

# PCFGs Can Do Better: Inducing Probabilistic Context-Free Grammars with Many Symbols

Songlin Yang<sup>♣</sup>, Yanpeng Zhao<sup>◇</sup>, Kewei Tu<sup>♣\*</sup>

<sup>♣</sup>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  
<sup>◇</sup>ILCC, University of Edinburgh  
{yangsl, tukw}@shanghaitech.edu.cn  
yannzhao.ed@gmail.com

## Abstract

Probabilistic context-free grammars (PCFGs) with neural parameterization have been shown to be effective in unsupervised phrase-structure grammar induction. However, due to the cubic computational complexity of PCFG representation and parsing, previous approaches cannot scale up to a relatively large number of (nonterminal and preterminal) symbols. In this work, we present a new parameterization form of PCFGs based on tensor decomposition, which has at most quadratic computational complexity in the symbol number and therefore allows us to use a much larger number of symbols. We further use neural parameterization for the new form to improve unsupervised parsing performance. We evaluate our model across ten languages and empirically demonstrate the effectiveness of using more symbols.<sup>1</sup>

## 1 Introduction

Unsupervised constituency parsing is the task of inducing phrase-structure grammars from raw text without using parse tree annotations. Early work induces probabilistic context-free grammars (PCFGs) via the Expectation Maximization algorithm and finds the result unsatisfactory (Lari and Young, 1990; Carroll and Charniak, 1992). Recently, PCFGs with neural parameterization (i.e., using neural networks to generate rule probabilities) have been shown to achieve good results in unsupervised constituency parsing (Kim et al., 2019a; Jin et al., 2019; Zhu et al., 2020). However, due to the cubic computational complexity of PCFG representation and parsing, these approaches learn PCFGs with relatively small numbers of nonterminals and preterminals. For example, Jin et al. (2019) use 30

nonterminals (with no distinction between preterminals and other nonterminals) and Kim et al. (2019a) use 30 nonterminals and 60 preterminals.

In this paper, we study PCFG induction with a much larger number of nonterminal and preterminal symbols. We are partly motivated by the classic work of latent variable grammars in supervised constituency parsing (Matsuzaki et al., 2005; Petrov et al., 2006; Liang et al., 2007; Cohen et al., 2012; Zhao et al., 2018). While the Penn treebank grammar contains only tens of nonterminals and preterminals, it has been found that dividing them into subtypes could significantly improve the parsing accuracy of the grammar. For example, the best model from Petrov et al. (2006) contains over 1000 nonterminal and preterminal symbols. We are also motivated by the recent work of Buhai et al. (2019) who show that when learning latent variable models, increasing the number of hidden states is often helpful; and by Chiu and Rush (2020) who show that a neural hidden Markov model with up to  $2^{16}$  hidden states can achieve surprisingly good performance in language modeling.

A major challenge in employing a large number of nonterminal and preterminal symbols is that representing and parsing with a PCFG requires a computational complexity that is cubic in its symbol number. To resolve the issue, we rely on a new parameterization form of PCFGs based on tensor decomposition, which reduces the computational complexity from cubic to at most quadratic. Furthermore, we apply neural parameterization to the new form, which is crucial for boosting unsupervised parsing performance of PCFGs as shown by Kim et al. (2019a).

We empirically evaluate our approach across ten languages. On English WSJ, our best model with 500 preterminals and 250 nonterminals improves over the model with 60 preterminals and 30 nonter-

\*Corresponding Author

<sup>1</sup>Our code: <https://github.com/sustcsonglin/TN-PCFG>

minals by 6.3% mean F1 score, and we also observe consistent decrease in perplexity and overall increase in F1 score with more symbols in our model, thus confirming the effectiveness of using more symbols. Our best model also surpasses the strong baseline Compound PCFGs (Kim et al., 2019a) by 1.4% mean F1. We further conduct multilingual evaluation on nine additional languages. The evaluation results suggest good generalizability of our approach on languages beyond English.

Our key contributions can be summarized as follows: (1) We propose a new parameterization form of PCFGs based on tensor decomposition, which enables us to use a large number of symbols in PCFGs. (2) We further apply neural parameterization to improve unsupervised parsing performance. (3) We evaluate our model across ten languages and empirically show the effectiveness of our approach.

## 2 Related work

### Grammar induction using neural networks:

There is a recent resurgence of interest in unsupervised constituency parsing, mostly driven by neural network based methods (Shen et al., 2018a, 2019; Drozdov et al., 2019, 2020; Kim et al., 2019a,b; Jin et al., 2019; Zhu et al., 2020). These methods can be categorized into two major groups: those built on top of a generative grammar and those without a grammar component. The approaches most related to ours belong to the first category, which use neural networks to produce grammar rule probabilities. Jin et al. (2019) use an invertible neural projection network (a.k.a. normalizing flow (Rezende and Mohamed, 2015)) to parameterize the preterminal rules of a PCFG. Kim et al. (2019a) use neural networks to parameterize all the PCFG rules. Zhu et al. (2020) extend their work to lexicalized PCFGs, which are more expressive than PCFGs and can model both dependency and constituency parse trees simultaneously.

In other unsupervised syntactic induction tasks, there is also a trend to use neural networks to produce grammar rule probabilities. In unsupervised dependency parsing, the Dependency Model with Valence (DMV) (Klein and Manning, 2004) has been parameterized neurally to achieve higher induction accuracy (Jiang et al., 2016; Yang et al., 2020). In part-of-speech (POS) induction, neurally parameterized Hidden Markov Models (HMM) also achieve state-of-the-art results (Tran et al., 2016; He et al., 2018).

**Tensor decomposition on PCFGs:** Our work is closely related to Cohen et al. (2013) in that both use tensor decomposition to parameterize the probabilities of binary rules for the purpose of reducing the time complexity of the inside algorithm. However, Cohen et al. (2013) use this technique to speed up inference of an existing PCFG, and they need to actually perform tensor decomposition on the rule probability tensor of the PCFG. In contrast, we draw inspiration from this technique to design a new parameterization form of PCFG that can be directly learned from data. Since we do not have a probability tensor to start with, additional tricks have to be inserted in order to ensure validity of the parameterization, as will be discussed later.

## 3 Background

### 3.1 Tensor form of PCFGs

PCFGs build upon context-free grammars (CFGs). We start by introducing CFGs and establishing notations. 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,<sup>2</sup>  $\Sigma$  is a finite set of terminal symbols, and  $\mathcal{R}$  is a set of rules in the following form:

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

PCFGs extend CFGs by associating each rule  $r \in \mathcal{R}$  with a probability  $\pi_r$ . Denote  $n$ ,  $p$ , and  $q$  as the number of symbols in  $\mathcal{N}$ ,  $\mathcal{P}$ , and  $\Sigma$ , respectively. It is convenient to represent the probabilities of the binary rules in the tensor form:

$$\mathbf{T}_{h_A, h_B, h_C} = \pi_{A \rightarrow BC}, \quad \mathbf{T} \in \mathbb{R}^{n \times m \times m},$$

where  $\mathbf{T}$  is an order-3 tensor,  $m = n + p$ , and  $h_A \in [0, n)$  and  $h_B, h_C \in [0, m)$  are symbol indices. For the convenience of computation, we assign indices  $[0, n)$  to nonterminals in  $\mathcal{N}$  and  $[n, m)$  to preterminals in  $\mathcal{P}$ . Similarly, for a preterminal rule we define

$$\mathbf{Q}_{h_T, h_w} = \pi_{T \rightarrow w}, \quad \mathbf{Q} \in \mathbb{R}^{p \times q}.$$

<sup>2</sup>Strictly, CFGs do not distinguish nonterminals  $\mathcal{N}$  (constituent labels) from preterminals  $\mathcal{P}$  (part-of-speech tags). They are both treated as nonterminals.  $\mathcal{N}, \mathcal{P}, \Sigma$  satisfy  $\mathcal{N} \cap \mathcal{P} = \emptyset$  and  $(\mathcal{N} \cup \mathcal{P}) \cap \Sigma = \emptyset$ .

Again,  $h_T$  and  $h_w$  are the preterminal index and the terminal index, respectively. Finally, for a start rule we define

$$\mathbf{r}_{h_A} = \pi_{S \rightarrow A}, \quad \mathbf{r} \in \mathbb{R}^n.$$

Generative learning of PCFGs involves maximizing the log-likelihood of every observed sentence  $\mathbf{w} = w_1, \dots, w_l$ :

$$\log p_\theta(\mathbf{w}) = \log \sum_{t \in T_G(\mathbf{w})} p(t),$$

where  $T_G(\mathbf{w})$  contains all the parse trees of the sentence  $\mathbf{w}$  under a PCFG  $\mathcal{G}$ . The probability of a parse tree  $t \in T_G$  is defined as  $p(t) = \prod_{r \in t_{\mathcal{R}}} \pi_r$ , where  $t_{\mathcal{R}}$  is the set of rules used in the derivation of  $t$ .  $\log p_\theta(\mathbf{w})$  can be estimated efficiently through the inside algorithm, which is fully differentiable and amenable to gradient optimization methods.

### 3.2 Tensor form of the inside algorithm

We first pad  $\mathbf{T}$ ,  $\mathbf{Q}$ , and  $\mathbf{r}$  with zeros such that  $\mathbf{T} \in \mathbb{R}^{m \times m \times m}$ ,  $\mathbf{Q} \in \mathbb{R}^{m \times q}$ ,  $\mathbf{r} \in \mathbb{R}^m$ , and all of them can be indexed by both nonterminals and preterminals.

The inside algorithm computes the probability of a symbol  $A$  spanning a substring  $\mathbf{w}_{i,j} = w_i, \dots, w_j$  in a recursive manner ( $0 \leq i < j < l$ ):

$$s_{i,j}^A = \sum_{k=i}^{j-1} \sum_{B,C} \pi_{A \rightarrow BC} \cdot s_{i,k}^B \cdot s_{k+1,j}^C. \quad (1)$$

$$\text{Base Case: } s_{i,i}^T = \pi_{T \rightarrow w_i}, 0 \leq i < l.$$

We use the tensor form of PCFGs to rewrite Equation 1 as:

$$\begin{aligned} s_{i,j}^{h_A} &= \sum_{k=i}^{j-1} \sum_{h_B, h_C} \mathbf{T}_{h_A, h_B, h_C} \cdot s_{i,k}^{h_B} \cdot s_{k+1,j}^{h_C} \\ &= \sum_{k=i}^{j-1} (\mathbf{T}_{h_A} \cdot \mathbf{s}_{k+1,j}) \cdot \mathbf{s}_{i,k}, \end{aligned} \quad (2)$$

where  $\mathbf{s}_{i,j}$ ,  $\mathbf{s}_{i,k}$ , and  $\mathbf{s}_{k+1,j}$  are all  $m$ -dimensional vectors; the dimension  $h_A$  corresponds to the symbol  $A$ . Thus

$$s_{i,j} = \sum_{k=i}^{j-1} (\mathbf{T} \cdot \mathbf{s}_{k+1,j}) \cdot \mathbf{s}_{i,k}. \quad (3)$$

Equation 3 represents the core computation of the inside algorithm as tensor-vector dot product. It is amenable to be accelerated on a parallel computing device such as GPUs. However, the time and space complexity is cubic in  $m$ , which makes it impractical to use a large number of nonterminals and preterminals.

## 4 Parameterizing PCFGs based on tensor decomposition

The tensor form of the inside algorithm has a high computational complexity of  $\mathcal{O}(m^3 l^3)$ . It hinders the algorithm from scaling to a large  $m$ . To resolve the issue, we resort to a new parameterization form of PCFGs based on tensor decomposition (TD-PCFGs) (Cohen et al., 2013). As discussed in Section 2, while Cohen et al. (2013) use a TD-PCFG to approximate an existing PCFG for speedup in parsing, we regard a TD-PCFG as a stand-alone model and learn it directly from data.

The basic idea behind TD-PCFGs is using Kruskal decomposition of the order-3 tensor  $\mathbf{T}$ . Specifically, we require  $\mathbf{T}$  to be in the Kruskal form,

$$\mathbf{T} = \sum_{l=1}^d \mathbf{T}^{(l)}, \quad \mathbf{T}^{(l)} = \mathbf{u}^{(l)} \otimes \mathbf{v}^{(l)} \otimes \mathbf{w}^{(l)}, \quad (4)$$

where  $\mathbf{u}^{(l)} \in \mathbb{R}^n$  is a column vector of a matrix  $\mathbf{U} \in \mathbb{R}^{n \times d}$ ;  $\mathbf{v}^{(l)}, \mathbf{w}^{(l)} \in \mathbb{R}^m$  are column vectors of matrices  $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{m \times d}$ , respectively;  $\otimes$  indicates Kronecker product. Thus  $\mathbf{T}^{(l)} \in \mathbb{R}^{m \times m \times m}$  is an order-3 tensor and

$$\mathbf{T}_{i,j,k}^{(l)} = \mathbf{u}_i^{(l)} \cdot \mathbf{v}_j^{(l)} \cdot \mathbf{w}_k^{(l)}.$$

The Kruskal form of the tensor  $\mathbf{T}$  is crucial for reducing the computation of Equation 3. To show this, we let  $\mathbf{x} = \mathbf{s}_{i,k}$ ,  $\mathbf{y} = \mathbf{s}_{k+1,j}$ , and  $\mathbf{z}$  be any summand in the right-hand side of Equation 3, so we have:

$$\mathbf{z} = (\mathbf{T} \cdot \mathbf{y}) \cdot \mathbf{x}. \quad (5)$$

Substitute  $\mathbf{T}$  in Equation 4 into Equation 5 and consider the  $i$ -th dimension of  $\mathbf{z}$ :

$$\begin{aligned} z_i &= (\mathbf{T}_i \cdot \mathbf{y}) \cdot \mathbf{x} \\ &= \sum_{j=1}^m \sum_{k=1}^m \sum_{l=1}^d \mathbf{T}_{i,j,k}^{(l)} \cdot \mathbf{x}_j \cdot \mathbf{y}_k \\ &= \sum_{j=1}^m \sum_{k=1}^m \sum_{l=1}^d \mathbf{u}_i^{(l)} \cdot \mathbf{v}_j^{(l)} \cdot \mathbf{w}_k^{(l)} \cdot \mathbf{x}_j \cdot \mathbf{y}_k \\ &= \sum_{l=1}^d \mathbf{u}_i^{(l)} \cdot \left( \sum_{j=1}^m \mathbf{v}_j^{(l)} \cdot \mathbf{x}_j \right) \cdot \left( \sum_{k=1}^m \mathbf{w}_k^{(l)} \cdot \mathbf{y}_k \right) \\ &= \sum_{l=1}^d \mathbf{u}_i^{(l)} \cdot (\mathbf{x}^T \mathbf{v}^{(l)}) \cdot (\mathbf{y}^T \mathbf{w}^{(l)}) \\ &= (\mathbf{e}_i^T \mathbf{U}) \cdot ((\mathbf{V}^T \mathbf{x}) \odot (\mathbf{W}^T \mathbf{y})), \end{aligned} \quad (6)$$

where  $\odot$  indicates Hadamard (element-wise) product;  $\mathbf{e}_i \in \mathbb{R}^m$  is a one-hot vector that selects the  $i$ -th row of  $\mathbf{U}$ . We have padded  $\mathbf{U}$  with zeros such that  $\mathbf{U} \in \mathbb{R}^{m \times d}$  and the last  $m - n$  rows are all zeros. Thus

$$\mathbf{z} = \mathbf{U} \cdot \left( (\mathbf{V}^T \mathbf{x}) \odot (\mathbf{W}^T \mathbf{y}) \right), \quad (7)$$

and accordingly,

$$\mathbf{s}_{i,j} = \mathbf{U} \cdot \sum_{k=i}^{j-1} \left( (\mathbf{V}^T \mathbf{s}_{i,k}) \odot (\mathbf{W}^T \mathbf{s}_{k+1,j}) \right). \quad (8)$$

Equation 8 computes the inside probabilities using TD-PCFGs. It has a time complexity  $\mathcal{O}(md)$ . By caching  $\mathbf{V}^T \mathbf{s}_{i,k}$  and  $\mathbf{W}^T \mathbf{s}_{k+1,j}$ , the time complexity of the inside algorithm becomes  $\mathcal{O}(dl^3 + mdl^2)$  (Cohen et al., 2013), which is at most quadratic in  $m$  since we typically set  $d = \mathcal{O}(m)$ . Interestingly, Equation 8 has similar forms to recursive neural networks (Socher et al., 2013) if we treat inside score vectors as span embeddings.

One problem with TD-PCFGs is that, since we use three matrices  $\mathbf{U}$ ,  $\mathbf{V}$  and  $\mathbf{W}$  to represent tensor  $\mathbf{T}$  of binary rule probabilities, how we can ensure that  $\mathbf{T}$  is non-negative and properly normalized, i.e., for a given left-hand side symbol  $A$ ,  $\sum_{j,k} \mathbf{T}_{h_A,j,k} = 1$ . Simply reconstructing  $\mathbf{T}$  with  $\mathbf{U}$ ,  $\mathbf{V}$  and  $\mathbf{W}$  and then performing normalization would take  $\mathcal{O}(m^3)$  time, thus defeating the purpose of TD-PCFGs. Our solution is to require that the three matrices are non-negative and meanwhile  $\mathbf{U}$  is row-normalized and  $\mathbf{V}$  and  $\mathbf{W}$  are column-normalized (Shen et al., 2018b).

**Theorem 1.** *Given non-negative matrices  $\mathbf{U} \in \mathbb{R}^{n \times d}$  and  $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{m \times d}$ , if  $\mathbf{U}$  is row-normalized and  $\mathbf{V}$  and  $\mathbf{W}$  are column-normalized, then  $\mathbf{U}$ ,  $\mathbf{V}$ , and  $\mathbf{W}$  are a Kruskal decomposition of a tensor  $\mathbf{T} \in \mathbb{R}^{n \times m \times m}$  where  $\mathbf{T}_{i,j,k} \in [0, 1]$  and  $\mathbf{T}_i$  is normalized such that  $\sum_{j,k} \mathbf{T}_{i,j,k} = 1$ .*

*Proof.*

$$\begin{aligned} \sum_{j=1}^m \sum_{k=1}^m \mathbf{T}_{i,j,k} &= \sum_{j=1}^m \sum_{k=1}^m \sum_{l=1}^d \mathbf{u}_i^{(l)} \cdot \mathbf{v}_j^{(l)} \cdot \mathbf{w}_k^{(l)} \\ &= \sum_{l=1}^d \mathbf{u}_i^{(l)} \cdot \left( \sum_{j=1}^m \mathbf{v}_j^{(l)} \right) \cdot \left( \sum_{k=1}^m \mathbf{w}_k^{(l)} \right) \\ &= \sum_{l=1}^d \mathbf{u}_i^{(l)} \cdot 1 \cdot 1 = 1 \end{aligned}$$

□

## 5 Neural parameterization of TD-PCFGs

We use neural parameterization for TD-PCFGs as it has demonstrated its effectiveness in inducing PCFGs (Kim et al., 2019a). In a neurally parameterized TD-PCFGs, the original TD-PCFG parameters are generated by neural networks, rather than being learned directly; parameters of the neural network will thus be the parameters to be optimized. This modeling approach breaks the parameter number limit of the original TD-PCFG, so we can control the total number of parameters flexibly. When the total number of symbols is small, we can over-parameterize the model as over-parameterization has been shown to ease optimization (Arora et al., 2018; Xu et al., 2018; Du et al., 2019). On the other hand, when the total number of symbols is huge, we can decrease the number of parameters to save GPU memories and speed up training.

Specifically, we use neural networks to generate the following set of parameters of a TD-PCFG:

$$\Theta = \{ \mathbf{U}, \mathbf{V}, \mathbf{W}, \mathbf{Q}, \mathbf{r} \}.$$

The resulting model is referred to as neural PCFGs based on tensor decomposition (TN-PCFGs).

We start with the neural parameterization of  $\mathbf{U} \in \mathbb{R}^{n \times d}$  and  $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{m \times d}$ . We use shared symbol embeddings  $\mathbf{E}_s \in \mathbb{R}^{m \times k}$  ( $k$  is the symbol embedding dimension) in which each row is the embedding of a nonterminal or preterminal. We first compute an unnormalized  $\tilde{\mathbf{U}}$  by applying a neural network  $f_u(\cdot)$  to symbol embeddings  $\mathbf{E}_s$ :

$$\tilde{\mathbf{U}} = f_u(\mathbf{E}_s) = \left( \text{ReLU} \left( \mathbf{E}_s \mathbf{M}_u^{(1)} \right) \right) \mathbf{M}_u^{(2)},$$

where  $\mathbf{M}_u^{(1)} \in \mathbb{R}^{k \times k}$  and  $\mathbf{M}_u^{(2)} \in \mathbb{R}^{k \times d}$  are learnable parameters of  $f_u(\cdot)$ . For simplicity, we omit the learnable bias terms. We compute unnormalized  $\tilde{\mathbf{V}}$  and  $\tilde{\mathbf{W}}$  in a similar way. Note that only  $\mathbf{E}_s$  is shared in computing the three unnormalized matrices. Then we apply the Softmax activation function to each row of  $\tilde{\mathbf{U}}$  and to each column of  $\tilde{\mathbf{V}}$  and  $\tilde{\mathbf{W}}$ , and obtain normalized  $\mathbf{U}$ ,  $\mathbf{V}$ , and  $\mathbf{W}$ .

For preterminal-rule probabilities  $\mathbf{Q} \in \mathbb{R}^{p \times q}$  and start-rule probabilities  $\mathbf{r} \in \mathbb{R}^n$ , we follow (Kim et al., 2019a) and define them as:

$$\begin{aligned} \mathbf{Q}_{h_T, h_w} &= \pi_{T \rightarrow w} = \frac{\exp(\mathbf{u}_w^T f_t(\mathbf{w}_T))}{\sum_{w' \in \Sigma} \exp(\mathbf{u}_{w'}^T f_t(\mathbf{w}_T))}, \\ \mathbf{r}_{h_A} &= \pi_{S \rightarrow A} = \frac{\exp(\mathbf{u}_A^T f_s(\mathbf{w}_S))}{\sum_{A' \in \mathcal{N}} \exp(\mathbf{u}_{A'}^T f_s(\mathbf{w}_S))}, \end{aligned}$$

where  $\mathbf{w}$  and  $\mathbf{u}$  are symbol embeddings;  $f_s(\cdot)$  and  $f_t(\cdot)$  are neural networks that encode the input into a vector (see details in Kim et al. (2019a)). Note that the symbol embeddings are not shared between preterminal rules and start rules.

## 6 Parsing with TD-PCFGs

Parsing seeks the most probable parse  $t^\star$  from all the parses  $T_G(\mathbf{w})$  of a sentence  $\mathbf{w}$ :

$$t^\star = \arg \max_{t \in T_G(\mathbf{w})} p(t|\mathbf{w}). \quad (9)$$

Typically, the CYK algorithm<sup>3</sup> can be directly used to solve this problem exactly: it first computes the score of the most likely parse; and then automatic differentiation is applied to recover the best tree structure  $t^\star$  (Eisner, 2016; Rush, 2020). This, however, relies on the original probability tensor  $\mathbf{T}$  and is incompatible with our decomposed representation.<sup>4</sup> If we reconstruct  $\mathbf{T}$  from  $\mathbf{U}$ ,  $\mathbf{V}$ ,  $\mathbf{W}$  and then perform CYK, then the resulting time and space complexity would degrade to  $\mathcal{O}(m^3l^3)$  and become unaffordable when  $m$  is large. Therefore, we resort to Minimum Bayes-Risk (MBR) style decoding because we can compute the inside probabilities efficiently.

Our decoding method consists of two stages. The first stage computes the conditional probability of a substring  $\mathbf{w}_{i,j}$  being a constituent in a given sentence  $\mathbf{w}$  (a.k.a. posteriors of spans being a constituent):

$$p(\mathbf{w}_{i,j}|\mathbf{w}) = \frac{1}{p(\mathbf{w})} \sum_{t \in T_G(\mathbf{w})} p(t) \cdot \mathbb{1}_{\{\mathbf{w}_{i,j} \in t\}}.$$

We can estimate the posteriors efficiently by using automatic differentiation after obtaining all the inside probabilities. This has the same time complexity as our improved inside algorithm, which is  $\mathcal{O}(dl^3 + mdl^2)$ . The second stage uses the CYK algorithm to find the parse tree that has the highest expected number of constituents (Smith and Eisner, 2006):

$$t^\star = \arg \max_{t \in T_G(\mathbf{w})} \sum_{\mathbf{w}_{i,j} \in t} p(\mathbf{w}_{i,j}|\mathbf{w}). \quad (10)$$

<sup>3</sup>The CYK algorithm is similar to the inside algorithm. The only difference is that it uses MAX whenever the inside algorithm performs SUM over  $k$  and  $B, C$  (cf. Equation 1).

<sup>4</sup>In Equation 8 all symbols become entangled through  $\mathbf{V}^T \mathbf{s}_{i,k}$  and  $\mathbf{W}^T \mathbf{s}_{k+1,j}$ . We are unable to perform MAX over  $B, C$  as in the CYK algorithm.

The time complexity of the second stage is  $\mathcal{O}(l^3)$ , so the overall time complexity of our decoding method is  $\mathcal{O}(dl^3 + mdl^2)$ , which is much faster than  $\mathcal{O}(m^3l^3)$  in general.

## 7 Experimental setup

### 7.1 Datasets

We evaluate TN-PCFGs across ten languages. We use the Wall Street Journal (WSJ) corpus of the Penn Treebank (Marcus et al., 1994) for English, the Penn Chinese Treebank 5.1 (CTB) (Xue et al., 2005) for Chinese, and the SPRML dataset (Seddah et al., 2014) for the other eight morphology-rich languages. We use a unified data preprocessing pipeline<sup>5</sup> provided by Zhao and Titov (2021). The same pipeline has been used in several recent papers (Shen et al., 2018a, 2019; Kim et al., 2019a; Zhao and Titov, 2020). Specifically, for every treebank, punctuation is removed from all data splits and the top 10,000 frequent words in the training data are used as the vocabulary.

### 7.2 Settings and hyperparameters

For baseline models we use the best configurations reported by the authors. For example, we use 30 nonterminals and 60 preterminals for N-PCFGs and C-PCFGs. We implement TN-PCFGs and reimplement N-PCFGs and C-PCFGs using automatic differentiation (Eisner, 2016) and we borrow the idea of Zhang et al. (2020) to batchify the inside algorithm. Inspired by Kim et al. (2019a), for TN-PCFGs we set  $n/p$ , the ratio of the nonterminal number to the preterminal number, to 1/2. For  $\mathbf{U} \in \mathbb{R}^{n \times d}$  and  $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{m \times d}$  we set  $d = p$  when there are more than 200 preterminals and  $d = 200$  otherwise. The symbol embedding dimension  $k$  is set to 256. We optimize TN-PCFGs using the Adam optimizer (Kingma and Ba, 2015) with  $\beta_1 = 0.75$ ,  $\beta_2 = 0.999$ , and learning rate 0.001 with batch size 4. We use the unit Gaussian distribution to initialize embedding parameters. We do not use the curriculum learning strategy that is used by Kim et al. (2019a) when training TN-PCFGs.

### 7.3 Evaluation

Following Kim et al. (2019a), we train a TN-PCFG for each treebank separately. For each setting we run the TN-PCFG and the baselines four times with different random seeds and for ten epochs each

<sup>5</sup><https://github.com/zhaoyanpeng/xcfg>.

Model	WSJ	
	Mean	Max
Left Branching		8.7
Right Branching		39.5
Random Trees	18.1	18.2
Systems without pretrained word embeddings		
PRPN <sup>†</sup> (Shen et al., 2018a)	47.3	47.9
ON <sup>†</sup> (Shen et al., 2019)	48.1	50.0
N-PCFG (Kim et al., 2019a)	50.8	52.6
C-PCFG (Kim et al., 2019a)	55.2	60.1
NL-PCFG (Zhu et al., 2020)	55.3	
N-PCFG <sup>*</sup>	50.9 <sub>±2.3</sub>	54.6
N-PCFG <sup>*</sup> w/ MBR	52.3 <sub>±2.3</sub>	55.8
C-PCFG <sup>*</sup>	55.4 <sub>±2.2</sub>	59.0
C-PCFG <sup>*</sup> w/ MBR	56.3 <sub>±2.1</sub>	60.0
TN-PCFG $p = 60$ (ours)	51.4 <sub>±4.0</sub>	55.6
TN-PCFG $p = 500$ (ours)	<b>57.7<sub>±4.2</sub></b>	<b>61.4</b>
Systems with pretrained word embeddings		
DIORA (Drozdov et al., 2019)		56.8
S-DIORA (Drozdov et al., 2020)	57.6	64.0
CT (Cao et al., 2020)	62.8	65.9
Oracle Trees		84.3

Table 1: Unlabeled sentence-level F1 scores on the WSJ test data. <sup>†</sup> indicates numbers reported by Kim et al. (2019a). <sup>\*</sup> indicates our reimplementations of N-PCFGs and C-PCFGs.  $p$  denotes the preterminal number.

time. Early stopping is performed based on the perplexity on the development data. The best model in each run is selected according to the perplexity on the development data. We tune model hyperparameters only on the development data of WSJ and use the same model configurations on the other treebanks.<sup>6</sup> We report average sentence-level F1 score<sup>7</sup> as well as their biased standard deviations.

## 8 Experimental results

We evaluate our models mainly on WSJ (Section 8.1-8.3). We first give an overview of model

<sup>6</sup>Shi et al. (2020) suggest not using the gold parses of the development data for hyperparameter tuning and model selection in unsupervised parsing. Here we still use the gold parses of the WSJ development set for the English experiments in order to conduct fair comparison with previous work. No gold parse is used in the experiments of any other language.

<sup>7</sup>Following Kim et al. (2019a), 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.

Model	Time (minutes)	Total Parameters (M)
N-PCFG	19	5.3
C-PCFG	20	15.4
TN-PCFG ( $p = 60$ )	13	3.6
TN-PCFG ( $p = 500$ )	26	4.2

Table 2: Average running time per epoch and the parameter number of each model.

performance in Section 8.1 and then conduct ablation study of TN-PCFGs in Section 8.2. We quantitatively and qualitatively analyze constituent labels induced by TN-PCFGs in Section 8.3. In Section 8.4, we conduct a multilingual evaluation over nine additional languages.

### 8.1 Main results

Our best TN-PCFG model uses 500 preterminals ( $p = 500$ ). We compare it with a wide range of recent unsupervised parsing models (see the top section of Table 1). Since we use MBR decoding for TN-PCFGs, which produces higher F1-measure than the CYK decoding (Goodman, 1996), for fair comparison we also use MBR decoding for our reimplemented N-PCFGs and C-PCFGs (see the middle section of Table 1).

We draw three key observations from Table 1: (1) TN-PCFG ( $p = 500$ ) achieves the best mean and max F1 score. Notably, it outperforms the strong baseline model C-PCFG by 1.4% mean F1. Compared with TN-PCFG ( $p = 60$ ), TN-PCFG ( $p = 500$ ) brings a 6.3% mean F1 improvement, demonstrating the effectiveness of using more symbols. (2) Our reimplementations of N-PCFGs and C-PCFGs are comparable to those of Kim et al. (2019a), (3) MBR decoding indeed gives higher F1 scores (+1.4% mean F1 for N-PCFG and +0.9% mean F1 for C-PCFG).

In Table 1 we also show the results of Constituent test (CT) (Cao et al., 2020) and DIORA (Drozdov et al., 2019, 2020), two recent state-of-the-art approaches. However, our work is not directly comparable to these approaches. CT relies on pretrained language models (RoBERTa) and DIORA relies on pretrained word embeddings (context insensitive ELMo). In contrast, our model and the other approaches do not use pretrained word embeddings and instead learn word embeddings from scratch. We are also aware of URNNG (Kim et al., 2019b), which has a max F1 score of 45.4%, but it uses punctuation and hence is not directly comparable to the models listed in the table.

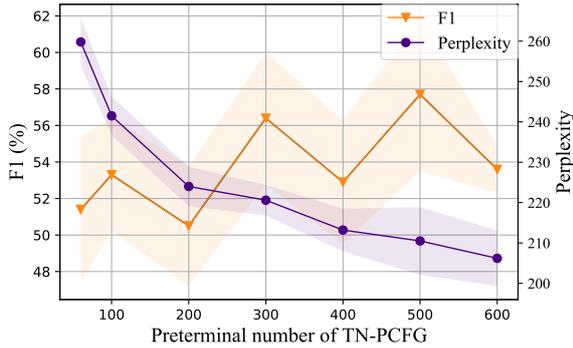


Figure 1: F1 scores and perplexities w.r.t. the preterminal number  $p$  of TN-PCFGs on the WSJ test data. Recall that the nonterminal number  $n$  is set to half of  $p$ .

We report the average running time<sup>8</sup> per epoch and the parameter numbers of different models in Table 2. We can see that TN-PCFG ( $p = 500$ ), which uses a much larger number of symbols, has even fewer parameters and is not significantly slower than N-PCFG.

## 8.2 Influence of symbol number

Figure 1 illustrates the change of F1 scores and perplexities as the number of nonterminals and preterminals increase. We can see that, as the symbol number increases, the perplexities decrease while F1 scores tend to increase.

## 8.3 Analysis on constituent labels

We analyze model performance by breaking down recall numbers by constituent labels (Table 3). We use the top six frequent constituent labels in the WSJ test data (NP, VP, PP, SBAR, ADJP, and ADVP). We first observe that the right-branching baseline remains competitive. It achieves the highest recall on VPs and SBARs. TN-PCFG ( $p = 500$ ) displays a relatively even performance across the six labels. Specifically, it performs best on NPs and PPs among all the labels and it beats all the other models on ADJPs. Compared with TN-PCFG ( $p = 60$ ), TN-PCFG ( $p = 500$ ) results in the largest improvement on VPs (+19.5% recall), which are usually long (with an average length of 11) in comparison with the other types of constituents. As NPs and VPs cover about 54% of the total constituents in the WSJ test data, it is not surprising that models which are accurate on these labels have high F1 scores (e.g., C-PCFGs and TN-PCFGs ( $p = 500$ )).

<sup>8</sup>We measure the running time on a single Titan V GPU.

We further analyze the correspondence between the nonterminals of trained models and gold constituent labels. For each model, we look at all the correctly-predicted constituents in the test set and estimate the empirical posterior distribution of nonterminals assigned to a constituent given the gold label of the constituent (see Figure 2). Compared with the other three models, in TN-PCFG ( $p = 500$ ), the most frequent nonterminals are more likely to correspond to a single gold label. One possible explanation is that it contains much more nonterminals and therefore constituents of different labels are less likely to compete for the same nonterminal.

Figure 2d (TN-PCFG ( $p = 500$ )) also illustrates that a gold label may correspond to multiple nonterminals. A natural question that follows is: *do these nonterminals capture different subtypes of the gold label?* We find it is indeed the case for some nonterminals. Take the gold label NPs (noun phrases), while not all the nonterminals have clear interpretation, we find that NT-3 corresponds to constituents which represent a company name; NT-99 corresponds to constituents which contain a possessive affix (e.g., “s” in “the market’s decline”); NT-94 represents constituents preceded by an indefinite article. We further look into the gold label PPs (preposition phrases). Interestingly, NT-108, NT-175, and NT-218 roughly divided preposition phrases into three groups starting with ‘with, by, from, to’, ‘in, on, for’, and ‘of’, respectively. See Appendix for more examples.

## 8.4 Multilingual evaluation

In order to understand the generalizability of TD-PCFGs on languages beyond English, we conduct a multilingual evaluation of TD-PCFGs on CTB and SPMRL. We use the best model configurations obtained on the English development data and do not perform any further tuning on CTB and SPMRL. We compare TN-PCFGs with N-PCFGs and C-PCFGs and use MBR decoding by default. The results are shown in Table 4. In terms of the average F1 over the nine languages, all the three models beat trivial left- and right-branching baselines by a large margin, which suggests they have good generalizability on languages beyond English. Among the three models, TN-PCFG ( $p = 500$ ) fares best. It achieves the highest F1 score on six out of nine treebanks. On Swedish, N-PCFG is worse than the right-branching baseline (-13.4% F1), while

Model	NP	VP	PP	SBAR	ADJP	ADVP	S-F1
Left Branching	10.4	0.5	5.0	5.3	2.5	8.0	8.7
Right Branching	24.1	<b>71.5</b>	42.4	<b>68.7</b>	27.7	38.1	39.5
Random Trees	22.5 $\pm 0.3$	12.3 $\pm 0.3$	19.0 $\pm 0.5$	9.3 $\pm 0.6$	24.3 $\pm 1.7$	26.9 $\pm 1.3$	18.1 $\pm 0.1$
NPCFG <sup>†</sup>	71.2	33.8	58.8	52.5	32.5	45.5	50.8
C-NPCFG <sup>†</sup>	74.7	41.7	68.8	56.1	40.4	52.5	55.2
N-PCFG* w/ MBR	72.3 $\pm 3.6$	28.1 $\pm 6.6$	<b>73.0</b> $\pm 2.6$	53.6 $\pm 10.0$	40.8 $\pm 4.2$	43.8 $\pm 8.3$	52.3 $\pm 2.3$
C-PCFG* w/ MBR	73.6 $\pm 2.5$	45.0 $\pm 6.0$	71.4 $\pm 1.4$	54.8 $\pm 5.6$	44.3 $\pm 5.9$	<b>61.6</b> $\pm 17.6$	56.3 $\pm 2.1$
TN-PCFG $p = 60$	<b>77.2</b> $\pm 2.6$	28.9 $\pm 13.8$	58.9 $\pm 17.5$	44.0 $\pm 17.6$	47.5 $\pm 5.1$	54.9 $\pm 6.0$	51.4 $\pm 4.0$
TN-PCFG $p = 500$	75.4 $\pm 4.9$	48.4 $\pm 10.7$	67.0 $\pm 11.7$	50.3 $\pm 13.3$	<b>53.6</b> $\pm 3.3$	59.5 $\pm 2.6$	<b>57.7</b> $\pm 4.2$

Table 3: Recall on the six frequent constituent labels in the WSJ test data, <sup>†</sup> denotes results reported by Kim et al. (2019a), S-F1 represents sentence-level F1 measure.

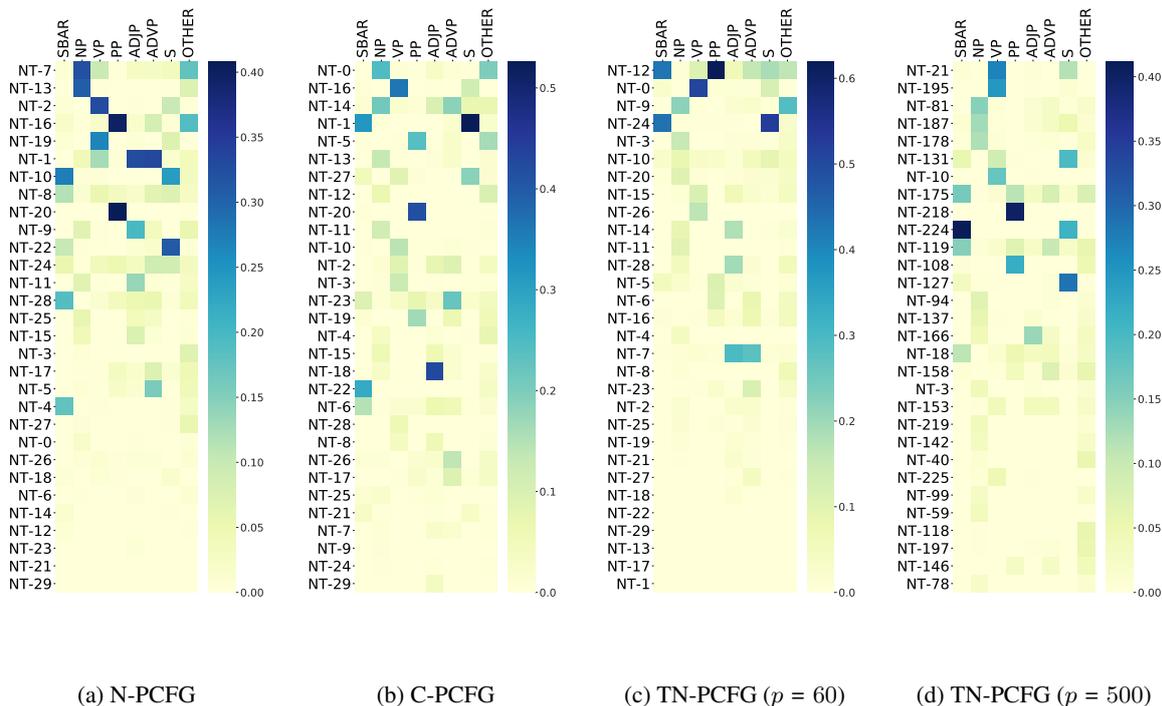


Figure 2: Correspondence between nonterminals and gold constituent labels. For each gold label, we visualize the proportion of correctly-predicted constituents that correspond to each nonterminal. Nonterminals NT-# are listed in descending order of the prediction frequency (e.g., NT-21 is the most predicted nonterminal in TN-PCFG ( $p = 500$ )). Only the top 30 frequent nonterminals are listed. We show the seven most frequent gold labels and aggregate the rest (denoted by OTHER).

Model	Chinese	Basque	German	French	Hebrew	Hungarian	Korean	Polish	Swedish	Mean
Left Branching <sup>†</sup>	7.2	17.9	10.0	5.7	8.5	13.3	18.5	10.9	8.4	11.2
Right Branching <sup>†</sup>	25.5	15.4	14.7	26.4	30.0	12.7	19.2	34.2	30.4	23.2
Random Trees <sup>†</sup>	15.2	19.5	13.9	16.2	19.7	14.1	22.2	21.4	16.4	17.6
N-PCFG w/ MBR	26.3 $\pm 2.5$	35.1 $\pm 2.0$	42.3 $\pm 1.6$	<b>45.0</b> $\pm 2.0$	<b>45.7</b> $\pm 2.2$	43.5 $\pm 1.2$	28.4 $\pm 6.5$	43.2 $\pm 0.8$	17.0 $\pm 9.9$	36.3
C-PCFG w/ MBR	38.7 $\pm 6.6$	<b>36.0</b> $\pm 1.2$	43.5 $\pm 1.2$	<b>45.0</b> $\pm 1.1$	45.2 $\pm 0.5$	<b>44.9</b> $\pm 1.5$	30.5 $\pm 4.2$	43.8 $\pm 1.3$	33.0 $\pm 15.4$	40.1
TN-PCFG $p = 500$	<b>39.2</b> $\pm 5.0$	<b>36.0</b> $\pm 3.0$	<b>47.1</b> $\pm 1.7$	39.1 $\pm 4.1$	39.2 $\pm 10.7$	43.1 $\pm 1.1$	<b>35.4</b> $\pm 2.8$	<b>48.6</b> $\pm 3.1$	<b>40.0</b> $\pm 4.8$	<b>40.9</b>

Table 4: Sentence-level F1 scores on CTB and SPMRL. <sup>†</sup> denotes results reported by Zhao and Titov (2021).

TN-PCFG ( $p = 500$ ) surpasses the right-branching baseline by 9.6% F1.

## 9 Discussions

In our experiments, we do not find it beneficial to use the compound trick (Kim et al., 2019a) in TN-PCFGs, which is commonly used in previous work of PCFG induction (Kim et al., 2019a; Zhao

and Titov, 2020; Zhu et al., 2020). We speculate that the additional expressiveness brought by compound parameterization may not be necessary for a TN-PCFG with many symbols which is already sufficiently expressive; on the other hand, compound parameterization makes learning harder when we use more symbols.

We also find neural parameterization and the choice of nonlinear activation functions greatly influence the performance. Without using neural parameterization, TD-PCFGs have only around 30% S-F1 scores on WSJ, which are even worse than the right-branching baseline. Activation functions other than ReLU (such as tanh and sigmoid) result in much worse performance. It is an interesting open question why ReLU and neural parameterization are crucial in PCFG induction.

When evaluating our model with a large number of symbols, we find that only a small fraction of the symbols are predicted in the parse trees (for example, when our model uses 250 nonterminals, only tens of them are found in the predicted parse trees of the test corpus). We expect that our models can benefit from regularization techniques such as state dropout (Chiu and Rush, 2020).

## 10 Conclusion

We have presented TD-PCFGs, a new parameterization form of PCFGs based on tensor decomposition. TD-PCFGs rely on Kruskal decomposition of the binary-rule probability tensor to reduce the computational complexity of PCFG representation and parsing from cubic to at most quadratic in the symbol number, which allows us to scale up TD-PCFGs to a much larger number of (nonterminal and preterminal) symbols. We further propose neurally parameterized TD-PCFGs (TN-PCFGs) and learn neural networks to produce the parameters of TD-PCFGs. On WSJ test data, TN-PCFGs outperform strong baseline models; we empirically show that using more nonterminal and preterminal symbols contributes to the high unsupervised parsing performance of TN-PCFGs. Our multilingual evaluation on nine additional languages further reveals the capability of TN-PCFGs to generalize to languages beyond English.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (61976139).

## References

- Sanjeev Arora, Nadav Cohen, and Elad Hazan. 2018. [On the optimization of deep networks: Implicit acceleration by overparameterization](#). In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 244–253. PMLR.
- Rares-Darius Buhai, Yoni Halpern, Yoon Kim, Andrej Risteski, and David Sontag. 2019. Empirical study of the benefits of overparameterization in learning latent variable models. *arXiv preprint arXiv:1907.00030*.
- 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.
- Glenn Carroll and Eugene Charniak. 1992. *Two experiments on learning probabilistic dependency grammars from corpora*. Department of Computer Science, Univ.
- 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.
- Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. [Spectral learning of latent-variable PCFGs](#). In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 223–231, Jeju Island, Korea. Association for Computational Linguistics.
- 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. 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 Lan-*

- guage Technologies, Volume 1 (Long and Short Papers), pages 1129–1141, Minneapolis, Minnesota. Association for Computational Linguistics.
- Simon S. Du, Xiyu Zhai, Barnabás Póczos, and Aarti Singh. 2019. [Gradient descent provably optimizes over-parameterized neural networks](#). In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.
- 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.
- Joshua Goodman. 1996. [Parsing algorithms and metrics](#). In *34th Annual Meeting of the Association for Computational Linguistics*, pages 177–183, Santa Cruz, California, USA. 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.
- 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, 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 3rd International Conference on Learning Representations*.
- 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.
- 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.
- Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. [The infinite PCFG using hierarchical Dirichlet processes](#). In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697, Prague, Czech Republic. 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*.
- Takuya Matsuzaki, Yusuke Miyao, and Jun’ichi Tsujii. 2005. [Probabilistic CFG with latent annotations](#). In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*, pages 75–82, Ann Arbor, Michigan. Association for Computational Linguistics.
- Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. [Learning accurate, compact, and interpretable tree annotation](#). In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia. Association for Computational Linguistics.
- Danilo Jimenez Rezende and Shakir Mohamed. 2015. [Variational inference with normalizing flows](#). In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1530–1538. JMLR.org.
- 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.
- Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. [Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages](#). In *Proceedings*

- of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages, pages 103–109, Dublin, Ireland. Dublin City University.
- Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018a. [Neural language modeling by jointly learning syntax and lexicon](#). In *International Conference on Learning Representations*.
- Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. [Ordered neurons: Integrating tree structures into recurrent neural networks](#). In *International Conference on Learning Representations*.
- Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2018b. [Efficient attention: Attention with linear complexities](#). *arXiv preprint arXiv:1812.01243*.
- Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2020. [On the role of supervision in unsupervised constituency parsing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7611–7621, Online. Association for Computational Linguistics.
- David A. Smith and Jason Eisner. 2006. [Minimum risk annealing for training log-linear models](#). In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. [Recursive deep models for semantic compositionality over a sentiment treebank](#). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.
- Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. [Unsupervised neural hidden Markov models](#). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71, Austin, TX. Association for Computational Linguistics.
- Ji Xu, Daniel J. Hsu, and Arian Maleki. 2018. [Benefits of over-parameterization with EM](#). In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 10685–10695.
- Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. [The penn chinese treebank: Phrase structure annotation of a large corpus](#). *Natural Language Engineering*, 11(2):207–238.
- Songlin Yang, Yong Jiang, Wenjuan Han, and Kewei Tu. 2020. [Second-order unsupervised neural dependency parsing](#). In *COLING*.
- 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-20*, pages 4046–4053. International Joint Conferences on Artificial Intelligence Organization. Main track.
- 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.
- Yanpeng Zhao and Ivan Titov. 2021. [An empirical study of compound pcfgs](#). *arXiv preprint arXiv:2103.02298*.
- Yanpeng Zhao, Liwen Zhang, and Kewei Tu. 2018. [Gaussian mixture latent vector grammars](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1189, Melbourne, Australia. Association for Computational Linguistics.
- Hao Zhu, Yonatan Bisk, and Graham Neubig. 2020. [The return of lexical dependencies: Neural lexicalized pcfgs](#). *CoRR*, abs/2007.15135.

## Appendix

To evaluate whether the nonterminals of trained models capture meaningful constituent clusters, we predict the unlabeled parse trees in the test set using MBR decoding, then choose the correctly predicted constituents and assign the maximum a posterior (MAP) nonterminal symbols to them. Table 5 lists some representative nonterminals and their example constituents.

Nonterminal	Description	Constituents
NT-3	Company name	Quantum Chemical Corp. / Dow Chemical Co. / Union Carbide Corp. / First Boston Corp. / Petrolane Inc. / British Petroleum Co. / Ford Motor Co. / Jaguar PLC / Jaguar shares / General Motors Corp. / Northern Trust Co. / Norwest Corp. / Fidelity Investments / Windsor Fund 's / Vanguard Group Inc.
NT-99	Possessive affix	Quantum 's shares / Quantum 's fortunes / year 's end / the company 's sway / September 's steep rise / the stock market 's plunge / the market 's decline / this year 's good inflation news / the nation 's largest fund company / Fidelity 's stock funds / Friday 's close / today 's closing prices / this year 's fat profits
NT-94	Infinite article	an acquisition of Petrolane Inc. / a transaction / a crisis / a bid / a minority interest / a role / a 0.7 % gain / an imminent easing of monetary policy / a blip / a steep run-up/ a steeper rise / a substantial rise / a broad-based advance / a one-time event / a buffer / a group / an opportunity / a buying opportunity
NT-166	Number	NP 99.8 million / 34.375 share /85 cents /a year ago /36 % / 1.18 billion /15 % / eight members / 0.9 % / 32.82 billion / 1.9 % / 1982 = 100 / about 2 billion / 8.8 % and 9.2 % / Ten points / 167.7 million shares / 20,000 shares / about 2,000 stores / up to 500 a square foot / 17 each / its 15 offering price / 190.58 points
NT-59 / 142 / 219	Determinater	The timing / nearly a quarter / the bottom / the troughs / the chemical building block / only about half / the start / the day / the drawn-out process / the possibility / the actions / the extent / the start / the pace / the beginning / the prices / less than 15 % / two years / the volume / the chaos / the last hour
NT-218	of	of plastics / of the chemical industry / of the boom / of 99.8 million or 3.92 a share / of its value / of Quantum 's competitors / of the heap / of its resources / of the plastics market / of polyethylene / its ethylene needs / of a few cents a pound / of intense speculation / of retail equity trading / of stock / of Securities Dealers
NT-108	from, by, with, to	to Mr. Hardiman / to pre-crash levels / through the third quarter / from institutions / through the market / to Contel Cellular / with investors / to the junk bond market / with borrowed money / by the House / from the deficit-reduction bill / by the end of the month / to the streamlined bill / by the Senate action / with the streamlined bill / from the Senate bill / to top paid executives / like idiots
NT-175	for, on, in	for the ride / on plastics / for plastics producers / on plastics / for this article / at Oppenheimer & Co. / on Wall Street / in a Sept. 29 meeting with analysts / in Morris III / in the hospital / between insured and insurer / for Quantum / in the U.S. / in the market 's decline / top of a 0.7 % gain in August / in the economy / in the producer price index / in the prices of consumer and industrial goods
NT-158	Start with PP	Within the next year / To make its point / summer / Under the latest offer / At this price / In March / In the same month / Since April / In the third quarter / Right now / As she calculates it / Unlike 1987 course / in both instances / in the October 1987 crash / In 1984 / Under the agreement / In addition / In July / In computer publishing / On the other hand / In personal computers / At the same time / As a result of this trend / Beginning in mid-1987 / In addition
NT-146	PP in Sentence	in August / in September / in July / for September / after Friday 's close / in the House-passed bill / in the Senate bill / in fiscal 1990 / in tax increases / over five years / in air conditioners / in Styrofoam / in federal fees / in fiscal 1990 / in Chicago / in Japan / in Europe in Denver / at Salomon Bros / in California / in principle / in August / in operating costs / in Van Nuys / in San Diego / in Texas / in the dollar / in the past two years / in Dutch corporate law / since July / in 1964 / in June / for instance / in France / in Germany / in Palermo / for quotas

Table 5: Nonterminals and the corresponding example constituents.