PCFGs Can Do Better

Inducing Probabilistic Context-Free Grammars with Many Symbols

Songlin Yang, Yanpeng Zhao, Kewei Tu

May 10, 2021

ShanghaiTech University, University of Edinburgh

Introduction

PCFG definition

CFG is a 5-tuple: $\mathcal{G} = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R})$

where

S : start symbol

$\mathcal{N}$: nonterminals (constituent labels)
$\mathcal{P}$: preterminals (POS tags)
} *symbols*

Σ : terminals (words)

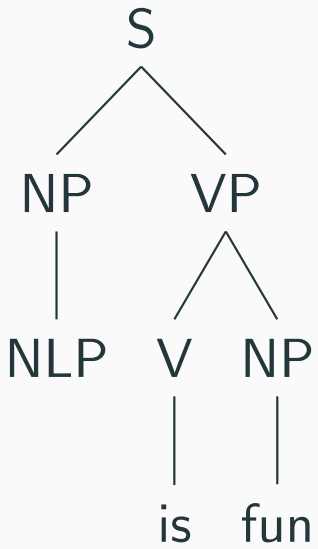
$\mathcal{R}$ is a set of grammar rules, where $r \in \mathcal{R}$ is one of the following form:

$S \to A$		$A \in \mathcal{N}$	start rule
$A \to BC$	$A \in \mathcal{N},$	$B, C \in \mathcal{N} \cup \mathcal{P}$	binary rule
$T \to w$		$T \in \mathcal{P}, w \in \Sigma$	unary rule

PCFGs extend CFGs by associating each rule $r \in \mathcal{R}$ with a probability π_r.

Tree probability

$p(t) = \prod_{r \in t_{\mathcal{R}}} \pi_r$, where $t_{\mathcal{R}}$ is the set of rules used in the derivation of t.



$\text{p(t)} = \pi(S \to NP, VP) \times \pi(VP \to V, NP) \times \pi(NP \to NLP) \times \pi(V \to is) \times \pi(NP \to fun)$

PCFG Induction

Goal: Learn a PCFG from raw text alone.

Objective function: log-likelihood of sentence w

$$\log p_\theta(w) = \log \sum_{t \in T_{\mathcal{G}}(w)} p(t)\,,$$

which can be estimated via the inside algorithm in $O(l^3 m^3)$ time.

l: sentence length. m: symbol number.

Problem

High complexity, does not scale to m (due to the cubic complexity)

Previous work:

- 30 nonterminals, 60 preterminals (Compound PCFG) [4]
- 30 nonterminals, no additional preterminals (Depth-bounded PCFG) [3]
- 20 nonterminals, 40 preterminals (Neural Lexicalized PCFG) [5]

Motivations

- Classical work in supervised constituency parsing show that dividing grammar symbols into subtypes is helpful in increasing parsing performance.
- Recent works show that increasing the number of hidden states is helpful in latent-variable learning [1, 2]

We present a parameterization form of PCFGs based on tensor decomposition:

- lower complexity
- allowing a much larger number of symbols
- leading to better PCFG induction performance

Tensor decomposition on PCFGs

Tensor form of PCFGs

n: nonterminals number

p: preterminal number

q: terminals number

$m = n + p$

h: index

Tensor form of PCFG

Start rule:

$$S \to A \quad A \in \mathcal{N}$$

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

Tensor form of PCFG

Binary rule:

$$A \to BC, \quad A \in \mathcal{N}, \quad B, C \in \mathcal{N} \cup \mathcal{P}$$

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

Tensor form of PCFG

Unary rule:

$$T \to w, \quad T \in \mathcal{P}, w \in \Sigma$$

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

Tensor form of the inside algorithm

$$s_{i,j}^{A} = \sum_{k=i}^{j-1} \sum_{B,C} \pi_{A \to BC} \cdot s_{i,k}^{B} \cdot s_{k+1,j}^{C} \,. \tag{1}$$

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

Substitute $\pi_{A \to BC}$ with its tensor form:

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

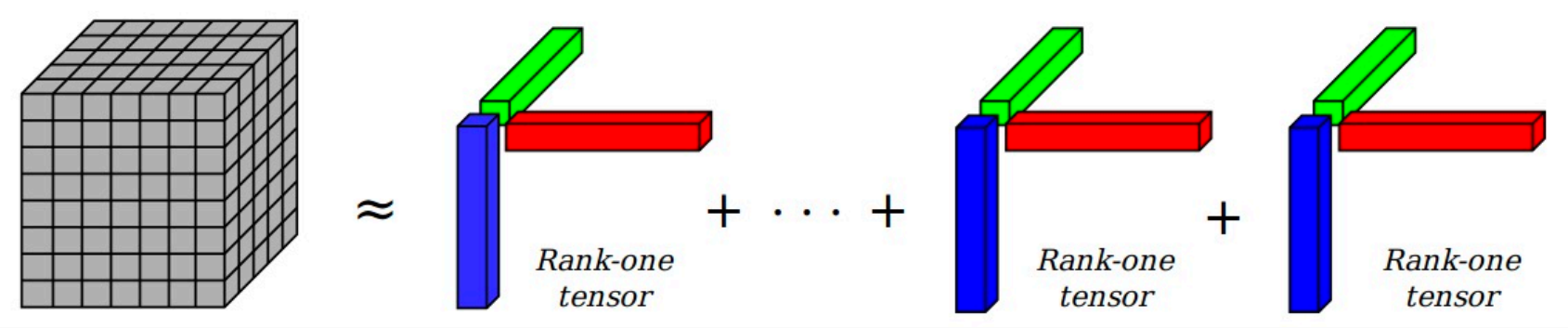
Then:

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

CP decomposition

Kruskal form:

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



CP decomposition on the inside algorithm

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

Substitute $\mathbf{T}$ using the Kruskal form:

$$\mathbf{s}_{i,j} = \mathbf{U} \cdot \sum_{k=i}^{j-1} \left(\left(\mathbf{V}^T \mathbf{s}_{i,k} \right) \odot \left(\mathbf{W}^T \mathbf{s}_{k+1,j} \right) \right) . \tag{5}$$

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 $\mathcal{O}(l^3 r + mrl^2)$, which is at most quadratic in m as we typically set $r = O(m)$.

Difference between Cohen's work and ours

- (Cohen etal, 2013) use the same technique to accelerate PCFG parsing.
- They use supervised training to obtain a PCFG model firstly, then use tensor decomposition to approximate the existing PCFG in order to accelerate inference.
- They actually perform the CP decomposition on an existing $\mathbf{T}$.
- Instead, we do not have $\mathbf{T}$ to start with in unsupervised learning.
- We learn the model end-to-end without explicit CP decomposition!

Problem

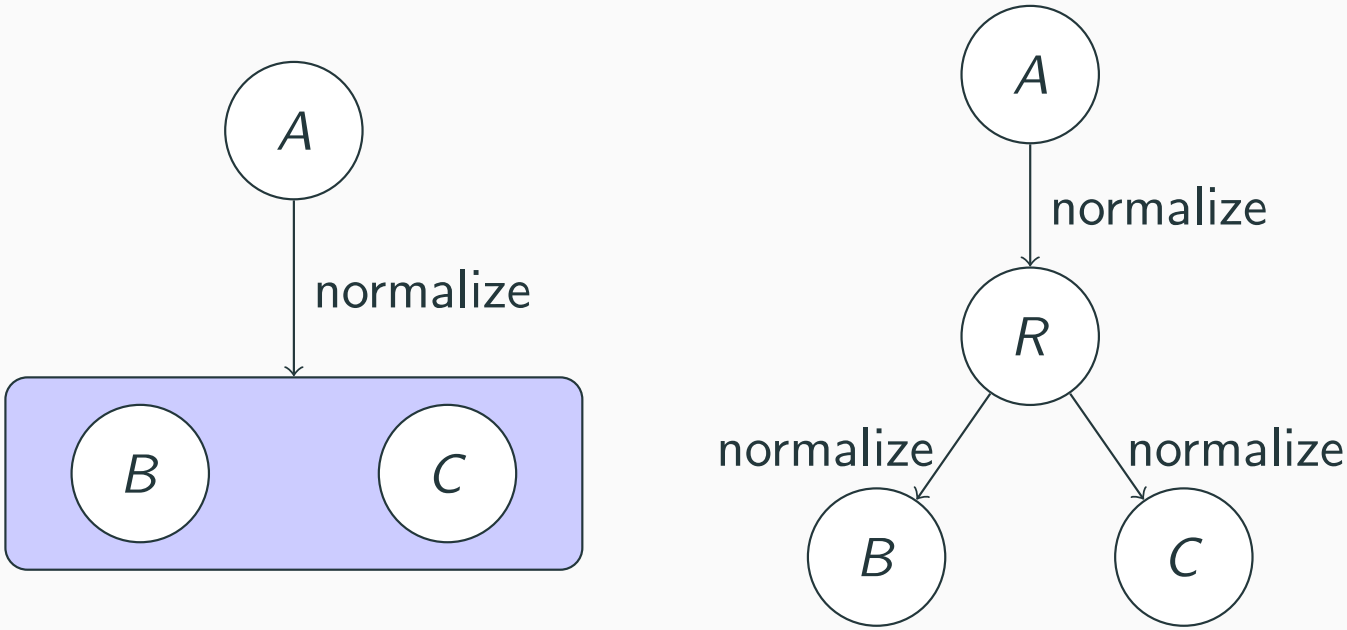
How can we make sure $\mathbf{T}$ is valid?

Specifically, how we can ensure that $\mathbf{T}$ is non-negative and properly normalized (i.e. $\sum_{j,k} \mathbf{T}_{h_A,j,k} = 1$ for given h_A).

Solution

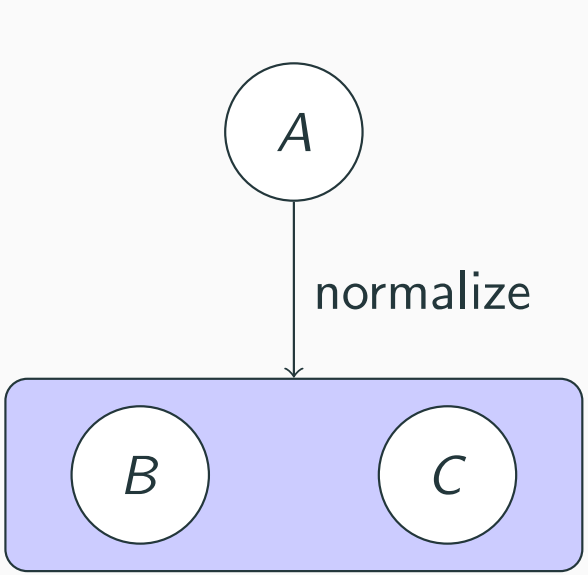
If $\mathbf{V}$ and $\mathbf{W}$ are column-normalized, $\mathbf{U}$ is row-normalized, then $\mathbf{T}$ is valid.

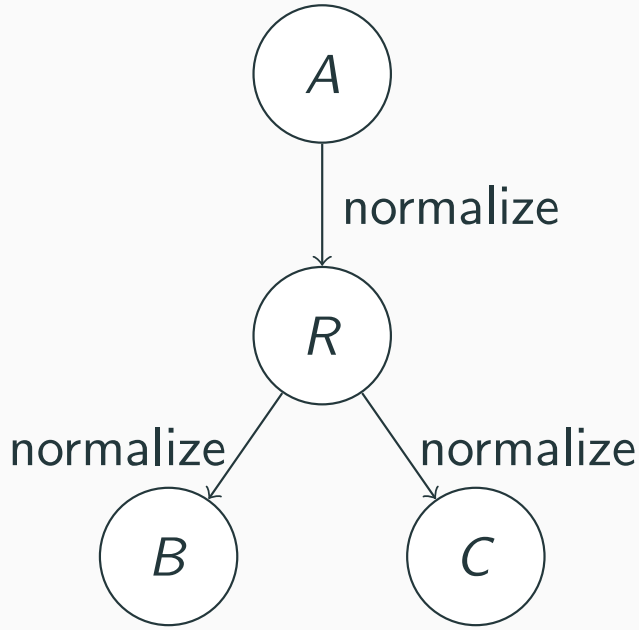
CP decomposition in probabilistic perspective



The newly introduced dimension in CP decomposition can be regarded as a latent variable R. Domain size $|R|$ = tensor rank.

CP decomposition in probabilistic perspective

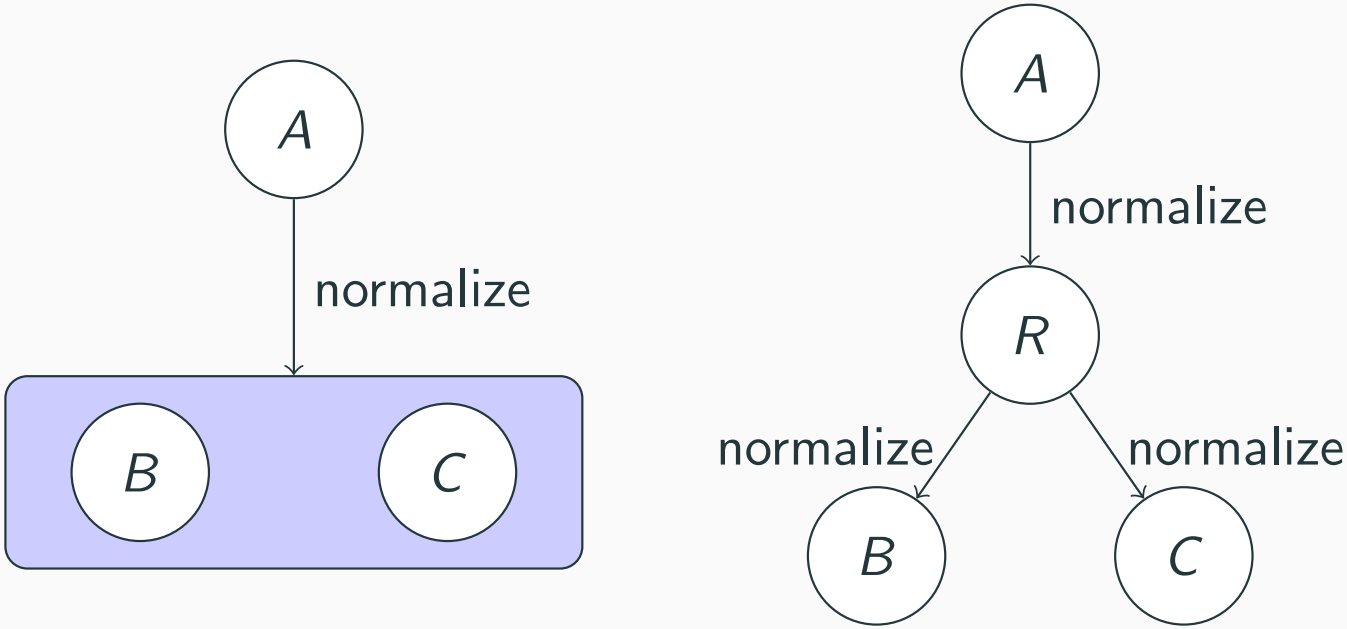




Left: normalize $A \to B, C$, total complexity $O(m^3)$.

CP decomposition in probabilistic perspective

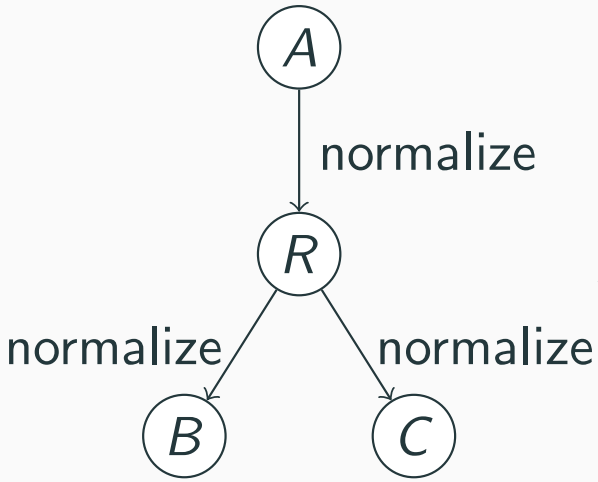
CP decomposition in probabilistic perspective:



Right: normalize $A \to R, R \to B, R \to C$, total complexity: $O(mr)$.

It is clear that $P(A \to B, C)$ is properly normalized (by reading the Bayesian graph).

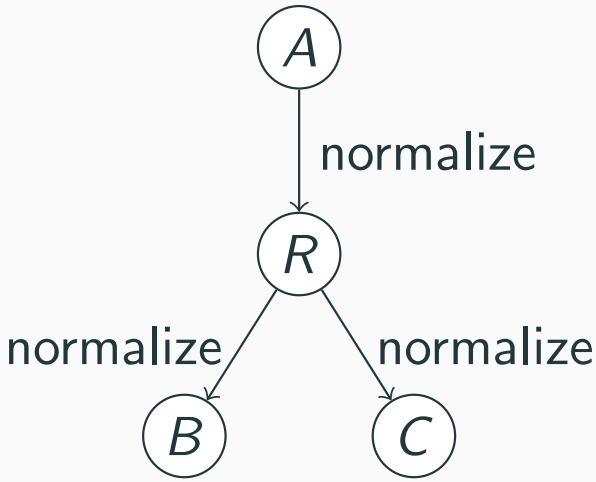
CP decomposition in probabilistic perspective



$$p(A \to B, C) = \sum_R p(A \to R)p(R \to B)p(R \to C)$$

We can obtain the binary rule probabilities by marginalizing over the latent variable R.

CP decomposition in probabilistic perspective



$$p(A \to B, C) = \sum_R p(A \to R)p(R \to B)p(R \to C)$$

$$\mathrm{p}(\mathrm{A} \to B, C) = \sum_{l=1}^{r} \mathbf{u}^{(l)} \otimes \mathbf{v}^{(l)} \otimes \mathbf{w}^{(l)}$$

Row normalization of U is equivalent to normalize $p(A \to R)$,

Column normalization of V, W is equivalent to normalize $p(R \to B), p(R \to C)$ respectively.

Neural Parameterization

Use distributed representation for each symbol. Use neural networks to calculate grammar rule probabilities.

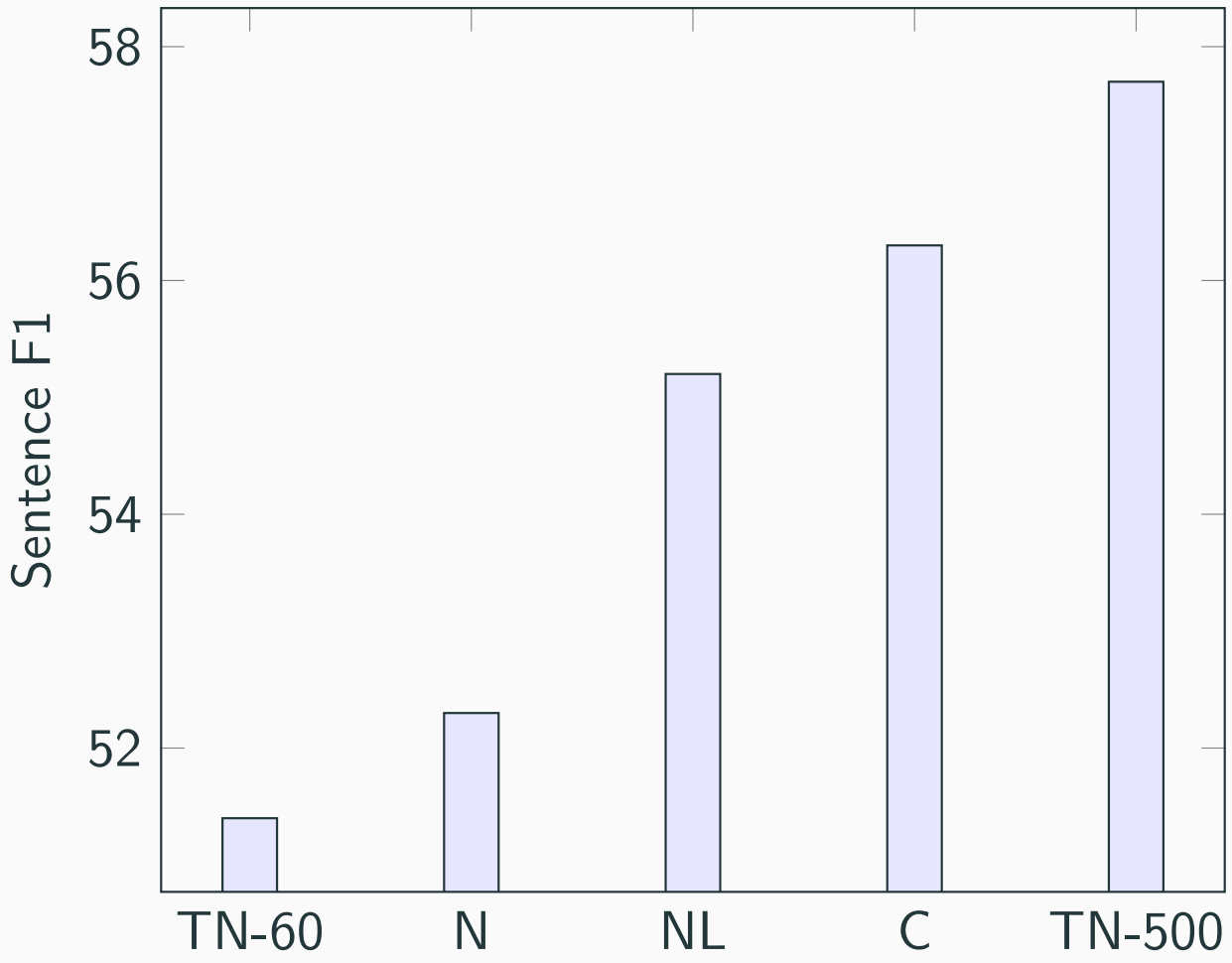
Use Softmax function to ensure the row/column normalization of $\mathbf{U}, \mathbf{V}, \mathbf{W}$.

Without neural parameterization, performance drops significantly (even underperfroms right-branching baseline in WSJ)

Activation functions other than ReLU perform much worse.

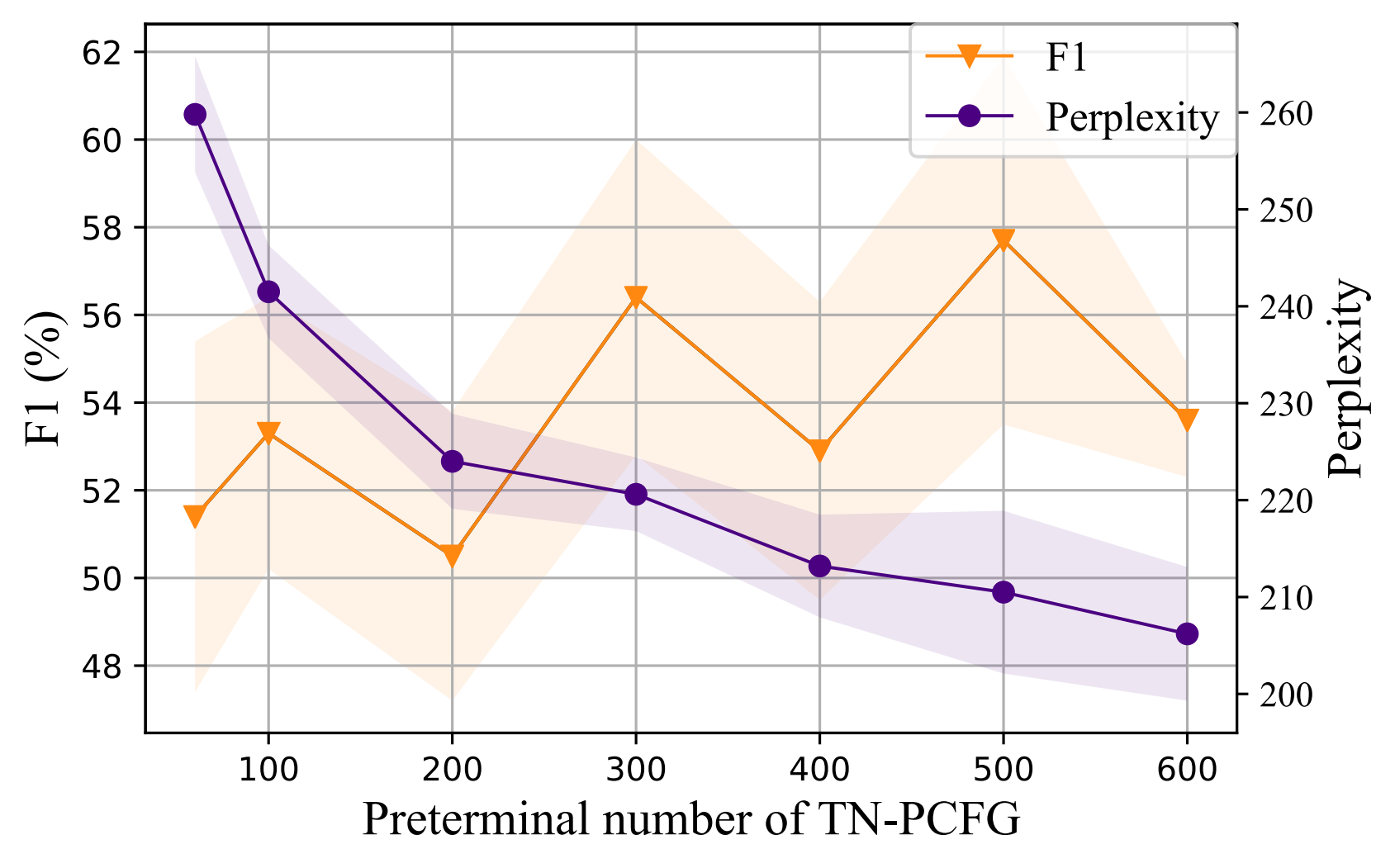
Experiment

Comparison on WSJ test set



TN-X: stands for X preterminals; N: [4]; C: [4]; NL: [5]

Influence of symbol number



We set the ratio n:p=1:2. When increasing the number of symbols, perplexities decrease, F1 scores tend to increase but have a large variance.

Multilingual Experiment

We tune the model in WSJ dev set and test the generalizability beyond English without tuning on each treebanks.

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

Summary

- We have presented TD-PCFGs, a new parameterization form of PCFGs based on tensor decomposition, reducing complexity from cubic to at most quadratic in m.
- We further use neural parameterization to improve unsupervised parsing performance.
- Experiments in 11 languages show the effectiveness in increasing the number of symbols.

Code

Our code is publicly available at:

`https://github.com/sustcsonglin/TN-PCFG`

We also provide faster implementation of neural/compound/lexicalized/bilexicalized PCFGs !

Questions?

References i

R. Buhai, Y. Halpern, Y. Kim, A. Risteski, and D. A. Sontag.
Empirical study of the benefits of overparameterization in learning latent variable models.
In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1211–1219. PMLR, 2020.

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

References ii

L. Jin, F. Doshi-Velez, T. Miller, L. Schwartz, and W. Schuler.
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, July 2019. Association for Computational Linguistics.

Y. Kim, C. Dyer, and A. Rush.
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, July 2019. Association for Computational Linguistics.

References iii


H. Zhu, Y. Bisk, and G. Neubig.
The return of lexical dependencies: Neural lexicalized PCFGs.
Transactions of the Association for Computational Linguistics, 8:647–661, 2020.