Unsupervised Recurrent Neural Network Grammar

Wang Ge

Group Meeting

2019/5/29

Kim et al. 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics

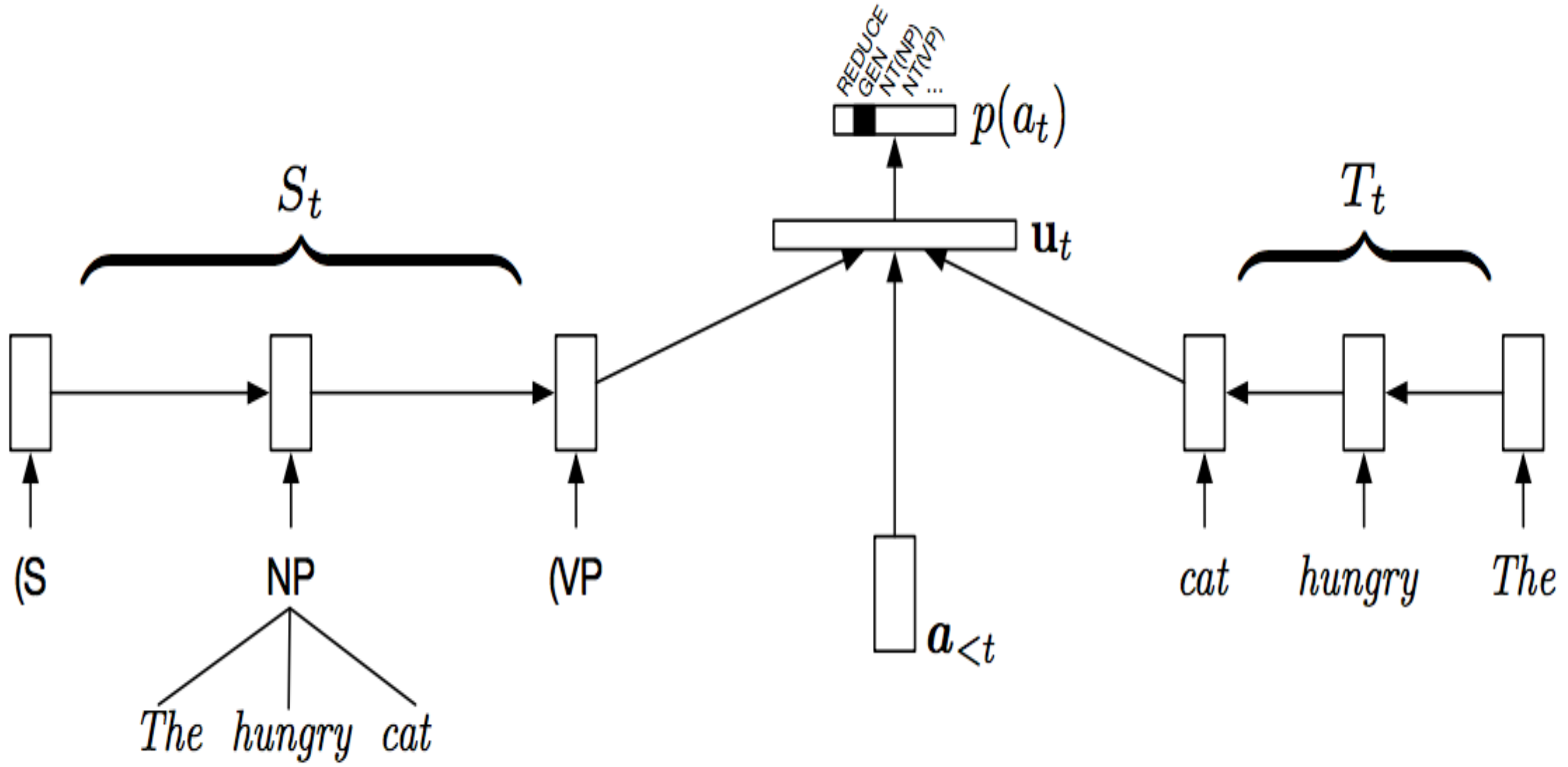
What is Recurrent Neural Network Grammar

- A transition-based generative model proposed by Dyer et al in 2016

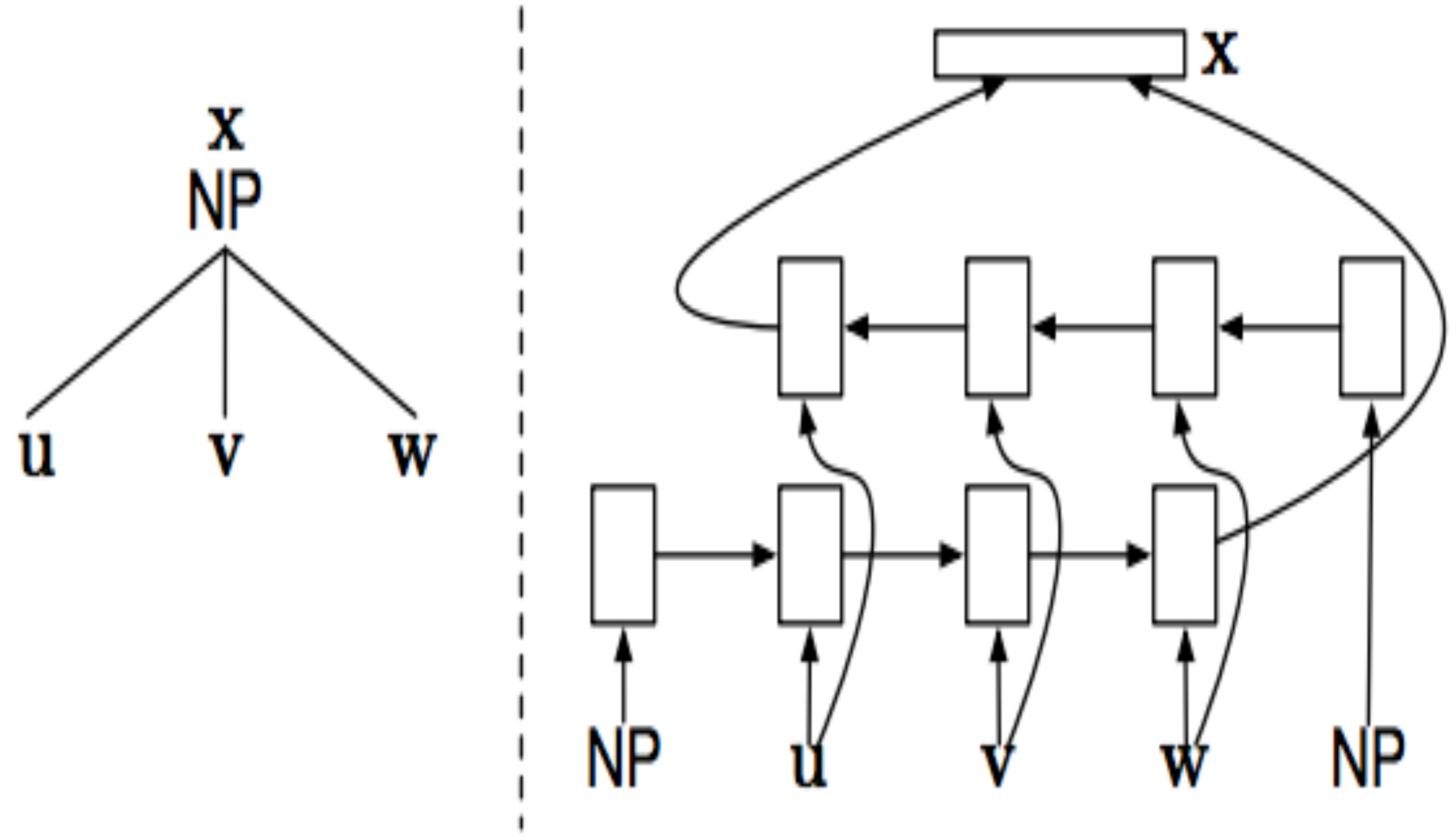
	Stack	Terminals	Action
0			NT(S)
1	(S		NT(NP)
2	(S | (NP		GEN(*The*)
3	(S | (NP | *The*	*The*	GEN(*hungry*)
4	(S | (NP | *The* | *hungry*	*The* | *hungry*	GEN(*cat*)
5	(S | (NP | *The* | *hungry* | *cat*	*The* | *hungry* | *cat*	REDUCE
6	(S | (NP *The hungry cat*)	*The* | *hungry* | *cat*	NT(VP)
7	(S | (NP *The hungry cat*) | (VP	*The* | *hungry* | *cat*	GEN(*meows*)
8	(S | (NP *The hungry cat*) | (VP *meows*	*The* | *hungry* | *cat* | *meows*	REDUCE
9	(S | (NP *The hungry cat*) | (VP *meows*)	*The* | *hungry* | *cat* | *meows*	GEN(.)
10	(S | (NP *The hungry cat*) | (VP *meows*) | .	*The* | *hungry* | *cat* | *meows* | .	REDUCE
11	(S (NP *The hungry cat*) (VP *meows*) .)	*The* | *hungry* | *cat* | *meows* | .	

- Two basic actions: generation(shift),reduce
- Actions determined by stacks, buffered generations and previous actions
- Composition of stack LSTMs and bi-directional LSTMs

Network Structure of RNNG



The overall network structure: concatenate stack, buffer and action representations, then use MLPs to decide next action



The part for reduce: form the representation of the reduced span with bi-directional LSTM

What Does RNNG Learn About Syntax

Further studies on RNNG find following facts:

- The composition function, i.e. the reduce part is the key to RNNG’s good performance in parsing
- The information in buffers and actions and redundant, determining actions on the stack only is enough
- Non-terminal Labels add only a little information about phrase behavior, a good grammar can be learned without them

Is it possible to learn an RNNG without golden tree annotation?

Learn RNNG with an Autoencoder

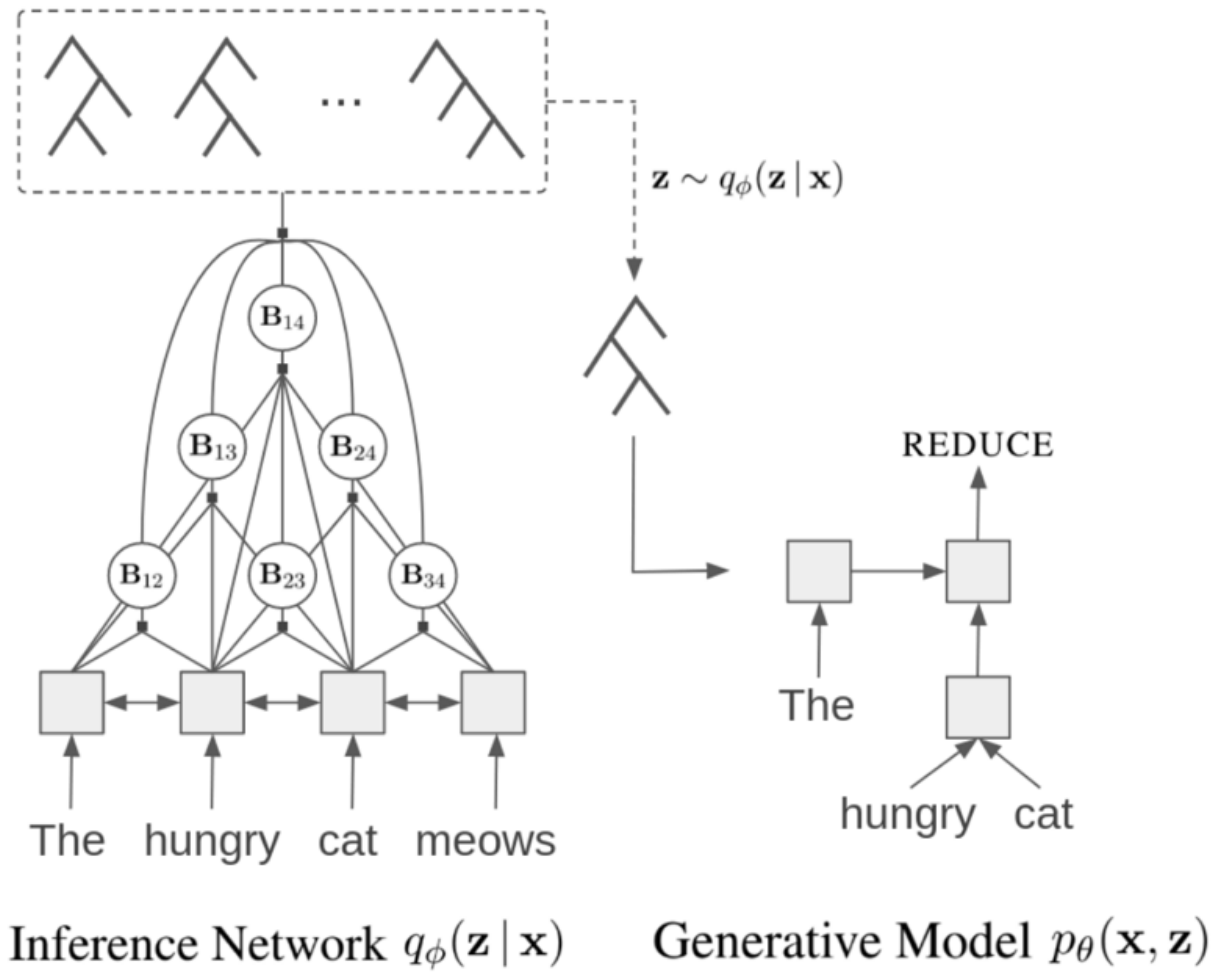
- It is natural to learn a grammar like RNNG unsupervisedly with an autoencoder

- Use an inference network $q(\mathbf{z}|\mathbf{x})$ to sample parse trees $\mathbf{z}$ according to input sentence

- Learn a generative model $p(\mathbf{z}, \mathbf{x})$ that maximize the probability to reconstruct input sentence

- Ensure that backpropagation is possible for every step

Approach Overview



- The inference network predicts a binary tree, indicated by boolean variables that decide if two spans form a subtree
- The generative model is similar to that of the original RNNG, except that the reduce action is realized by tree LSTMs and no non-terminal labels are involved

Encoder Details

- Use bi-directional LSTMs to compute the span score

$$s_{ij} = \mathrm{MLP}([\overrightarrow{\mathbf{h}}_{j+1} - \overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_{i-1} - \overleftarrow{\mathbf{h}}_j])$$

- Formulate the inference network as a CRF parser

$$q_\phi(\mathbf{B} \,|\, \mathbf{x}) = \frac{1}{Z_T(\mathbf{x})} \exp\Big(\sum_{i \le j} \mathbf{B}_{ij} s_{ij}\Big), \quad Z_T(\mathbf{x}) = \sum_{\mathbf{B}' \in \mathcal{B}_T} \exp\Big(\sum_{i \le j} \mathbf{B}'_{ij} s_{ij}\Big),$$

- Finally map the boolean variables to RNNG actions

Define that $\quad f : \mathcal{B}_T \to \mathcal{Z}_T \quad\quad q_\phi(\mathbf{z} \,|\, \mathbf{x}) \triangleq q_\phi(f^{-1}(\mathbf{z}) \,|\, \mathbf{x}).$

Optimization Details

- The optimization objective function is the ELBO:

$$\mathbb{E}_{q_\phi(\mathbf{z}\,|\,\mathbf{x})}[\log p_\theta(\mathbf{x},\mathbf{z})] + \mathbb{H}[q_\phi(\mathbf{z}\,|\,\mathbf{x})],$$

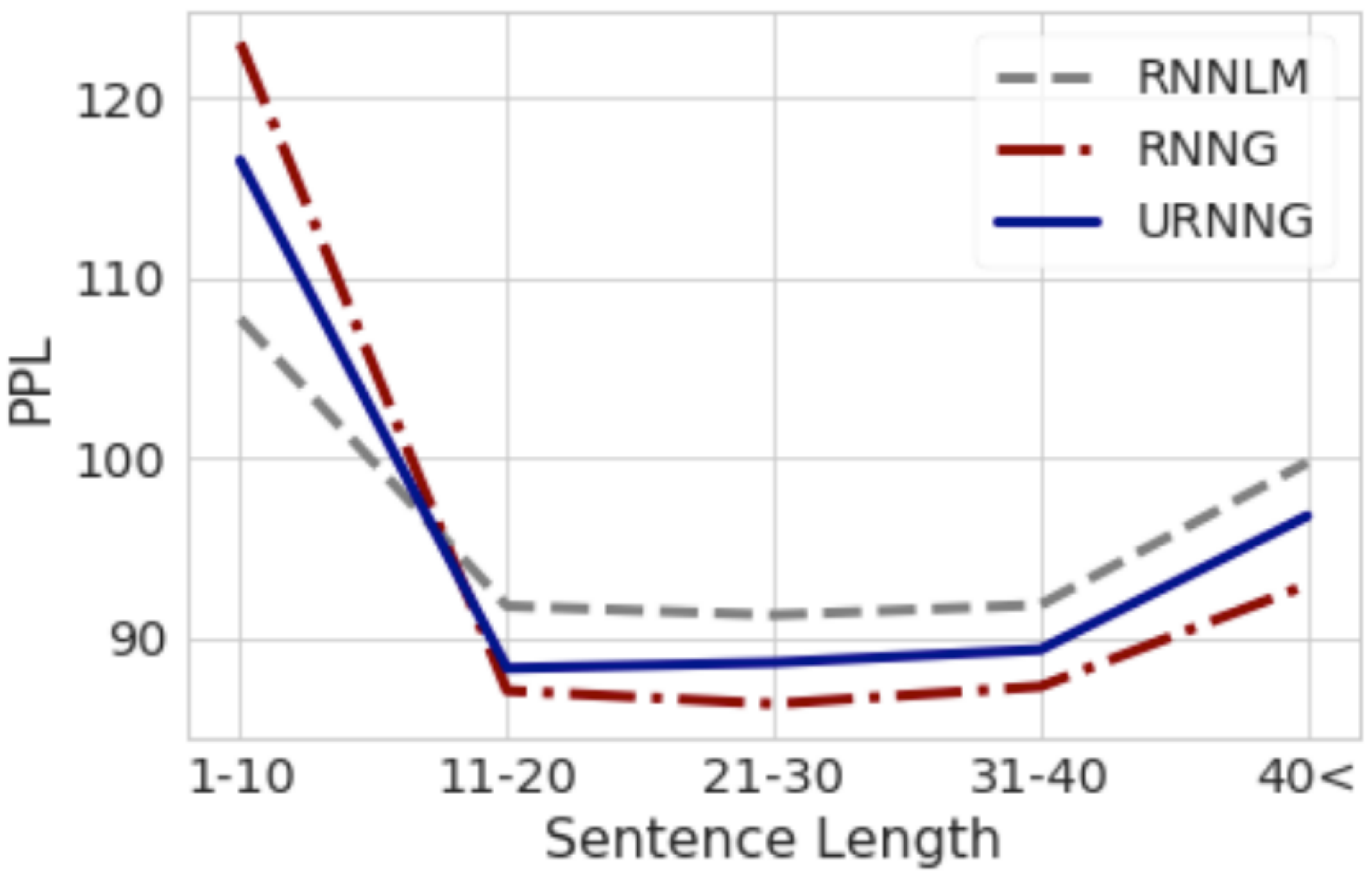
- The optimization of decoder and encoder parameter:

$$\nabla_\theta \,\mathrm{ELBO}(\theta,\phi;\mathbf{x}) \approx \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z}^{(k)}), \qquad \frac{1}{K}\sum_{k=1}^{K}(\log p_\theta(\mathbf{x},\mathbf{z}^{(k)}) - r^{(k)})\nabla_\phi \log q_\phi(\mathbf{z}^{(k)}\,|\,\mathbf{x}),$$

where $r^{(k)} = \frac{1}{K-1}\sum_{j\neq k}\log p_\theta(\mathbf{x},\mathbf{z}^{(j)})$ is used to control variance

Experimental Results

Model	PTB PPL	PTB F_1	CTB PPL	CTB F_1
RNNLM	93.2	–	201.3	–
PRPN (default)	126.2	32.9	290.9	32.9
PRPN (tuned)	96.7	41.2	216.0	36.1
Left Branching Trees	100.9	10.3	223.6	12.4
Right Branching Trees	93.3	34.8	203.5	20.6
Random Trees	113.2	17.0	209.1	17.4
URNNG	90.6	40.7	195.7	29.1
RNNG	88.7	68.1	193.1	52.3
RNNG $\rightarrow$ URNNG	85.9	67.7	181.1	51.9
Oracle Binary Trees	–	82.5	–	88.6



Experimental Results

Tree	PTB	CTB
Gold	40.7	29.1
Left	9.2	8.4
Right	68.3	51.2
Self	92.3	87.3
RNNG	55.4	47.1
PRPN	41.0	47.2

Label	URNNG	PRPN
SBAR	74.8%	28.9%
NP	39.5%	63.9%
VP	76.6%	27.3%
PP	55.8%	55.1%
ADJP	33.9%	42.5%
ADVP	50.4%	45.1%

	RNNLM	PRPN	RNNG	URNNG
PPL	93.2	96.7	88.7	90.6
Overall	62.5%	61.9%	69.3%	64.6%
Subj.	63.5%	63.7%	89.4%	67.2%
Obj. Rel.	62.6%	61.0%	67.6%	65.7%
Refl.	60.7%	68.8%	57.3%	60.5%
NPI	58.7%	39.5%	46.8%	55.0%

Experimental Results from further analysis on trees learned by URNNG