenhagen, Denmark. Association for Computational Linguistics.

Wenjuan Han, Yong Jiang, and Kewei Tu. 2019. Enhancing unsupervised generative dependency parser with contextual information. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5315–5325, Florence, Italy. Association for Computational Linguistics.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Unsupervised learning of syntactic structure with invertible neural projections. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1292–1302, Brussels, Belgium. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Ruyue Hong, Jiong Cai, and Kewei Tu. 2020. Deep inside-outside recursive autoencoder with all-span objective. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3610–3615, Barcelona, Spain (Online). International Committee on Computational Linguistics.

F. Jelinek, J. D. Lafferty, and R. L. Mercer. 1992. Basic methods of probabilistic context free grammars. In *Speech Recognition and Understanding*, pages 345–360, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas. Association for Computational Linguistics.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2018a. Depth-bounding is effective: Improvements and evaluation of unsupervised PCFG induction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2721–2731, Brussels, Belgium. Association for Computational Linguistics.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2018b. 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.

Lifeng Jin and William Schuler. 2020. Grounded PCFG induction with images. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 396–408, Suzhou, China. Association for Computational Linguistics.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sanggoo Lee. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019. 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.

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.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan. Association for Computational Linguistics.

Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Bowen Li, Taeuk Kim, Reinald Kim Amplayo, and Frank Keller. 2020. Heads-up! unsupervised constituency parsing via self-attention heads. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 409–424, Suzhou, China. Association for Computational Linguistics.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Manchester, UK. Coling 2008 Organizing Committee.

Hiroshi Noji. 2016. Left-corner methods for syntactic modeling with universal structural constraints. *CoRR*, abs/1608.00293.

Hiroshi Noji, Yusuke Miyao, and Mark Johnson. 2016. Using left-corner parsing to encode universal structural constraints in grammar induction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 33–43, Austin, Texas. Association for Computational Linguistics.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Alexander Rush. 2020. Torch-struct: Deep structured prediction library. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 335–342, Online. Association for Computational Linguistics.

Atul Sahay, Anshul Nasery, Ayush Maheshwari, Ganesh Ramakrishnan, and Rishabh Iyer. 2021. Rule augmented unsupervised constituency parsing.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron C. Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Yikang Shen, Shawn Tan, Seyedarian Hosseini, Zhouhan Lin, Alessandro Sordoni, and Aaron C. Courville. 2019. Ordered memory. *CoRR*, abs/1910.13466.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. 2020. Structformer: Joint unsupervised induction of dependency and constituency structure from masked language modeling.

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.

Tianze Shi, Ozan İrsoy, Igor Malioutov, and Lillian Lee. 2021. Learning syntax from naturally-occurring bracketings. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2941–2949, Online. Association for Computational Linguistics.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 787–794, Sydney, Australia. Association for Computational Linguistics.

Songlin Yang, Yong Jiang, Wenjuan Han, and Kewei Tu. 2020. Second-order unsupervised neural dependency parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3911–3924, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021. PCFGs can do better: Inducing probabilistic context-free grammars with many symbols. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1487–1498, Online. Association for Computational Linguistics.

Songyang Zhang, Linfeng Song, Lifeng Jin, Kun Xu, Dong Yu, and Jiebo Luo. 2021. Video-aided unsupervised grammar induction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1513–1524, Online. Association for Computational Linguistics.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020. Fast and accurate neural CRF constituency parsing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4046–4053. ijcai.org.

Yanpeng Zhao and Ivan Titov. 2020. Visually grounded compound PCFGs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4369–4379, Online. Association for Computational Linguistics.

Hao Zhu, Yonatan Bisk, and Graham Neubig. 2020. The return of lexical dependencies: Neural lexicalized PCFGs. *Transactions of the Association for Computational Linguistics*, 8:647–661.

A. Zwicky. 1993. Heads in grammatical theory: Heads, bases and functors.

A Full Parameterization

We give the full parameterizations of the following probability distributions.

$$p(A|S) = \frac{\exp(\mathbf{u}_S^\top h_1(\mathbf{w}_A))}{\sum_{A' \in \mathcal{N}} \exp(\mathbf{u}_S^\top h_1(\mathbf{w}_{A'}))},$$

$$p(w|A) = \frac{\exp(\mathbf{u}_A^\top h_2(\mathbf{w}_w))}{\sum_{w' \in \Sigma} \exp(\mathbf{u}_A^\top h_2(\mathbf{w}_{w'}))},$$

$$p(w|H) = \frac{\exp(\mathbf{u}_H^\top h_3(\mathbf{w}_w))}{\sum_{w' \in \Sigma} \exp(\mathbf{u}_H^\top h_3(\mathbf{w}_{w'}))},$$

$$p(H|A,w) = \frac{\exp(\mathbf{u}_H^\top f([\mathbf{w}_A; \mathbf{w}_w]))}{\sum_{H' \in \mathcal{H}} \exp(\mathbf{u}_{h'}^\top f([\mathbf{w}_A; \mathbf{w}_w]))},$$

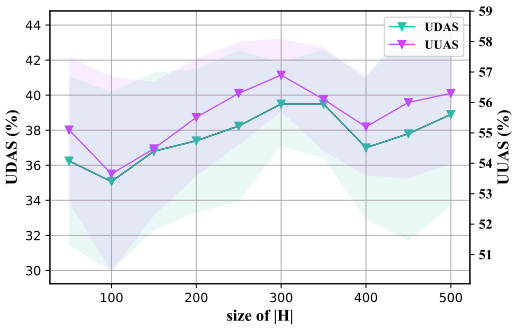


Figure 4: Influence of d_H on UUAS and UDAS.

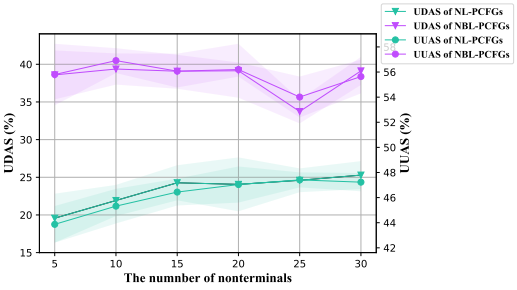


Figure 5: Influence of the number of nonterminals on UUAS and UDAS.

$$h_i(\mathbf{x}) = g_{i,1}\left(g_{i,2}\left(\mathbf{W}_i\mathbf{x}\right)\right)$$
$$g_{i,j}(\mathbf{y}) = \mathrm{ReLU}\left(\mathbf{V}_{i,j}\,\mathrm{ReLU}\left(\mathbf{U}_{i,j}\mathbf{y}\right)\right) + \mathbf{y}$$
$$f([\mathbf{x},\mathbf{y}]) = h_4(\mathrm{ReLU}(\mathbf{W}[x;y]) + y)$$

B Influence of the domain size of H and the number of nonterminals

Figure 4 illustrates the change of UUAS and UDAS with the increase of d_H. We find similar tendencies compared to the change of F1 scores and perplexities with the increase of d_H. $d_H = 300$ performs best. Figure 5 illustrates the change of UUAS and UDAS when increasing the number of nonterminals. We can see that NL-PCFGs benefit from using more nonterminals while NBL-PCFGs have a better performance when the number of nonterminals is relatively small.