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 $\Rightarrow$ given large enough data and model, this should work well.

- But this flexibility comes at a cost:
  - weak inductive biases for capturing hierarchical structure $\Rightarrow$ 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 transduced by a node in the source tree \( \Rightarrow \) 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.
Grammar defines a CFG over target side given source tree $s$
QCFG

\[ G[s] = (S, N, P, \Sigma, R[s], \theta) \]

Start symbol
Nonterminals / Preterminals
Target terminals
QCFG

\[ G[s] = (S, N, P, \Sigma, R[s], \theta) \]

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

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

Start rule
\[ S \rightarrow A[\alpha_i], \quad A \in \mathcal{N}, \]

Binary rules
\[ A[\alpha_i] \rightarrow B[\alpha_j]C[\alpha_k], \quad A \in \mathcal{N}, \quad B, C \in \mathcal{N} \cup \mathcal{P}, \]

Unary rules
\[ D[\alpha_i] \rightarrow w, \quad D \in \mathcal{P}, w \in \Sigma \]
QCFG

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

Start rule

\[ S \rightarrow A[\alpha_i], \quad A \in \mathcal{N}, \]

Binary rules

\[ A[\alpha_i] \rightarrow B[\alpha_j]C[\alpha_k], \quad A \in \mathcal{N}, \quad B, C \in \mathcal{N} \cup \mathcal{P}, \]

Unary rules

\[ D[\alpha_i] \rightarrow w, \quad D \in \mathcal{P}, w \in \Sigma \]

\[ \alpha_i, \alpha_j, \alpha_k \in s \]

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

\[
S[\alpha_0]
\]

\[
\begin{align*}
&\alpha_0 \\
&\quad \alpha_1 \\
&\quad \quad x_1 \\
&\quad \alpha_2 \\
&\quad \quad \alpha_3 \\
&\quad \quad \quad \alpha_4 \\
&\quad \quad \quad \quad x_2 \\
&\quad \quad \alpha_5 \\
&\quad \quad \quad x_3 \\
&\quad \alpha_6 \\
&\quad \quad x_4
\end{align*}
\]
QCFG Example

\[ S[\alpha_0] \]

\[ N_4[\alpha_2] \]
QCFG Example

\[
\begin{array}{c}
\alpha_0 \\
\alpha_1 & \alpha_2 \\
\alpha_3 & \alpha_6 \\
\alpha_4 & \alpha_5 & x_4 \\
x_2 & x_3 & x_1
\end{array}
\]

\[
\begin{array}{c}
S[\alpha_0] \\
N_4[\alpha_2] \\
P_1[\alpha_6] & N_2[\alpha_3]
\end{array}
\]
QCFG Example

\[
\begin{align*}
S[\alpha_0] \\
\quad &\quad N_4[\alpha_2] \\
\quad &\quad \quad P_1[\alpha_6] \\
\quad &\quad \quad \quad y_1 \\
\quad &\quad \quad \quad P_3[\alpha_5] \\
\quad &\quad \quad \quad \quad y_2 \\
\quad &\quad \quad \quad P_1[\alpha_4] \\
\quad &\quad \quad \quad \quad \quad y_3 \\
\end{align*}
\]
QCFG Example

\[
S[\alpha_0] \\
\downarrow \\
N_4[\alpha_2] \\
\downarrow \\
\begin{array}{c} P_1[\alpha_6] \\
\downarrow \\
y_1 \\
\end{array}
\quad
\begin{array}{c} N_2[\alpha_3] \\
\downarrow \\
P_3[\alpha_5] & P_1[\alpha_4] \\
\downarrow & \downarrow \\
y_2 & y_3 \\
\end{array}
\]

\[
\alpha_0 \\
\alpha_1 \\
\alpha_2 \\
\alpha_3 \\
\alpha_4 \\
\alpha_5 \\
\alpha_6 \\
x_1 \\
x_2 \\
x_3 \\
x_4 \\
y_1 \\
y_2 \\
y_3
\]
QCFG Example
Parameterization

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

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

\[ e_A[\alpha_i] = u_A + h_{\alpha_i} \]
Parameterization

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

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

\[ e_A[\alpha_i] = u_A + h_{\alpha_i} \]

Nonterminal symbol embedding \hspace{1cm} Source tree node embedding (from TreeLSTM)
Parameterization

\[ G[s] = (S, N, P, \Sigma, R[s], \theta) \]

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

\[ e_{A[\alpha_i]} = u_A + h_{\alpha_i} \]

- Rule probabilities given by neural network over embeddings.

\[ p_\theta(A[\alpha_i] \rightarrow B[\alpha_j]C[\alpha_k]) \propto \exp \left( f_1(e_{A[\alpha_i]})^T (f_2(e_{B[\alpha_j]}) + f_3(e_{C[\alpha_k]})) \right) \]
Learning

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

\[ \sum_{t \in T(y)} p_\theta(t \mid s) = p_\theta(y \mid s) \]

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

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

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

Source PCFG

$\mathbf{p}_\phi(\mathbf{s} \mid \mathbf{x})$

Target QCFG

$\mathbf{p}_\theta(\mathbf{t} \mid \mathbf{s})$
Learning

- Log marginal likelihood given by:

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

Target QCFG   Source PCFG

- Exact marginalization intractable ⇒ optimize a lower bound:

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

(Score function estimator with self-critical control variate)
Inference

- Given the source MAP tree from the PCFG,

  \[ \hat{s} = \arg\max_s p_\phi(s \mid x) \]

  finding the MAP QCFG tree is intractable

  \[ \arg\max_y p_\theta(y \mid \hat{s}) \]

- Approximate decoding strategy:
  - Sample target trees \( t^{(1)}, \ldots, t^{(K)} \) from \( G[\hat{s}] \)
  - 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

<table>
<thead>
<tr>
<th>Model</th>
<th>Simple Split</th>
<th>Add Primitive</th>
<th>Add Template</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>≈ 100%</td>
<td>1.7%</td>
<td>2.5%</td>
<td>13.8%</td>
</tr>
<tr>
<td>CNN</td>
<td>≈ 100%</td>
<td>69.2%</td>
<td>56.7%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Transformer</td>
<td>≈ 100%</td>
<td>1.0%</td>
<td>53.3%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>
## Results: SCAN

<table>
<thead>
<tr>
<th>Model</th>
<th>Add Primitive</th>
<th>Add Template</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>1.7%</td>
<td>2.5%</td>
<td>13.8%</td>
</tr>
<tr>
<td>CNN</td>
<td>69.2%</td>
<td>56.7%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Transformer</td>
<td>1.0%</td>
<td>53.3%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Neural QCFG</td>
<td>96.8%</td>
<td>98.7%</td>
<td>95.7%</td>
</tr>
</tbody>
</table>

(Many other methods that also solve this dataset)
Results: SCAN
Results: SCAN

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

\[
\begin{align*}
P_0[\text{run}] & \rightarrow \text{RUN} \\
P_0[\text{look}] & \rightarrow \text{LOOK} \\
P_0[\text{walk}] & \rightarrow \text{WALK} \\
P_0[\text{jump}] & \rightarrow \text{JUMP} \\
P_0[\text{right}] & \rightarrow \text{TURN-RIGHT} \\
P_0[\text{left}] & \rightarrow \text{TURN-LEFT} \\
N_4[\text{look left}] & \rightarrow P_0[\text{left}] P_0[\text{look}] \\
N_4[\text{look right}] & \rightarrow P_0[\text{right}] P_0[\text{look}] \\
N_4[\text{walk left}] & \rightarrow P_0[\text{left}] P_0[\text{walk}] \\
N_4[\text{walk right}] & \rightarrow P_0[\text{right}] P_0[\text{walk}] \\
N_1[\text{look right twice}] & \rightarrow N_4[\text{look right}] N_4[\text{look right}] \\
N_1[\text{walk left twice}] & \rightarrow N_4[\text{walk left}] N_4[\text{walk left}] \\
N_1[\text{look thrice}] & \rightarrow N_8[\text{look thrice}] P_0[\text{look}] \\
N_1[\text{look right thrice}] & \rightarrow N_8[\text{look right thrice}] N_4[\text{look right}] \\
N_8[\text{look right thrice}] & \rightarrow N_4[\text{look right}] N_4[\text{look right}] \\
N_1[\text{walk left thrice}] & \rightarrow N_8[\text{walk left thrice}] N_4[\text{walk left}] \\
N_8[\text{walk left thrice}] & \rightarrow N_4[\text{walk left}] N_4[\text{walk left}] \\
\end{align*}
\]
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:

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

\[
v \in \sum^+
\]
## Results: StylePTB

<table>
<thead>
<tr>
<th>Transfer Type</th>
<th>Approach</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>-active to passive</td>
<td>GPT2-finetune</td>
<td>0.476</td>
<td>0.329</td>
<td>0.238</td>
<td>0.189</td>
</tr>
<tr>
<td></td>
<td>Seq2Seq</td>
<td>0.373</td>
<td>0.220</td>
<td>0.141</td>
<td>0.103</td>
</tr>
<tr>
<td></td>
<td>Retrieve-Edit</td>
<td>0.681</td>
<td>0.598</td>
<td>0.503</td>
<td>0.427</td>
</tr>
<tr>
<td></td>
<td>Human</td>
<td>0.931</td>
<td>0.881</td>
<td>0.835</td>
<td>0.795</td>
</tr>
</tbody>
</table>

- [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

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU on regular test set</th>
<th>Accuracy on <em>daxy</em> examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>25.1</td>
<td>12.5%</td>
</tr>
<tr>
<td>Neural QCFG</td>
<td>23.5</td>
<td>100%</td>
</tr>
<tr>
<td>+ BiLSTM Encoder</td>
<td>26.8</td>
<td>75.0%</td>
</tr>
</tbody>
</table>
## Results: Machine Translation

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU on regular test set</th>
<th>Accuracy on daxy examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>25.1</td>
<td>12.5%</td>
</tr>
<tr>
<td>Neural QCFG</td>
<td>23.5</td>
<td>100%</td>
</tr>
<tr>
<td>+ BiLSTM Encoder</td>
<td>26.8</td>
<td>75.0%</td>
</tr>
<tr>
<td>Transformer</td>
<td>30.4</td>
<td>100%</td>
</tr>
</tbody>
</table>

(Mostly negative results on MT)
Results: Machine Translation

N₁₃[im as tall as my father .]

N₂[im as tall as my father]

P₈[i]

N₁₂[m as tall as my father]

P₈[m]

N₁₃[as tall as my father]

je

suis

N₂[as tall as]

P₅[as]

N₂[my father]

P₄[father]

P₁[my]

mon

P₄[pere]

aussi

grand
Limitations

- Much (much) more expensive than regular seq2seq due to the $O(|N|(|N| + |P|)^2 S^3 T^3)$ 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.