SimplePCFG: Simple Hardware-Efficient PCFGs with Independent Left and Right Productions

Wei Liu*, Songlin Yang*, Yoon Kim, Kewei Tu
School of Information Science and Technology, ShanghaiTech University
MIT CSAIL

Advantages Brought by Scaling

- Low-rank parameterization enables a dramatic increase in the numbers of nonterminals(NT) preterminals(PT), from just over **30** and **60** to upwards to **5,000** and **10,000** respectively
- The Sentence-F1 score in unsupervised parsing sees an increase from **55.2** to **64.1**, a significant improvement attributable to scaling

The Achilles' Heel of Low-rank PCFGs

- Despite benefiting from scaling in unsupervised parsing, low-rank PCFGs perform poorly as a **language model** and **underperform similarly-sized HMMs**
- On the Penn Treebanck, PCFGs scaled via low-rank parameterization with thousands of states achieves ≅ **170**
- However, it lags behind a similarly-sized HMM which obtains ≅ **130** perplexity, even though **HMMs are subclass of PCFGs**.

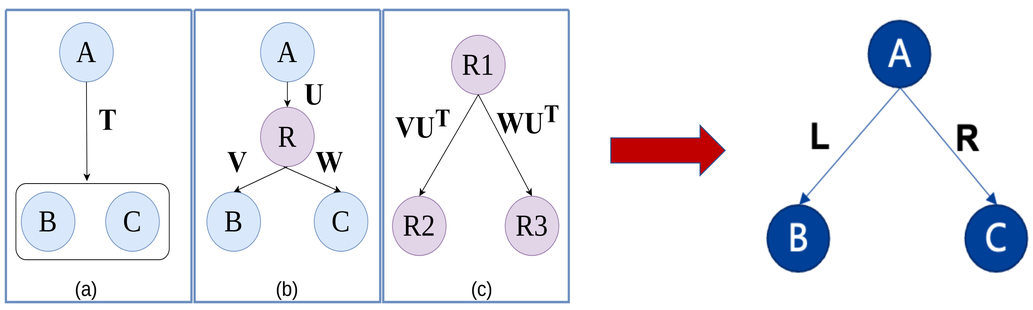
Low-Rank PCFGs

- The previous approach to scaling HMMs and PCFGs to thousands of nontermals is parameterizing the rule probability tensor $\mathbf{T} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{M}|\times|\mathcal{M}|}$ to be low-rank
- Low-rank PCFGs can be viewed as introducing a new latent variable, namely a "rank variable" R, where $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ are tensor/matrix representations of rule probabilities.
- In fact, a low-rank PCFG can be parameterized as a PCFG with independent left/right productions by marginalizing nonterminal variables and viewing the rank variables as new nonterminal variables
- As such, low-rank PCFGs parameterize $\boldsymbol{L}, \boldsymbol{R}$ in a more restrictive manner: $\boldsymbol{L} = \boldsymbol{V}\boldsymbol{U}^T, \boldsymbol{R} = \boldsymbol{W}\boldsymbol{U}^T$. We speculate that the shared $\boldsymbol{U}^T$ would restrict the expressiveness of low-rank PCFGs and thus hinder optimization, which motivates our simple PCFGs.

SimplePCFG

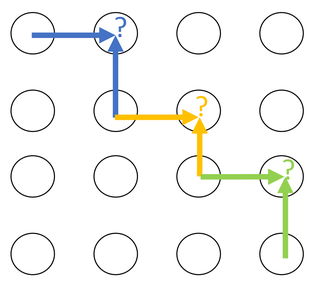
In simple PCFGs, we simplify all these things.

- We decompose $\pi_{A\to BC}$ into $\boldsymbol{\pi_{B\leftarrow A}} \cdot \boldsymbol{\pi_{A\to C}}$, effectively assuming that left and right children are generated independently
- In simplePCFG, we parameterize $\boldsymbol{L}, \boldsymbol{R}$ directly instead of through the shared $\boldsymbol{U}^T$, which in fact contributes to building a **more flexible parameterization**



FlashInside: A Hardware-efficient Inside Algorithm

To facilitate scaling of simple PCFGs, we introduce FlashInside, a hardware-efficient IO-aware implementation of the inside algorithm. It consists of four techniques:
Span-level Parallelism, **The log-einsum-exp trick**, **Kernel Fusion** and **Recomputation**



Span-Level Parallelism

$$\boldsymbol{a}_{ij} = \boldsymbol{x}^{\dagger} + \log\left(\mathbf{L}\exp(\boldsymbol{o}_{ij} - \boldsymbol{x}^{\dagger})\right)$$
$$\boldsymbol{b}_{ij} = \boldsymbol{x}^{\dagger} + \log\left(\mathbf{R}\exp(\boldsymbol{o}_{ij} - \boldsymbol{x}^{\dagger})\right)$$

Log-einsum-Trick for SimplePCFG
$\boldsymbol{a}, \boldsymbol{b}$ is used for computing inside probability and $\boldsymbol{L}, \boldsymbol{R}$ are left and right production rules

Speed & Memory Comparison

Algorithm	$\lvert\mathcal{N}\rvert$	l	Speed	Memory
log-sum-exp	512	20	1x	100x
log-einsum-exp	512	20	4.8x	3x
FlashInside	512	20	9.5x	1x
log-einsum-exp	8192	20	1x	2x
FlashInside	8192	20	6x	1x
log-sum-exp	512	40	1x	50x
log-einsum-exp	512	40	16x	3x
FlashInside	512	40	44x	1x
log-einsum-exp	8192	40	1x	2.4x
FlashInside	8192	40	39x	1x

Language Modeling

- Simple Neural PCFG (SN-PCFG) outperforms previous low-rank PCFG with similar size
- SN-PCFG successfully **exceeds similarly-sized HMMs**

Model	NT	ppl (↓)
NHMM	4096	147
LHMM	16384	131.8
Rank HMM	16384	127.0
Rank HMM	32768	126.4
Rank PCFG†	4096	$174.5_{\pm 11.1}$
Rank PCFG†	8192	$161.2_{\pm 8.9}$
SN-PCFG	4096	$125.4_{\pm 4.1}$
SN-PCFG	8192	$\mathbf{119.0}_{\pm 5.3}$

Unsupervised Parsing

- SC-PCFG is simple compound neural PCFG
- SN-PCFG or SC-PCFG outperforms previous PCFG models on un-supervised parsing benchmarks across different languages
- Simple PCFG vs. Neural PCFG: Despite the better scalability of simple PCFGs, we find that under the same number of NT (i.e. 128), SN-PCFG expectedly under-performs N-PCFG

Model	NT	S-F1 (↑)	ppl (↓)
N-PCFG	30	50.8	252.6
C-PCFG	30	55.2	-
TN-PCFG	500	57.7	210.0
Rank PCFG	4500	64.1	168.0
Rank PCFG†	4096	$60.1_{\pm 7.6}$	$165.1_{\pm 7.7}$
Rank PCFG†	8192	$61.1_{\pm 5.9}$	$171.2_{\pm 11.7}$
N-PCFG†	128	$56.7_{\pm 3.7}$	$181.1_{\pm 15.3}$
SN-PCFG	128	$51.1_{\pm 4.1}$	$231.7_{\pm 8.1}$
SN-PCFG	4096	$\mathbf{65.1}_{\pm 2.1}$	$\mathbf{132.5}_{\pm 4.9}$
SN-PCFG	8192	$62.9_{\pm 2.8}$	$134.6_{\pm 9.1}$
SC-PCFG	512	$54.3_{\pm 4.8}$	-
SC-PCFG	2048	$60.6_{\pm 3.6}$	-
PRPN	-	37.4	-
ON	-	47.7	-
$\text{DIORA}_{\text{+span constraint}}$	-	61.2	-
S-DIORA	-	57.6	-
Constituency test	-	62.8	-
StructFormer	-	54.0	-
Fast-R2D2	-	57.2	-
Right-Branching	-	39.5	-
Oracle Trees	-	84.3	-

Model	NT	Chinese		French		German	
		S-F1(↑)	ppl(↓)	S-F1(↑)	ppl(↓)	S-F1(↑)	ppl(↓)
Left-Branching	-	7.2	-	5.7	-	10.0	
Right-Branching	-	25.5	-	26.4	-	14.07	-
Random Trees	-	15.2	-	16.2	-	13.9	-
kim-2022-revisiting	-	-	-	41.9	-	47.3	-
li-lu-2023-contextual	-	-	-	48.7	-	40.8	-
N-PCFG	30	$26.3_{\pm 2.5}$		$45.0_{\pm 2.0}$		$42.3_{\pm 1.6}$	
C-PCFG	30	$38.7_{\pm 6.6}$	-	$45.0_{\pm 1.1}$	-	$43.5_{\pm 1.2}$	-
TN-PCFG	250	$39.2_{\pm 5.0}$		$39.1_{\pm 4.1}$		$47.1_{\pm 1.7}$	
Rank PCFG	4096	$31.00_{\pm 8.9}$	$409.4_{\pm 29.5}$	$31.2_{\pm 9.3}$	$355.8_{\pm 13.7}$	$35.6_{\pm 9.1}$	$215.3_{\pm 57.1}$
Rank PCFG	8192	$32.4_{\pm 8.2}$	$372.6_{\pm 31.4}$	$32.9_{\pm 10.6}$	$332.2_{\pm 60.8}$	$38.9_{\pm 9.6}$	$190.5_{\pm 65.9}$
SN-PCFG	4096	$39.9_{\pm 6.3}$	$328.3_{\pm 62.1}$	$38.0_{\pm 3.1}$	$379.7_{\pm 5.2}$	$46.7_{\pm 4.9}$	$\mathbf{157.8}_{\pm 65.6}$
SN-PCFG	8192	$41.2_{\pm 3.5}$	$\mathbf{288.2}_{\pm 11.7}$	$43.3_{\pm 9.9}$	$\mathbf{259.9}_{\pm 70.2}$	$46.9_{\pm 5.1}$	$159.5_{\pm 77.2}$
SC-PCFG	512	$38.4_{\pm 7.4}$	-	$47.9_{\pm 1.2}$	-	$47.7_{\pm 1.0}$	-
SC-PCFG	2048	$\mathbf{42.9}_{\pm 2.9}$	-	$\mathbf{49.9}_{\pm 1.7}$	-	$\mathbf{49.1}_{\pm 1.0}$	-