Sequence-to-Sequence Learning with Latent Neural Grammars

Yoon Kim
MIT

Background: Seq2seq with Neural Networks

- **Goal**: model the distribution over output sequence **y** given input sequence **x**

$$p_\theta(y \mid x) = \prod_{t=1}^{T} p_\theta(y_t \mid x, y_{<t})$$

- Sequence-to-sequence Learning with Neural Networks [Cho et al. ’14, Sutskever et al. ’14]: autoregressive factorization.

Background: Seq2seq with Neural Networks

- Any distribution over the output can be factorized left-to-right via the chain rule ⇒ given large enough data and model, this should work well.

- But this flexibility comes at a cost:
  - weak inductive biases for capturing hierarchical structure ⇒ over-reliance on surface-form correlations
  - sample inefficiency
  - opaque generation process

This Work: Seq2seq with Latent Neural Grammars

This Work: Seq2seq with Latent Neural **Grammars**

- Model $p_\theta(y \mid x)$ with a (quasi) synchronous **grammar** (vs. a "flat" autoregressive model)

This Work: Seq2seq with Latent **Neural** Grammars

- Model $p_\theta(y \mid x)$ with a (quasi) synchronous grammar (vs. a "flat" autoregressive model)
- Use **neural features** for efficient parameterizations over combinatorial input space of derivation rules.

This Work: Seq2seq with **Latent** Neural Grammars

- Model $p_\theta(y \mid x)$ with a (quasi) synchronous grammar (vs. a "flat" autoregressive model)
- Use neural features for efficient parameterizations over combinatorial input space of derivation rules.
- Both source and target trees are as fully **latent** and induced during training.

Quasi-Synchronous Context-Free Grammars

- QCFG [Smith and Eisner '06]: A monolingual grammar over the target side conditioned on a source tree, where the target-side rules dynamically depend on source tree nodes.

- Hierarchical generative process where each node in the target is tranduced by a node in the source tree ⇒ provides provenance for how each output part is generated!

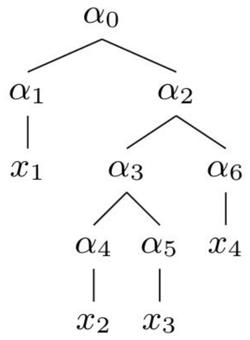
- Unlike classic synchronous context-free grammars, does not require source and target trees to be isomorphic.

QCFG

$$\boxed{G[\boldsymbol{s}]} = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}[\boldsymbol{s}], \theta)$$

Grammar defines a CFG over target side given source tree ***s***

$\boldsymbol{s} =$



QCFG

$$G[\boldsymbol{s}] = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}[\mathbf{s}], \theta)$$

Start symbol
Nonterminals / Preterminals
Target terminals

QCFG

Model parameters

$$G[\boldsymbol{s}] = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}[\boldsymbol{s}], \theta)$$

Context-free rules where each target derivation is aligned to a source tree node

QCFG

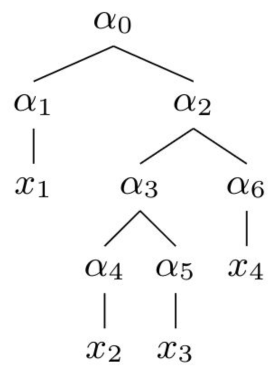
$$G[\boldsymbol{s}] = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}[\boldsymbol{s}], \theta)$$

Start rule	$S \rightarrow A[\alpha_i],$	$A \in \mathcal{N},$
Binary rules	$A[\alpha_i] \rightarrow B[\alpha_j]C[\alpha_k],$	$A \in \mathcal{N}, \quad B, C \in \mathcal{N} \cup \mathcal{P},$
Unary rules	$D[\alpha_i] \rightarrow w,$	$D \in \mathcal{P}, w \in \Sigma$

QCFG

$$G[\boldsymbol{s}] = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}[\boldsymbol{s}], \theta)$$

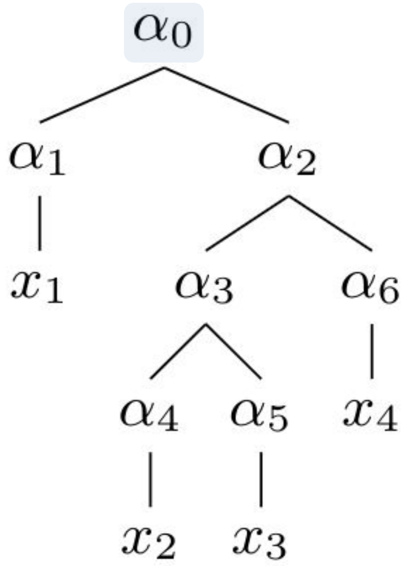
Start rule	$S \rightarrow A[\alpha_i],$	$A \in \mathcal{N},$
Binary rules	$A[\alpha_i] \rightarrow B[\alpha_j]C[\alpha_k],$	$A \in \mathcal{N}, \quad B, C \in \mathcal{N} \cup \mathcal{P},$
Unary rules	$D[\alpha_i] \rightarrow w,$	$D \in \mathcal{P}, w \in \Sigma$



$$\alpha_i, \alpha_j, \alpha_k \in \boldsymbol{s}$$

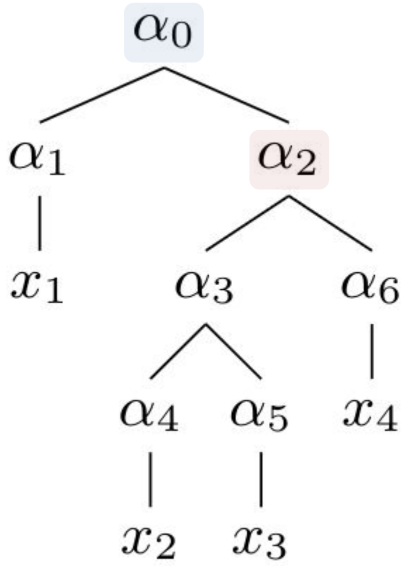
Each nonterminal is decorated with a node in the source tree.

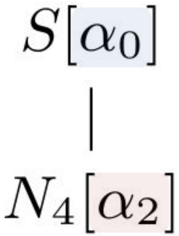
QCFG Example



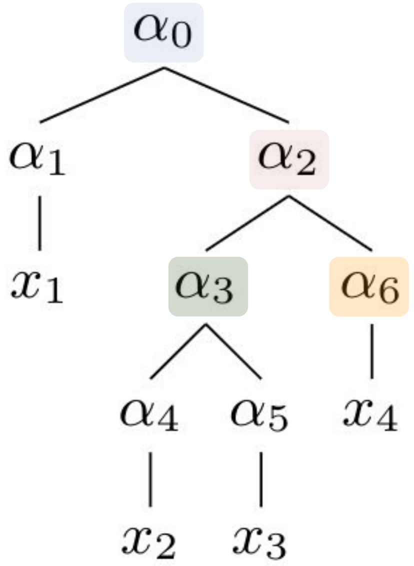
$$S[\alpha_0]$$

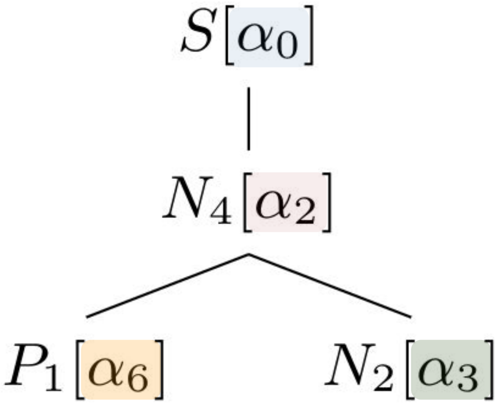
QCFG Example



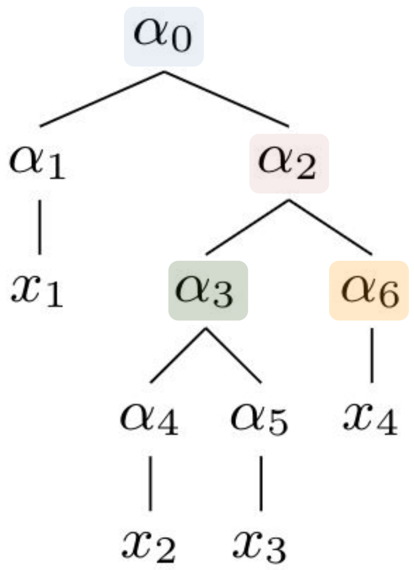


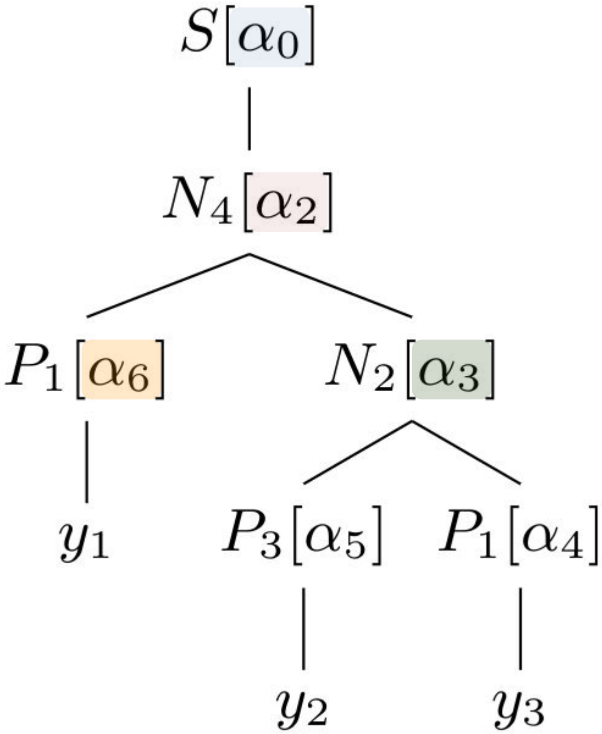
QCFG Example



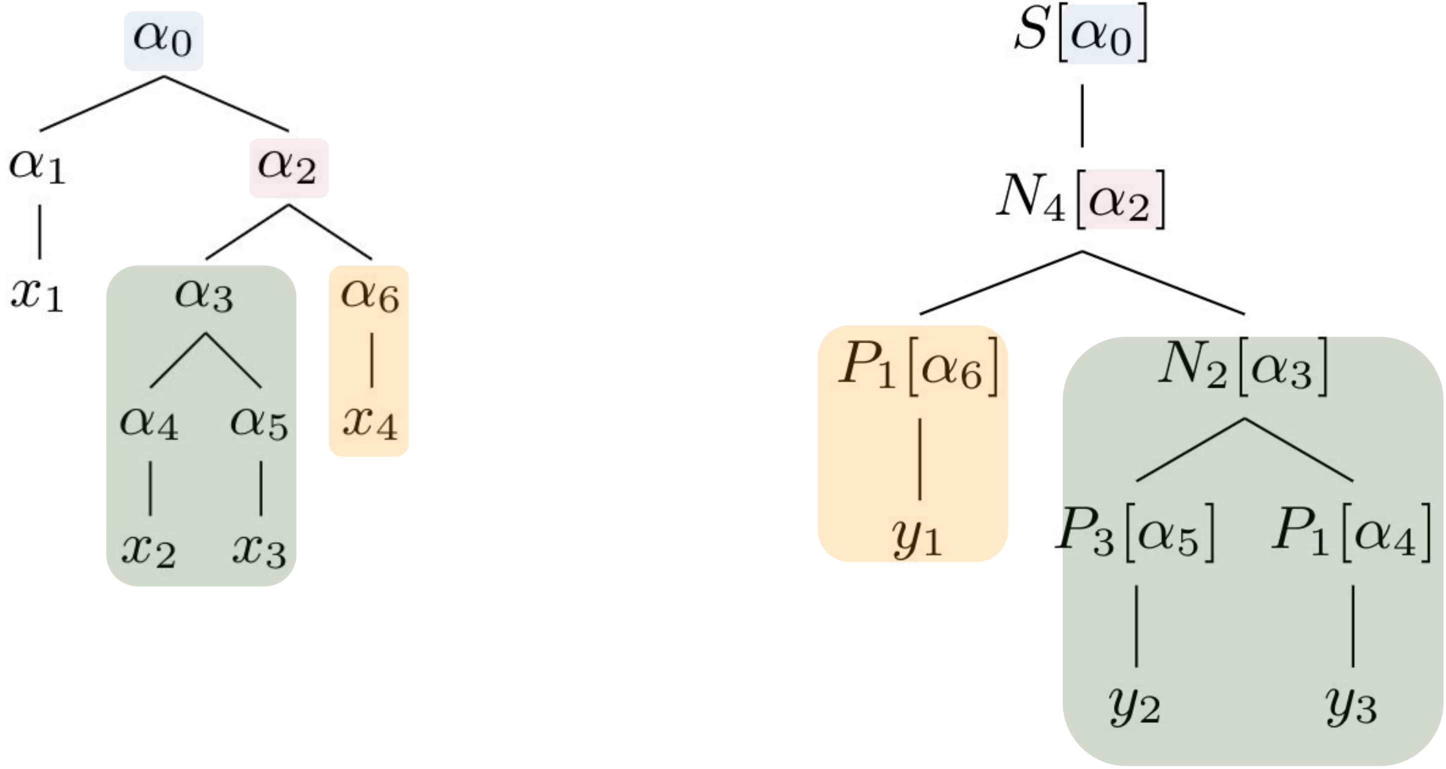


QCFG Example

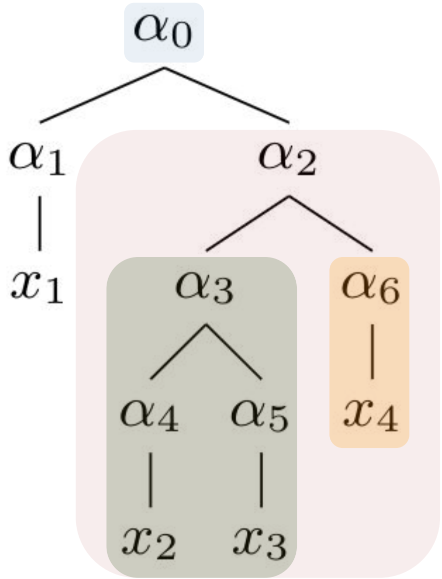


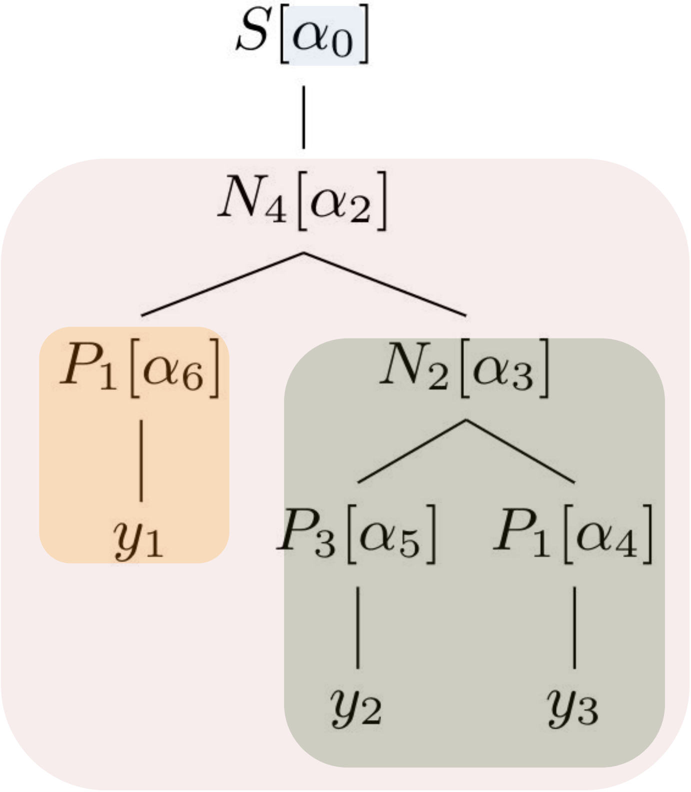


QCFG Example



QCFG Example





Parameterization

$$G[\boldsymbol{s}] = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}[\boldsymbol{s}], \theta)$$

- Prior work: handcrafted features over source tree nodes.
- This work: neural parameterization of derivation rules.

$$\mathbf{e}_{A[\alpha_i]} = \mathbf{u}_A + \mathbf{h}_{\alpha_i}$$

Parameterization

$$G[\boldsymbol{s}] = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}[\boldsymbol{s}], \theta)$$

- Prior work: handcrafted features over source tree nodes.

- This work: neural parameterization of derivation rules.

$$\mathbf{e}_{A[\alpha_i]} = \mathbf{u}_A + \mathbf{h}_{\alpha_i}$$

Nonterminal symbol embedding

Source tree node embedding (from TreeLSTM)

Parameterization

$$G[\boldsymbol{s}] = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}[\boldsymbol{s}], \theta)$$

- Prior work: handcrafted features over source tree nodes.
- This work: neural parameterization of derivation rules.

$$\mathbf{e}_{A[\alpha_i]} = \mathbf{u}_A + \mathbf{h}_{\alpha_i}$$

- Rule probabilities given by neural network over embeddings.

$$p_\theta(A[\alpha_i] \to B[\alpha_j]C[\alpha_k])$$
$$\propto \exp\left(f_1(\mathbf{e}_{A[\alpha_i]})^\top (f_2(\mathbf{e}_{B[\alpha_j]}) + f_3(\mathbf{e}_{C[\alpha_k]}))\right)$$

Learning

- QCFG defines a distribution over target trees (and strings) given a source tree.

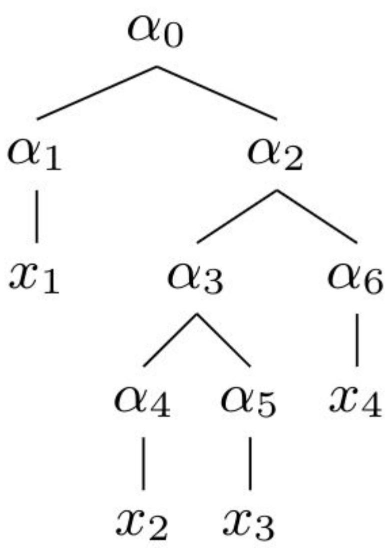
$$\sum_{\boldsymbol{t} \in \mathcal{T}(\boldsymbol{y})} p_\theta(\boldsymbol{t} \mid \boldsymbol{s}) \;=\; p_\theta(\boldsymbol{y} \mid \boldsymbol{s})$$

($\mathcal{O}(|\mathcal{N}|(|\mathcal{N}| + |\mathcal{P}|)^2 S^3 T^3)$with the usual inside algorithm)

- Past work: source tree given by a pipelined parser.

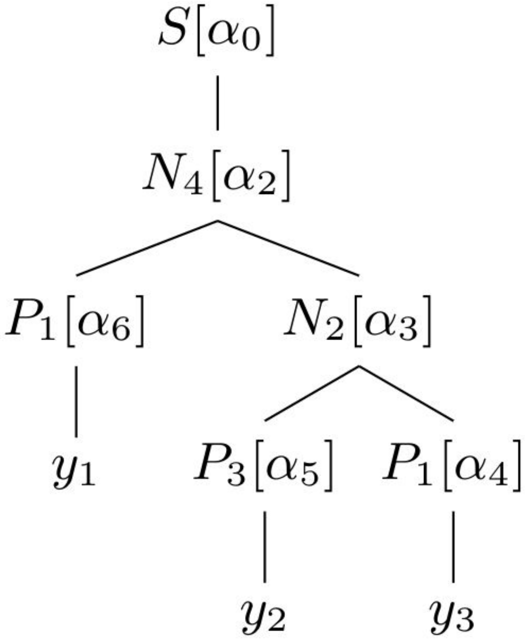
- This work: learn source parser $p_\phi(\boldsymbol{s} \mid \boldsymbol{x})$ alongside the QCFG. Source parser is a neural PCFG from [Kim et al. ’19].

Learning



$$p_\phi(\boldsymbol{s} \mid \boldsymbol{x})$$

Source PCFG



$$p_\theta(\boldsymbol{t} \mid \boldsymbol{s})$$

Target QCFG

Learning

- Log marginal likelihood given by:

$$\log p_{\theta,\phi}(\boldsymbol{y} \mid \boldsymbol{x}) = \log \left(\sum_{\boldsymbol{s} \in \mathcal{T}(\boldsymbol{x})} \sum_{\boldsymbol{t} \in \mathcal{T}(\boldsymbol{y})} p_\theta(\boldsymbol{t} \mid \boldsymbol{s}) p_\phi(\boldsymbol{s} \mid \boldsymbol{x}) \right)$$

Target QCFG Source PCFG

- Exact marginalization intractable ⇒ optimize a lower bound:

$$\log p_{\theta,\phi}(\boldsymbol{y} \mid \boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{s} \sim p_\phi(\boldsymbol{s} \mid \boldsymbol{x})} \left[\log p_\theta(\boldsymbol{y} \mid \boldsymbol{s}) \right]$$

(Score function estimator with self-critical control variate)

Inference

- Given the source MAP tree from the PCFG,

$$\widehat{\boldsymbol{s}} = \operatorname{argmax}_{\boldsymbol{s}} p_\phi(\boldsymbol{s} \mid \boldsymbol{x})$$

finding the MAP QCFG tree is intractable

$$\operatorname{argmax}_{\boldsymbol{y}} p_\theta(\boldsymbol{y} \mid \widehat{\boldsymbol{s}})$$

- Approximate decoding strategy:
  - Sample target trees $\boldsymbol{t}^{(1)}, \ldots \boldsymbol{t}^{(K)}$ from $G[\widehat{\boldsymbol{s}}]$
  - Rescore and return the yield with lowest perplexity.

Experimental Setup

- Experiments on three seq2seq tasks:
  - SCAN [Lake and Baroni ’18]: synthetic language navigation task to test for compositional generalization.
  - StylePTB [Lyu et al. ’21]: style transfer on the Penn Treebank.
  - Small-scale machine translation [Lake and Baroni ’18]

Results: SCAN

- SCAN: simple language navigation task

“jump twice after walk” ⇒ *WALK JUMP JUMP*

- Seq2seq models have a hard time generalizing compositionally

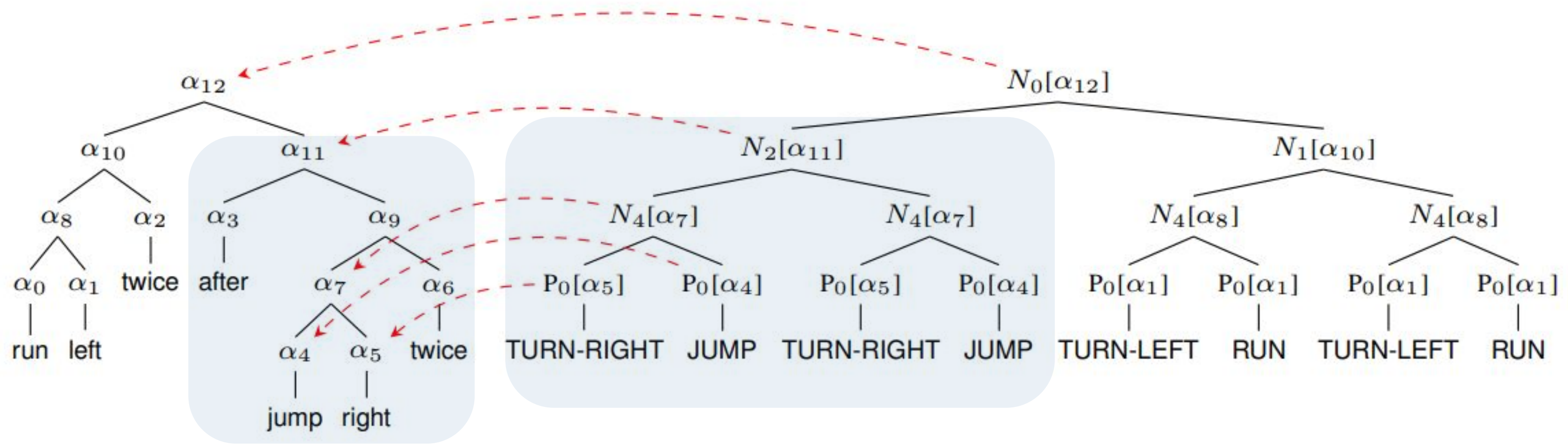
Model	**Simple Split**	**Add Primitive**	**Add Template**	**Length**
RNN	≈ 100%	1.7%	2.5%	13.8%
CNN	≈ 100%	69.2%	56.7%	0.0%
Transformer	≈ 100%	1.0%	53.3%	0.0%

Results: SCAN

Model	**Add Primitive**	**Add Template**	**Length**
RNN	1.7%	2.5%	13.8%
CNN	69.2%	56.7%	0.0%
Transformer	1.0%	53.3%	0.0%
Neural QCFG	**96.8%**	**98.7%**	**95.7%**

(Many other methods that also solve this dataset)

Results: SCAN



Results: SCAN

Frequently-occurring rules obtained MAP QCFG trees from the training set.

P_0[run] $\rightarrow$ RUN
P_0[look] $\rightarrow$ LOOK
P_0[walk] $\rightarrow$ WALK
P_0[jump] $\rightarrow$ JUMP
P_0[right] $\rightarrow$ TURN-RIGHT
P_0[left] $\rightarrow$ TURN-LEFT
N_4[look left] $\rightarrow$ P_0[left] P_0[look]
N_4[look right] $\rightarrow$ P_0[right] P_0[look]
N_4[walk left] $\rightarrow$ P_0[left] P_0[walk]
N_4[walk right] $\rightarrow$ P_0[right] P_0[walk]
N_1[look right twice] $\rightarrow$ N_4[look right] N_4[look right]
N_1[walk left twice] $\rightarrow$ N_4[walk left] N_4[walk left]
N_1[look thrice] $\rightarrow$ N_8[look thrice] P_0[look]
N_1[look right thrice] $\rightarrow$ N_8[look right thrice] N_4[look right]
N_8[look right thrice] $\rightarrow$ N_4[look right] N_4[look right]
N_1[walk left thrice] $\rightarrow$ N_8[walk left thrice] N_4[walk left]
N_8[walk left thrice] $\rightarrow$ N_4[walk left] N_4[walk left]

Results: StylePTB

- Experiments on three “hard” style transfer tasks identified by [Lyu et al. ’21]: *active-to-passive*, *adjective emphasis*, *verb emphasis*. (500-3000 examples)

- Incorporate a phrase-level copy mechanism via a special-purpose nonterminal:

$$p_\theta(A_{\text{COPY}}[\alpha_i] \to v) \stackrel{\text{def}}{=} \mathbb{1}\{v = \text{yield}(\alpha_i)\}$$

$$v \in \Sigma^+$$

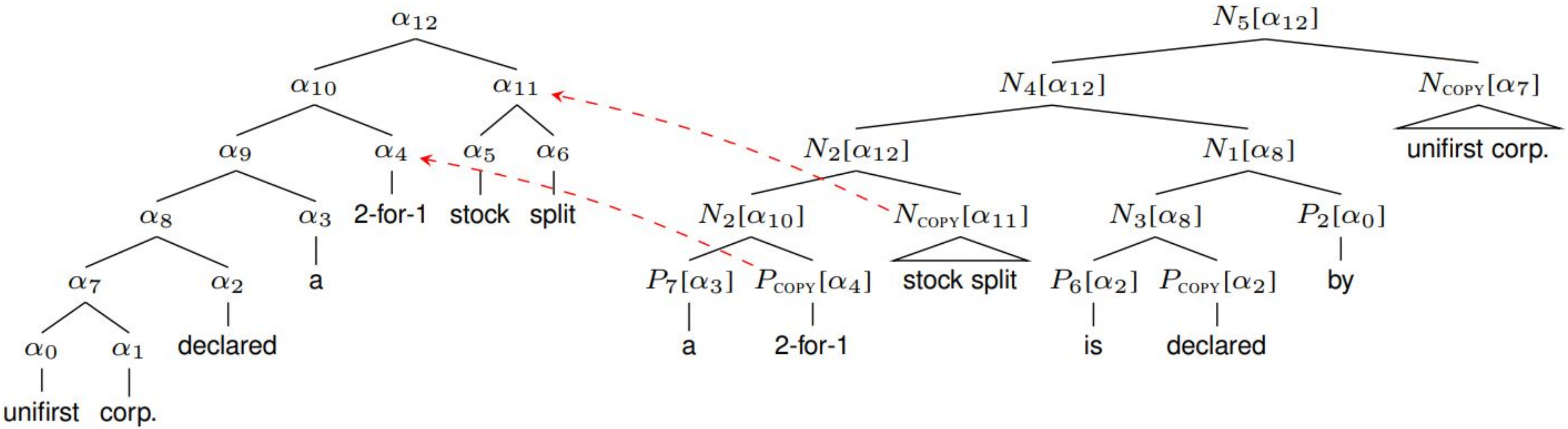
Results: StylePTB

Transfer Type	Approach	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Active to Passive	GPT2-finetune	0.476	0.329	0.238	0.189
	Seq2Seq	0.373	0.220	0.141	0.103
	Retrieve-Edit	0.681	0.598	0.503	0.427
	Human	0.931	0.881	0.835	0.795
	Seq2Seq	0.505	0.349	0.253	0.190
	Neural QCFG	0.431	0.637	0.548	0.472
	Seq2Seq + copy	0.838	0.735	0.673	0.598
	Neural QCFG + copy	0.836	0.771	0.713	0.662

[Lyu et al. '21]

[This work]

Results: StylePTB



(Linguistically incorrect tree)

Results: Machine Translation

- Small-scale English-French machine translation (6000 sentences).

- Compositional generalization [Lake and Baroni ’18]:

    - Add 1000 instances of “*i am daxy ⇒ je suis daxiste*”
    - Must generalize to unseen combinations, e.g.

        he is daxy

        i am not daxy

        i am very daxy

        ...

Results: Machine Translation

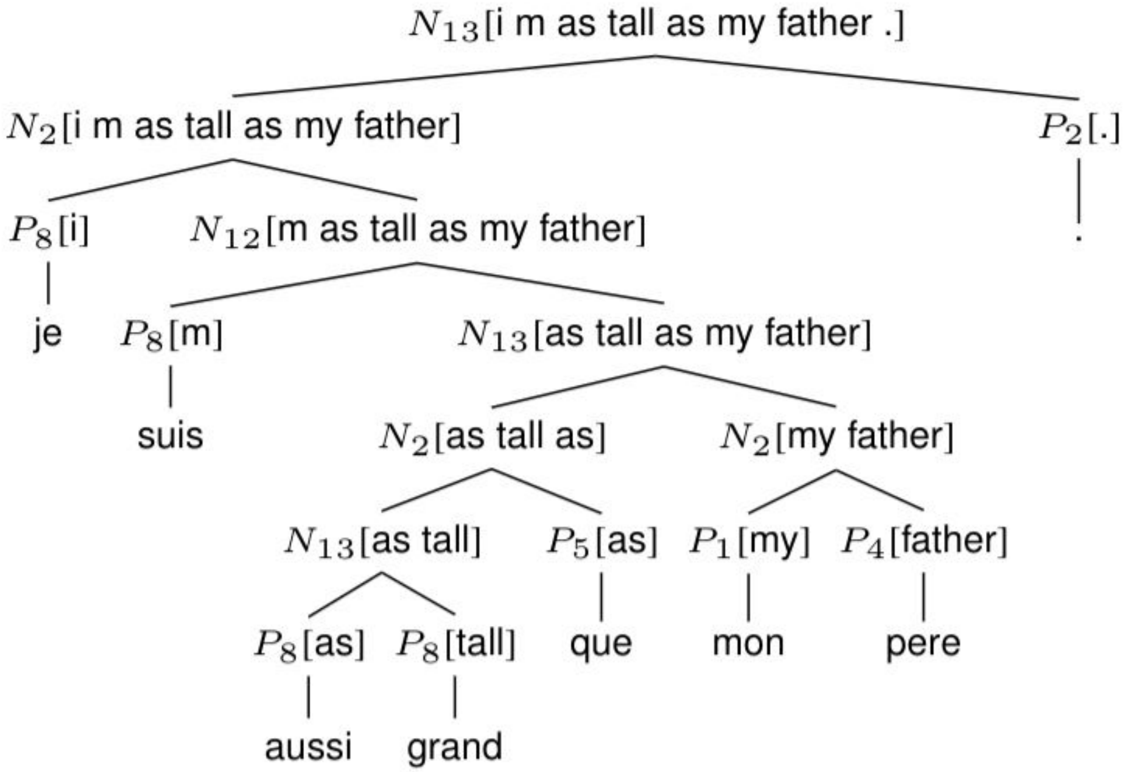
Model	**BLEU on regular test set**	**Accuracy on *daxy* examples**
LSTM	25.1	12.5%
Neural QCFG	23.5	100%
+ BiLSTM Encoder	26.8	75.0%

Results: Machine Translation

Model	**BLEU on regular test set**	**Accuracy on *daxy* examples**
LSTM	25.1	12.5%
Neural QCFG	23.5	100%
+ BiLSTM Encoder	26.8	75.0%
Transformer	30.4	100%

(Mostly negative results on MT)

Results: Machine Translation



Limitations

- Much (much) more expensive than regular seq2seq due to the $\mathcal{O}(|\mathcal{N}|(|\mathcal{N}|+|\mathcal{P}|)^2 S^3 T^3)$ dynamic program.
- Model is brittle: very sensitive to hyperparameters / random initialization.
- Thoroughly outperformed by a well-tuned Transformer on more real-world seq2seq tasks.

Discussion & Conclusion

- What is the role of grammars / neuro-symbolic approaches in the era of large pretrained language models?

- Future work

  - Condition on (embedding representations of) images / video / audio for grounded grammar induction.
  - Richer grammatical formalisms (e.g. synchronous tree-adjoning grammars).
  - Combining induced structures with flexible neural models.