Sequence-to-Sequence Learning with Latent Neural Grammars

Anonymous Author(s)
Affiliation
Address
email

Abstract

Sequence-to-sequence learning with neural networks has become the de facto standard for sequence prediction tasks. This approach typically models the local distribution over the next element with a powerful neural network that can condition on arbitrary context. While flexible and performant, these models often require large datasets for training and can fail spectacularly on benchmarks designed to test for compositional generalization. This work explores an alternative, hierarchical approach to sequence-to-sequence learning with synchronous grammars, where each node in the target tree is transduced by a subset of nodes in the source tree. The source and target trees are treated as latent and marginalized out during training. We develop a neural parameterization of the grammar which enables parameter sharing over combinatorial structures without the need for manual feature engineering. We apply this latent neural grammar to various domains—a diagnostic language navigation task designed to test for compositional generalization (SCAN), style transfer, and small-scale machine translation—and find that it performs respectably compared to standard baselines.

1 Introduction

Sequence-to-sequence learning with neural networks [53, 18, 89] encompasses a powerful and general class of methods for modeling the distribution over an output target sequence $\boldsymbol{y}$ given an input source sequence $\boldsymbol{x}$. Key to its success is a factorization of the output distribution via the chain rule coupled with a richly-parameterized neural network that models the local conditional distribution over the next element given the previous elements (and the input). While architectural innovations such as attention [7], convolutional layers [32], and transformers [93] have led to significant improvements, this element-by-element modeling remains core to the approach, and with good reason—since any distribution over the output can be factorized autoregressively via the chain rule, this approach should be able to well-approximate the true conditional distribution $p_\star(\boldsymbol{y} \mid \boldsymbol{x})$ given enough data and a powerful-enough neural network.

However, despite their excellent performance across key benchmarks these models are often sample inefficient and can moreover fail spectacularly on diagnostic tasks designed to test for compositional generalization [58, 54]. This is commonly attributed to the relatively weak inductive bias imposed by standard sequence-to-sequence models [64], which can result in learners that exhibit over-reliance on surface-form correlations rather than the underlying structure.

In this work, we explore an alternative, hierarchical approach to sequence-to-sequence learning with *latent neural grammars*. This work departs from previous approaches in three important ways. First, we model the distribution over the target sequence with a *quasi-synchronous grammar* [86] which assumes a hierarchical generative process whereby each node in the target tree is transduced by a subset of nodes in the source tree. Such node-level alignments provide provenance and a causal mechanism for how each output part is generated, thereby making the generation process more interpretable. We additionally find that the explicit modeling of source- and target-side hierarchy

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

with grammars improves compositional generalization compared to non-hierarchical autoregressive models. Second, in contrast the existing line of work on incorporating (often observed) tree structures into sequence modeling with neural networks [28, 4, 73, 30, 106, 1, 81, 14, 27, *inter alia*], we treat the source and target trees as fully *latent* and induce them during training. Finally, whereas previous work on synchronous grammars typically utilized log-linear models over handcrafted/pipelined features [16, 48, 97, 86, 94, 20, 34, *inter alia*] we make use of dense *neural* features to parameterize the grammar's rule probabilities, which enables efficient sharing of parameters over the combinatorial space of derivation rules without the need for any smoothing or feature engineering. We also use the grammar directly for end-to-end generation instead of as part of a larger pipelined system (e.g. to extract alignments) [103, 33, 11].

We apply our approach to a variety of sequence-to-sequence learning tasks—SCAN language navigation task designed to test for compositional generalization [58], style transfer on the the English Penn Treebank [63], and small-scale English-French machine translation—and find that it performs favorably compared to baseline approaches.

2 Neural Synchronous Grammars for Sequence-to-Sequence Learning

We use $\boldsymbol{x} = [x_1, \ldots, x_S]$, $\boldsymbol{y} = [y_1, \ldots, y_T]$ to denote the source/target strings, and further use $\boldsymbol{s}$, $\boldsymbol{t}$ to refer to source/target trees, represented as a set of nodes including the leaves (i.e. $\text{yield}(\boldsymbol{s}) = \boldsymbol{x}$ and $\text{yield}(\boldsymbol{t}) = \boldsymbol{y}$).

2.1 Quasi-Synchronous Grammars

Quasi-synchronous grammars, introduced by Smith and Eisner [86], define a monolingual grammar over target strings conditioned on a source tree, where the grammar's rule set depends dynamically on the source tree $\boldsymbol{s}$. In this paper we work with probabilistic quasi-synchronous context-free grammars (QCFG), which can be represented as a tuple $G[\boldsymbol{s}] = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}[\boldsymbol{s}], \theta)$ where S is the distinguished start symbol, $\mathcal{N}$ is the set of nonterminals which expand to other nonterminals, $\mathcal{P}$ is the set of nonterminals which expand to terminals (i.e. preterminals), Σ is the set of terminals, and $\mathcal{R}[\boldsymbol{s}]$ is a set of context-free rules conditioned on $\boldsymbol{s}$, where each rule is one of

$$
\begin{array}{ll}
S \to A[\alpha_i], & A \in \mathcal{N}, \quad \alpha_i \subseteq \boldsymbol{s} \\
A[\alpha_i] \to B[\alpha_j] C[\alpha_k], & A \in \mathcal{N}, \quad B, C \in \mathcal{N} \cup \mathcal{P}, \quad \alpha_i, \alpha_j, \alpha_k \subseteq \boldsymbol{s} \\
D[\alpha_i] \to w, & D \in \mathcal{P}, w \in \Sigma, \quad \alpha_i \subseteq \boldsymbol{s}.
\end{array}
$$

We use θ to parameterize the rule probabilities $p_\theta(r)$ for each $r \in \mathcal{R}[\boldsymbol{s}]$. In the above, α_i's are subsets of nodes in the source tree $\boldsymbol{s}$, and therefore QCFGs model a tree transduction process where each target tree node is aligned to a subset of source tree nodes. This quasi-synchronous generation process generalizes classic synchronous context-free grammars [100] and relaxes the assumption that the source tree is isormorphic to the target tree.[1] Since the α_i's take values in the power set of $\boldsymbol{s}$, the above formulation as presented is completely intractable. We follow prior work [86, 94] and restrict $\alpha_i, \alpha_j, \alpha_k \in \boldsymbol{s}$, which amounts to assuming that each target tree node can be aligned to exactly one source tree node.

In contrast to standard, "flat" sequence-to-sequence models where any hierarchical structure necessary for the task must be captured implicitly within the network's hidden layers, synchronous grammars explicitly model the hierarchical structure on both the source and target side, which acts as a strong source of inductive bias. This tree transduction process further results in a more interpretable generation process as each span in the target aligned to a span in the source via node-level alignments.[2] More generally, the grammar rules provide a symbolic interface to the model, and we show how this mechanism can be used to, for example, incorporate phrase-to-phrase copy mechanisms (section 2.4).

2.2 Parameterization

Since each source tree node α_i is likely to occur only a few times (or just once) in the training corpus, parameter sharing becomes crucial. Prior work on QCFGs typically utilized log-linear models over

[1] It is also possible to use richer grammatical formalisms to model syntactic divergences [84, 66]. However these approaches require more expensive algorithms for learning and inference.

[2] Similarly, latent variable attention [102, 6, 21, 80, 101] also provides for more a interpretable generation process than standard soft attention via explicit alignments. However, latent variable attention can only model word-to-word alignments, in contrast to synchronous grammars which can model phrase-to-phrase alignments.

hand-crafted features to share parameters across rules [86, 34]. In this work we instead work with a neural parameterization which allows for parameter sharing without the need for manual feature engineering.

Concretely, we first represent each symbol $A[\alpha_i]$ as an embedding,

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

where $\mathbf{u}_A$ is the nonterminal embedding for A, and $\mathbf{h}_{\alpha_i}$ is the representation of node α_i given by running a TreeLSTM over the source tree $\boldsymbol{s}$ [90, 112]. These embeddings are then combined to produce the probability of each rule,

$$\begin{aligned}
p_\theta(S \to A[\alpha_i]) &\propto \exp\left(\mathbf{u}_S^\top \mathbf{e}_{A[\alpha_i]}\right), \\
p_\theta(A[\alpha_i] \to B[\alpha_j]C[\alpha_k]) &\propto \exp\left(f_1(\mathbf{e}_{A[\alpha_i]})^\top (f_2(\mathbf{e}_{B[\alpha_j]}) + f_3(\mathbf{e}_{C[\alpha_k]}))\right), \\
p_\theta(D[\alpha_i] \to w) &\propto \exp\left(f_4(\mathbf{e}_{D[\alpha_i]})^\top \mathbf{u}_w + b_w\right),
\end{aligned}$$

where f_1, f_2, f_3, f_4 are feedforward networks with residual layers (see appendix A.1 for the exact parameterization). Therefore the parameters of this model are the nonterminal embeddings (i.e. $\mathbf{u}_A$ for $A \in \{S\} \cup \mathcal{N} \cup \mathcal{P}$), terminal embeddings/biases (i.e. $\mathbf{u}_w, b_w$ for $w \in \Sigma$), and the parameters of the TreeLSTM and the feedforward networks.

2.3 Learning and Inference

The QCFG described above defines a distribution over target trees (and by marginalization, target side strings) given a source tree. While prior work on QCFGs typically relied on an off-the-shelf parser over the source to obtain its parse tree, this limits the generality of the approach. In this work, we instead learn a probabilistic source-side parser along with the QCFG. This parser is a monolingual PCFG with parameters ϕ that defines a posterior distribution over binary parse trees given source strings, i.e. $p_\phi(\boldsymbol{s} \mid \boldsymbol{x})$. Our PCFG uses the parameterization from [55]. With the parser in hand, we are now ready to define the log marginal likelihood,

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

Here $\mathcal{T}(\boldsymbol{x})$ and $\mathcal{T}(\boldsymbol{y})$ are the set of trees whose leaves are $\boldsymbol{x}$ and $\boldsymbol{y}$. Unlike classic synchronous context-free grammars, full marginalization over both $\mathcal{T}(\boldsymbol{y})$ and $\mathcal{T}(\boldsymbol{x})$ is intractable. However, we observe that the inner summation $\sum_{\boldsymbol{t} \in \mathcal{T}(\boldsymbol{y})} p_\theta(\boldsymbol{t} \mid \boldsymbol{s}) = p_\theta(\boldsymbol{y} \mid \boldsymbol{s})$ is tractable to compute with dynamic programming (i.e. the inside algorithm [8]) in $\mathcal{O}(|\mathcal{N}|(|\mathcal{N}| + |\mathcal{P}|)^2 S^3 T^3)$, where S is the source length and T is the target length.[3] This motivates the following lower bound on the log marginal likelihood,

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

which is obtained by the usual application of Jensen's inequality (see appendix A.2).[4]

An unbiased Monte Carlo estimator for the gradient with respect to θ is straightforward to compute given a sample from $p_\phi(\boldsymbol{s} \mid \boldsymbol{x})$, since we can just backpropagate through the inside algorithm. For the gradient respect to ϕ, we use the score function estimator with a self-critical baseline [76],

$$\nabla_\phi\, \mathbb{E}_{\boldsymbol{s} \sim p_\phi(\boldsymbol{s} \mid \boldsymbol{x})} \left[\log p_\theta(\boldsymbol{y} \mid \boldsymbol{s})\right] \approx \left(\log p_\theta(\boldsymbol{y} \mid \boldsymbol{s}') - \log p_\theta(\boldsymbol{y} \mid \widehat{\boldsymbol{s}})\right) \nabla_\theta \log p(\boldsymbol{s}' \mid \boldsymbol{x}),$$

where $\boldsymbol{s}'$ is a sample from $p_\phi(\boldsymbol{s} \mid \boldsymbol{x})$ and $\widehat{\boldsymbol{s}}$ is the most likely tree in $p_\phi(\boldsymbol{s} \mid \boldsymbol{x})$. We also found it important to regularize the source parser by simultaneously training it as a monolingual PCFG, and therefore add $\nabla_\phi \log p_\phi(\boldsymbol{x})$ to the gradient expression above.[5] Obtaining the sample tree $\boldsymbol{s}'$, the

[3] This run time is the same as the bitext inside algorithm for classic synchronous context-free grammars.

[4] It is possible to tighten this bound with the use of a variational distribution $q_\psi(\boldsymbol{s} \mid \boldsymbol{x}, \boldsymbol{y})$, which results in the following evidence lower bound,

$$\log\, p_{\theta,\phi}(\boldsymbol{y} \mid \boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{s} \sim q_\psi(\boldsymbol{s} \mid \boldsymbol{x}, \boldsymbol{y})} \left[\log p_\theta(\boldsymbol{y} \mid \boldsymbol{s})\right] - \mathrm{KL}[q_\psi(\boldsymbol{s} \mid \boldsymbol{x}, \boldsymbol{y}) \,\|\, p_\phi(\boldsymbol{s} \mid \boldsymbol{x})].$$

Our use of the non-variational lower bound is implicitly assuming that $\boldsymbol{s}$ and $\boldsymbol{y}$ are conditionally independent given $\boldsymbol{x}$ (i.e. all uncertainty about $\boldsymbol{s}$ is captured by $\boldsymbol{x}$), which is reasonable for many language applications.

[5] This motivates our use of a generative rather than a discriminative parser on the source side.

argmax tree $\widehat{\boldsymbol{s}}$, and scoring the sampled tree $\log p(\boldsymbol{s}' \mid \boldsymbol{x})$ all require $\mathcal{O}(S^3)$ dynamic programs. Hence the runtime is still dominated by the $\mathcal{O}(S^3T^3)$ dynamic program to compute $p_\theta(\boldsymbol{y} \mid \boldsymbol{s}')$ and $p_\theta(\boldsymbol{y} \mid \widehat{\boldsymbol{s}})$. We found this to be manageable on modern GPUs with a vectorized implementation of the inside algorithm. Our implementation uses the Torch-Struct library [77].

Predictive inference For decoding, we first run MAP inference with the source parser to obtain $\widehat{\boldsymbol{s}} = \operatorname{argmax}_{\boldsymbol{s}} p_\phi(\boldsymbol{s} \mid \boldsymbol{x})$. Given $\widehat{\boldsymbol{s}}$, finding the most probable sequence $\operatorname{argmax}_{\boldsymbol{y}} p_\theta(\boldsymbol{y} \mid \widehat{\boldsymbol{s}})$ (i.e. the consensus string of the grammar $G[\widehat{\boldsymbol{s}}]$) is still difficult, and in fact NP-hard [85, 13, 62]. We therefore resort to an approximate decoding scheme where we sample K target trees $\boldsymbol{t}^{(1)}, \dots \boldsymbol{t}^{(K)}$ from $G[\widehat{\boldsymbol{s}}]$, rescore the yields of the sampled trees, and return the tree whose yield has the lowest perplexity.

2.4 Extensions

Here we show that the the formalism of synchronous grammars provides a flexible interface with which to interact with the model and imbue inductive biases.

Phrase-to-phrase copying Incorporating latent copy mechanisms into sequence-to-sequence models has led to significant improvements for tasks where there is overlap between the source and target sequences [49, 68, 40, 39, 79]. These models typically define a binary latent variable at each time step that decides to either copy from the source or generate from the target vocabulary. While useful, existing copy mechanisms can typically copy at only the word-level due to the word-level encoder/decoder.[6] In contrast, the hierarchical generative process of QCFGs makes it convenient to incorporate phrase-level copy mechanisms by using a special-purpose nonterminal/preterminal that always copies the yield of the source subtree that it is combined with. In particular, letting $A_{\text{COPY}} \in \mathcal{N}$ be a COPY nonterminal, we can expand the rule set $\mathcal{R}[\boldsymbol{s}]$ to include rules of the form $A_{\text{COPY}}[\alpha_i] \to v$ for $v \in \Sigma^+$, and define its probability to be

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

The preterminal copy mechanism is similarly defined. Computing $p_\theta(\boldsymbol{y} \mid \boldsymbol{s})$ in this modified grammar requires a straightforward modification of the usual inside algorithm.[7] In our style transfer experiments in section 3.2 we show that this phrase-level copying is crucial to obtaining good performance. While not explored in the present work, such a mechanism can readily be employed to embed known transformations rules (e.g. from bilingual lexicons or transliteration tables) into the modeling process.

Adding domain-specific constraints For some applications we may want to place additional restrictions on the rule set to operationalize domain-specific constraints and inductive biases. For example, setting $\alpha_j, \alpha_k \in \text{descendant}(\alpha_i)$ for rules of the form $A[\alpha_i] \to B[\alpha_j]C[\alpha_k]$ would constrain the target tree hierarchy to respect the source tree hierarchy, while restricting α_i to terminal nodes only (i.e. $\alpha_i \in \text{yield}(\boldsymbol{s})$) for rules of the form $D[\alpha_i] \to w$ would enforce that each target terminal be aligned to a source terminal. We indeed make use of such restrictions in our experiments.

Incorporating autoregressive language models Finally, we show that a simple extension of the QCFG can incorporate standard autoregressive language models. Let $p_{\text{LM}}(w \mid \gamma)$ be a distribution over the next element given by a (potentially conditional) language model given arbitrary context γ (e.g. $\gamma = \boldsymbol{y}_{<t}$ for a monolingual language model and $\gamma = [\boldsymbol{x}; \boldsymbol{y}_{<t}]$ for a sequence-to-sequence model). We can then add a special LM preterminal $D_{\text{LM}} \in \mathcal{P}$ with symbol embedding $\mathbf{u}_{D_{\text{LM}}}$ that is not combined with any source node, and define the associated probabilities to be,

$$\begin{aligned}
p_\theta(A[\alpha_i] \to D_{\text{LM}} C[\alpha_k]) &\propto \exp\left(f_1(\mathbf{e}_{A[\alpha_i]})^\top (f_2(\mathbf{u}_{D_{\text{LM}}}) + f_3(\mathbf{e}_{C[\alpha_k]}))\right), \\
p_\theta(A[\alpha_i] \to B[\alpha_j] D_{\text{LM}}) &\propto \exp\left(f_1(\mathbf{e}_{A[\alpha_i]})^\top (f_2(\mathbf{e}_{B[\alpha_j]}) + f_3(\mathbf{u}_{D_{\text{LM}}}))\right), \\
p_\theta(D_{\text{LM}} \to w) &= p_{\text{LM}}(w \mid \gamma).
\end{aligned}$$

Both the QCFG and the language model can be trained end-to-end against the log marginal likelihood. While this modified model can embed flexible sequence models within a QCFG,[8] it also inherits

[6] However see Panthaplackel et al. [72] and Wiseman et al. [96] for recent exceptions.

[7] Letting $\beta[s, t, N] = p_\theta(N \overset{*}{\to} \boldsymbol{y}_{s:t})$ be the inside variable for N deriving $\boldsymbol{y}_{s:t}$, we can simply force $\beta[s, t, A_{\text{copy}}[\alpha_i]] = \mathbb{1}\{\boldsymbol{y}_{s:t} = \text{yield}(\alpha_i)\}$.

[8] Alternatively, it is also possible embed a QCFG within an autoregressive language model with a binary switch variable z_t at each time step. This variable (with distribution $p_{\text{LM}}(z_t \mid \gamma)$) selects between $p_{\text{LM}}(y_t \mid \gamma)$ and $p_\theta(y_t \mid \boldsymbol{s}, \boldsymbol{y}_{<t})$, where the latter next-word probability distribution in the QCFG can be computed with a probabilistic Earley parser [88]. The difference between the two approaches stems from whether the switch decision is made by the QCFG or by the language model.

many of their attendant issues (e.g. left-to-right generation). In preliminary attempts to incorporate sequence-to-sequence models into the QCFG we found that this combined model quickly degenerated into the uninteresting case of always using the autoregressive model, and hence did not pursue this further. However it is possible that modifications to the approach (e.g. separate pretraining of both models followed by joint finetuning, or using a pretrained monolingual language model that remains fixed) could lead to improvements.

3 Experiments

We apply the neural QCFG described above to a variety of sequence-to-sequence learning tasks. These experiments are not intended to push the state-of-the-art on these tasks but rather intended to assess whether our approach performs respectably against standard baselines while simulatenously learning interesting and interpretable structures.

3.1 SCAN

We first experiment on SCAN [58], a diagnostic dataset where a model has to learn to translate simple English commands to actions (e.g. jump twice after walk $\Longrightarrow$ WALK JUMP JUMP). While conceptually simple, standard sequence-to-sequence models have been shown to fail on splits of the data designed to test for compositional generalization. We focus on four commonly-used splits: (1) *simple*, where train/test split is random, (2) *add primitive (jump)*, where the primitive command jump is seen in isolation in training and must combine with other commands during testing,[9] (3) *add template (around right)*, where the template around right is not seen during training, and (4) *length*, where the model is trained on action sequences of length at most 22 and tested on action sequences of length between 24 and 48. In these experiments, the nonterminals $A \in \mathcal{N}$ are only combined with source nodes that govern at least two nodes, and the preterminals $P \in \mathcal{P}$ are only combined with source terminals. We set $|\mathcal{N}| = 10$ and $|\mathcal{P}| = 1$, and place two additional restrictions on the rule set. First, for rules of the form $S \to A[\alpha_i]$ we restrict α_i to always be the root of the source tree. Second, for rules of the form $A[\alpha_i] \to B[\alpha_j]C[\alpha_k]$ we restrict α_j, α_k to be descendants of α_i, or α_i itself (i.e. $\alpha_j, \alpha_k \in \text{descendant}(\alpha_i) \cup \{\alpha_i\}$). These restrictions operationalize the constraint that the target tree hierarchy respects the source tree hierarchy, though still in a much looser sense than in an isomorphism. See appendix A.3.1 for the full experimental setup and hyperparameters.

Approach	Simple	Jump	A. Right	Length
Uses SCAN-specific knowledge				
Equivar. Seq2Seq [36]	100.0	99.1	92.0	15.9
Span-based SP [46]	100.0	–	100.0	–
NeSS [15]	100.0	100.0	100.0	100.0
LANE [61]	100.0	100.0	100.0	100.0
Program Synth. [70]	100.0	100.0	100.0	100.0
NQG-T5 [82]	100.0	100.0	–	100.0
Domain-agnostic				
RNN [58]	99.7	1.7	2.5	13.8
CNN [22]	100.0	69.2	56.7	0.0
Transformer [31]	–	1.0	53.3	0.0
T5-base [31]	–	99.5	33.2	14.4
Syntactic Attn [78]	100.0	91.0	28.9	15.2
Meta Seq2Seq [57]	–	99.9	99.9	16.6
CGPS [60]	99.9	98.8	83.2	20.3
GECA [5]	–	87.0	82.0	–
R&R Data Aug. [3]	–	88.0	82.0	–
Neural QCFG (ours)	96.9	96.8	98.7	95.7

Table 1: Results on SCAN. We separate out between approaches that exploit SCAN-specific information (top) versus those that do not (bottom).

Results Table 1 shows our results against various baselines on SCAN. While many approaches are able to solve this dataset almost perfectly, they often exploit SCAN-specific knowledge (e.g. that walk is an action word), which precludes their straightforward application to other domains. Among the domain-agnostic approaches that do not use SCAN-specific information, the neural QCFG performs respectably, especially on the challenging *length* split of the data. Figure 1 shows an example generation from the test set of the *add primitive (jump)* split, where we observe that the model seems to have learned sensible transduction rules (e.g. N_1[run left twice] $\to$ N_4[run left] N_4[run left]). The node-level alignments further provide explicit provenance for each target span and makes the generation process more interpretable than standard attention mechanisms.

[9] The QCFG defined in this paper places zero probability on length-one target strings, which presents an issue for this split of SCAN where jump $\Longrightarrow$ JUMP is the only context in which JUMP occurs in the training set. For length-one target strings we simply replicate the source and target strings, i.e. jump $\Longrightarrow$ JUMP becomes jump jump $\Longrightarrow$ JUMP JUMP.

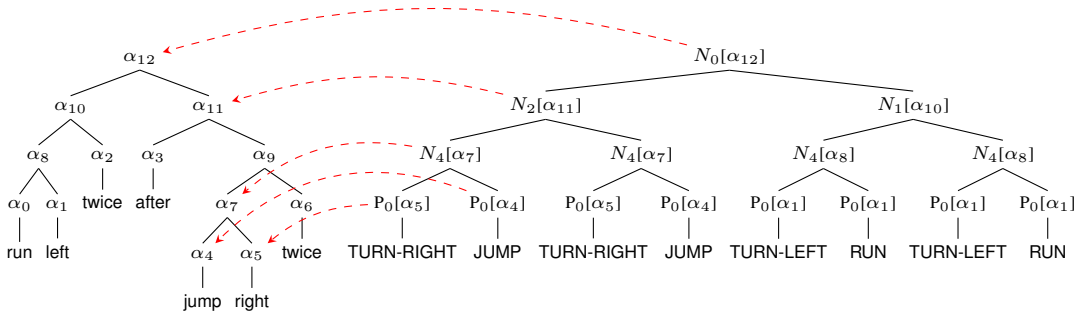

Figure 1: Generation from the neural QCFG on a test example from the *add primitive (jump)* split of SCAN. The induced tree from the learned source parser is shown on the left, and the target tree derivation is shown on the right. While the model does not distinguish between preterminals and terminals on the source tree, we have shown them separately for additional clarity. We also show some of the node-level alignments with dashed lines.

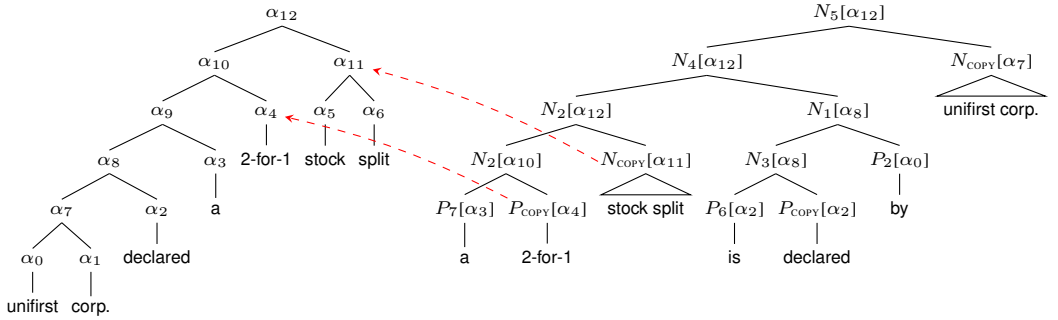

Figure 2: A test example from the *active to passive* style transfer task on the Penn Treebank. The induced tree from the learned source parser is shown on the left, and the target tree derivation is shown on the right. The source tree is linguistically incorrect but the model is still able to correctly tranduce the output. Some examples of copy nonterminals/preterminals and their aligned source nodes are shown with dashed arrows.

3.2 Style Transfer

We next apply our approach on style transfer on English utilizing the StylePTB dataset from Lyu et al. [63]. We focus on the three *hard transfer* tasks identified by the original authors: (1) *active to passive* (2808 examples), (2) *adjective emphasis* (696 examples), and (3) *verb/action emphasis* (1201 examples). The main difficulty with these tasks stems from the small training set combined with the relative complexity of these tasks. We set $|\mathcal{N}| = |\mathcal{P}| = 8$ and use the same restrictions on the rule set as in the SCAN experiments. We also experiment with the phrase-to-phrase copy mechanism as described in section 2.4. The original paper provides several strong baselines: finetuned GPT2, a standard sequence-to-sequence model, and the retrieve-and-edit model from Hashimoto et al. [45]. We also train our own baseline sequence-to-sequence models with a (word-level) copy mechanism. See appendix A.3.2 for more details.

Results Table 2 shows the results,[10] where we observe that the neural QCFG performs well compared to the various baselines. We further find that incorporating the copy mechanism improves results substantially for both the baseline LSTM and the neural QCFG.[11] Figure 2 shows a test example from the *active-to-passive* task, which shows the word- and phrase-level copying mechanism in action. In this example the source tree is linguistically incorrect, but the grammar is nonetheless able to appropriately transduce the output. Given that linguistic phrases are generally more likely to remain unchanged in these types tasks, biasing the model towards copying longer phrases (e.g. via posterior regularization) could potentially improve results. For example in figure 2 the ideal case would involve copying the phrase a 2-for-1 stock split, but this is not possible due to the incorrectly predicted source tree.

[10] As in the original paper we calculate the automatic evaluation metrics using the nlg-eval library, available at https://github.com/Maluuba/nlg-eval.

[11] Even for models that do not explicitly use the copy mechanism, we indirectly allow for copying by replacing the ⟨unk⟩ token with the source token that the preterminal is combined with in the neural QCFG case, or the source token that had the maximum attention weight in the LSTM case. This explains the outperformance of our baseline sequence-to-sequence models compared to the baselines from Lyu et al. [63].

Transfer Type	Approach	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr
Active to Passive	GPT2-finetune	0.476	0.329	0.238	0.189	0.216	0.464	1.820
	Seq2Seq	0.373	0.220	0.141	0.103	0.131	0.345	0.845
	Retrieve-Edit	0.681	0.598	0.503	0.427	0.383	0.663	4.535
	Human	0.931	0.881	0.835	0.795	0.587	0.905	8.603
	Seq2Seq-LSTM	0.505	0.349	0.253	0.190	0.235	0.475	2.000
	Neural QCFG	0.431	0.637	0.548	0.472	0.415	0.695	4.294
	Seq2Seq-LSTM + copy	0.838	0.735	0.673	0.598	0.467	0.771	5.941
	Neural QCFG + copy	0.836	0.771	0.713	0.662	0.499	0.803	6.410
Adj. Emphasis	GPT2-finetune	0.263	0.079	0.028	0.000	0.112	0.188	0.386
	Seq2Seq	0.187	0.058	0.018	0.000	0.059	0.179	0.141
	Retrieve-Edit	0.387	0.276	0.211	0.164	0.193	0.369	1.679
	Human	0.834	0.753	0.679	0.661	0.522	0.811	6.796
	Seq2Seq-LSTM	0.332	0.333	0.051	0.000	0.142	0.27	0.845
	Neural QCFG	0.348	0.178	0.062	0.000	0.162	0.317	0.667
	Seq2Seq-LSTM + copy	0.505	0.296	0.184	0.119	0.242	0.514	1.839
	Neural QCFG + copy	0.676	0.506	0.393	0.316	0.373	0.683	3.424
Verb Emphasis	GPT2-finetune	0.309	0.170	0.095	0.041	0.140	0.292	0.593
	Seq2Seq	0.289	0.127	0.066	0.038	0.098	0.275	0.300
	Retrieve-Edit	0.416	0.284	0.209	0.148	0.223	0.423	1.778
	Human	0.649	0.569	0.493	0.421	0.433	0.693	5.668
	Seq2Seq-LSTM	0.355	0.152	0.083	0.043	0.151	0.320	0.530
	Neural QCFG	0.431	0.250	0.140	0.073	0.219	0.408	1.097
	Seq2Seq-LSTM + copy	0.526	0.389	0.294	0.214	0.294	0.464	2.346
	Neural QCFG + copy	0.664	0.512	0.407	0.319	0.370	0.589	3.227

Table 2: Results on the *hard* style transfer tasks from the StylePTB dataset [63]. For each transfer type, the top four rows are from Lyu et al. [63], while the bottom four rows are from this paper.

3.3 Machine Translation

Our final experiment is on a small-scale English-French machine translation dataset from Lake and Baroni [58]. Here we are interested in evaluating the model in two ways: first, as a standard machine translation system on a randomly held out test set, and second, to see if the model can systematically generalize to previously unseen combinations. To assess the latter, Lake and Baroni [58] add 1000 repetitions of i am daxy $\implies$ je suis daxiste to the training set and test on 8 new sentences that use daxy in novel combinations (e.g. he is daxy $\implies$ it est daxiste and i am not daxy $\implies$ je ne sui pas daxiste). We use 9000 examples for training (1000 of which is the i am daxy example) and 1000 examples each for validation/test.

For these experiments, we set $|\mathcal{N}| = |\mathcal{P}| = 6$ and combine all source nodels with all nonterminals/preterminals. We place two restrictions on the rule set: for rules of the form $S \to A[\alpha_i]$ we restrict α_i to be the root of the source tree (as in the previous experiments), and for rules of the form $A[\alpha_i] \to B[\alpha_j]C[\alpha_k]$ we restrict α_j, α_k to be the direct children of α_i, or α_i itself if α_i has no children.[12] See appendix A.3.3 for the full experimental setup and hyperparameters.

Model	BLEU	daxy acc.
Seq2Seq-LSTM	28.4	12.5%
CGPS [60]	23.5	100%
Neural QCFG	30.7	87.5%

Table 3: Machine translation results.

Results Table 3 shows the BLEU on the test set of 1000 examples, as well as the accuracy on the 8 daxy sentences. Compared to the baseline LSTM attention model, the neural QCFG performs well on the regular test set (as measured by BLEU) as well as the daxy test set. CGPS [60], which learns separate representations for generating attention distributions vs. the output, performs particularly well on the daxy test set, but not as well on the regular test set. Figure 3 shows several examples of target tree derivations. As in the style transfer experiments, the source trees here would not be considered conventionally correct (the period is typically attached to the root), but the QCFG model is still able to correctly transduce the output by also learning unconventional trees on the target side as well.

[12] These restrictions are closer to the strict isormorphic requirement in synchronous context-free grammars than in the previous case. However they still allow for non-isormorphic trees since α_i can be inherited if it has no children.

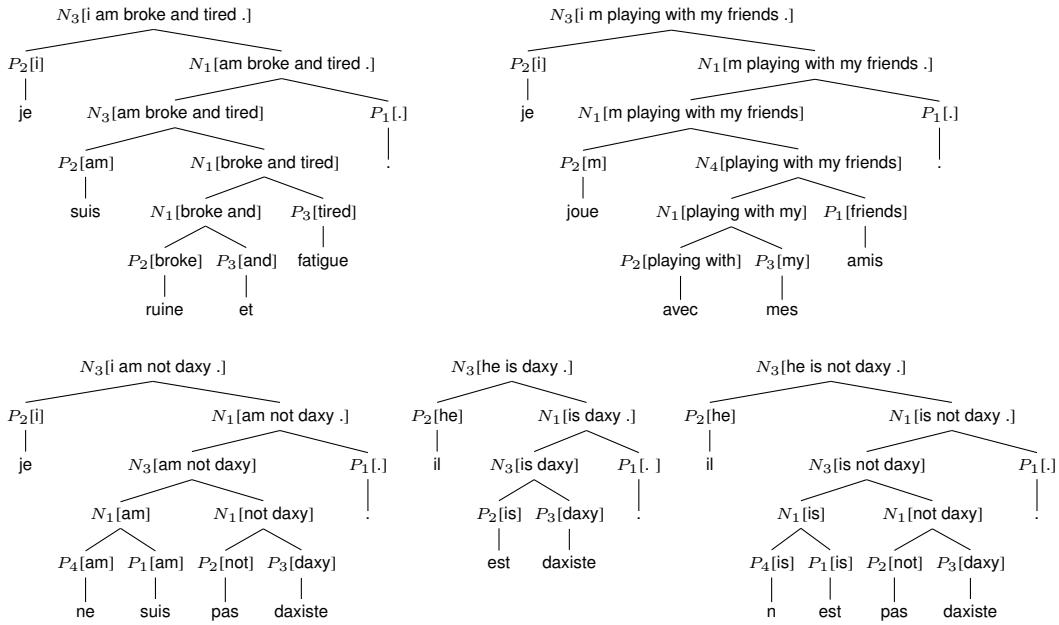

Figure 3: Target tree derivations from the English-French machine translation experiments. The top two trees are from the regular test set, while the bottom three trees are from the daxy test set. We do not explicitly show the source trees here and instead show the source phrases as arguments to the target tree nonterminals/preterminals.

4 Discussion

Neural quasi-synchronous grammars provide a general-purpose framework for many sequence-to-sequence learning tasks. On the positive side, our work can potentially pave the path towards more interpretable neural generative models that can provide explanations for their decisions via explicit node alignments and derivation rules. Further, by exposing the model's internals through the use of latent nonterminals, it also may be possible to achieve some level of control via manipulation of these variables. Such control mechanisms could, for example, be used to mitigate pronouns biases of that often exist in current neural generation systems.

However, several limitations remain. For one, the $O(S^3T^3)$ dynamic program will likely pose challenges in scaling this approach to larger datasets with longer sequences. The conditional independence assumptions made by the grammar also may not be appropriate for some tasks that involve complex dependencies. Finally, QCFGs may not be the most appropriate tool for domains in which the input/output is not naturally represented with tree structures, although it may be possible to extend these formalisms to consider (for example) graph-structured data.

5 Related Work

Synchronous grammars Synchronous grammars and tree transducers have a long and rich history in natural language processing [2, 84, 100, 65, 29, 25, 69, 48, 97, 37, 10, 19, *inter alia*]. In this work we focus on the formalism of quasi-synchronous grammars, which relaxes the assumption that the source and target tree be isomorphic. Quasi-synchronous grammars have enjoyed applications across many domains including in machine translation [86, 34, 35], question answering [94], paraphrase detection [20], sentence simplification [99, 98], and parser projection [87]. Prior work on quasi-synchronous grammars generally relied on pipelined parse trees for the source and only marginalized out the target tree, in contrast to the present work which treats both source and target trees as latent.

Compositional sequence-to-sequence learning Lake and Baroni [58] proposed the influential SCAN dataset for assessing the compositional generalization capabilities of neural sequence-to-sequence models. There has since been a large body of work on compositional sequence-to-sequence learning through various approaches including architectural modifications [60, 78, 36], grammars and neuro-symbolic models [71, 82, 70, 15, 61], meta-learning [57], and data augmentation [5, 41, 42, 3].

Deep latent variable models There has much recent work on neural parameterizations of classic probabilistic latent variable models including hidden Markov models [92, 95, 17], topic models [67, 23, 24], dependency models [50, 43, 12, 44, 104], and context-free grammars [55, 52, 111, 110, 105]. These works essentially extend feature-based unsupervised learning [9] to the neural case with the use of neural networks over embedding parameterizations, which makes it possible to easily condition the generative model on side information such as auxiliary latent variables [44, 55], images [109, 51, 47], videos [108], and source-side context [95, 83]. Since we marginalize over unobserved trees during learning, our work is also related to the line of work on marginalizing out latent variables/structures for sequence transduction tasks [38, 26, 75, 56, 107, 74, 101, 21, 59, 91].

6 Conclusion

In this paper we have studied sequence-to-sequence learning with latent neural grammars. We have shown that the formalism quasi-synchronous grammars provides a powerful tool with which to imbue inductive biases, operationalize domain-specific constraints, and interface with the model. Our approach performs favorably against standard baselines when applied to a variety of sequence-to-sequence learning tasks.

Future work in this area will consider: (1) revisiting richer grammatical formalisms (e.g. synchronous tree-adjoining grammars) with contemporary techniques parameterization and inference, (2) combining such structured models with pretrained language models, and (3) applying these methods to other structured domains such as programs and graphs. More generally, incorporating grammatical formalisms and symbolic systems into neural networks remains an exciting avenue for much future work.

References

[1] Roee Aharoni and Yoav Goldberg. Towards String-to-Tree Neural Machine Translation. In *Proceedings of ACL*, 2017.

[2] Alfred V. Aho and Jeffrey D. Ullman. Syntax Directed Translations and the Pushdown Assembler. *Journal of Computer and System Sciences*, 3:37–56, 1969.

[3] Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. Learning to Recombine and Resample Data for Compositional Generalization. In *Proceedings of ICLR*, 2021.

[4] David Alvarez-Melis and Tommi S. Jaakkola. Tree-structured Decoding with Doubly-Recurrent Neural Networks. In *Proceedings of ICLR*, 2017.

[5] Jacob Andreas. Good-Enough Compositional Data Augmentation. In *Proceedings of ACL*, 2020.

[6] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple Object Recognition with Visual Attention. In *Proceedings of ICLR*, 2015.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*, 2015.

[8] James K. Baker. Trainable Grammars for Speech Recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*, 1979.

[9] Taylor Berg-Kirkpatrick, Alexandre Bouchard-Cote, John DeNero, and Dan Klein. Painless Unsupervised Learning with Features. In *Proceedings of NAACL*, 2010.

[10] Phil Blunsom, Trevor Cohn, and Miles Osborne. Bayesian Synchronous Grammar Induction. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Proceedings of NeurIPS*, 2009.

[11] David Burkett, John Blitzer, and Dan Klein. Joint Parsing and Alignment with Weakly Synchronized Grammars. In *Proceedings of NAACL*, 2010.

[12] Jan Buys and Phil Blunsom. Neural Syntactic Generative Models with Exact Marginalization. In *Proceedings of NAACL*, 2018.

[13] F. Casacuberta and Colin De La Higuera. Computational complexity of problems on probabilistic grammars and transducers. In *Grammatical Inference: Algorithms and Applications*, volume 1891, pages 15–24. Springer, 2000.

[14] Xinyun Chen, Chang Liu, and Dawn Song. Tree-to-tree Neural Networks for Program Translation. In *Proceedings of NeurIPS*, 2018.

[15] Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. Compositional Generalization via Neural-Symbolic Stack Machines. In *Proceedings of NeurIPS*, 2020.

[16] David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, 2005.

[17] Justin T. Chiu and Alexander M. Rush. Scaling Hidden Markov Language Models. In *Proceedings of EMNLP*, 2020.

[18] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014.

[19] Trevor Cohn and Mirella Lapata. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34(1):637–674, 2009.

[20] Dipanjan Das and Noah A. Smith. Paraphrase Identification as Probabilistic Quasi-Synchronous Recognition. In *Proceedings of ACL*, 2009.

[21] Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander M. Rush. Latent Alignment and Variational Attention. In *Proceedings of NeurIPS*, 2018.

[22] Roberto Dessì and Marco Baroni. CNNs found to jump around more skillfully than RNNs: Compositional generalization in seq2seq convolutional networks. In *Proceedings of ACL*, 2019.

[23] Adji B. Dieng, Francisco J. R. Ruiz, and David M. Blei. The Dynamic Embedded Topic Model. *arXiv:1907.05545*, 2019.

[24] Adji B. Dieng, Francisco J. R. Ruiz, and David M. Blei. Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 8, 2020.

[25] Yuan Ding and Martha Palmer. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL*, 2005.

[26] Markus Dreyer, Jason Smith, and Jason Eisner. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of EMNLP*, 2008.

[27] Wenchao Du and Alan W Black. Top-Down Structurally-Constrained Neural Response Generation with Lexicalized Probabilistic Context-Free Grammar. In *Proceedings of NAACL*, 2019.

[28] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent Neural Network Grammars. In *Proceedings of NAACL*, 2016.

[29] Jason Eisner. Learning Non-Isomorphic Tree Mappings for Machine Translation. In *Proceedings of ACL*, 2003.

[30] Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. Learning to Parse and Translate Improves Neural Machine Translation. In *Proceedings of ACL*, 2017.

[31] Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures. *arXiv:2007.08970*, 2020.

[32] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of ICML*, 2017.

[33] Daniel Gildea. Loosely Tree-Based Alignment for Machine Translation. In *Proceedings of ACL*, 2003.

[34] Kevin Gimpel and Noah A. Smith. Feature-rich translation by quasi-synchronous lattice parsing. In *Proceedings of EMNLP*, 2009.

[35] Kevin Gimpel and Noah A. Smith. Quasi-synchronous phrase dependency grammars for machine translation. In *Proceedings of EMNLP*, 2011.

[36] Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. Permutation Equivariant Models for Compositional Generalization in Language . In *Proceedings of ICLR*, 2020.

[37] Jonathan Graehl, Kevin Knight, and Jonathan May. Training tree transducers. *Computational Linguistics*, 34(3):391–427, 2008.

[38] Alex Graves. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of ICML*, 2006.

[39] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. 2016.

[40] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the Unknown Words. In *Proceedings of ACL*, 2016.

[41] Demi Guo, Yoon Kim, and Alexander Rush. Sequence-level mixed sample data augmentation. In *Proceedings of EMNLP*, 2020.

[42] Yinuo Guo, Hualei Zhu, Zeqi Lin, Bei Chen, Jian-Guang Lou, and Dongmei Zhang. Revisiting Iterative Back-Translation from the Perspective of Compositional Generalization. In *Proceedings of AAAI*, 2021.

[43] Wenjuan Han, Yong Jiang, and Kewei Tu. Dependency Grammar Induction with Neural Lexicalization and Big Training Data. In *Proceedings of EMNLP*, 2017.

[44] Wenjuan Han, Yong Jiang, and Kewei Tu. Enhancing Unsupervised Generative Dependency Parser with Contextual Information. In *Proceedings of ACL*, 2019.

[45] Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy Liang. A Retrieve-and-Edit Framework for Predicting Structured Outputs. In *Proceedings of NeurIPS*, 2018.

[46] Jonathan Herzig and Jonathan Berant. Span-based Semantic Parsing for Compositional Generalization. *arXiv:2009.06040*, 2020.

[47] Yining Hong, Qing Li, Song-Chun Zhu, and Siyuan Huang. VLGrammar: Grounded Grammar Induction of Vision and Language. *arXiv:2103.12975*, 2021.

[48] Liang Huang, Kevin Knight, and Aravind Joshi. A Syntax-Directed Translator with Extended Domain of Locality. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, 2006.

[49] Robin Jia and Percy Liang. Data Recombination for Neural Semantic Parsing. In *Proceedings of ACL*, 2016.

[50] Yong Jiang, Wenjuan Han, and Kewei Tu. Unsupervised neural dependency parsing. In *Proceedings of EMNLP*, 2016.

[51] Lifeng Jin and William Schuler. Grounded PCFG induction with images. In *Proceedings of AACL*. Association for Computational Linguistics, 2020.

[52] Lifeng Jin, Finale Doshi-Velez, Timothy Miller, Lane Schwartz, and William Schuler. Unsupervised Learning of PCFGs with Normalizing Flow. In *Proceedings of ACL*, 2019.

[53] Nal Kalchbrenner and Phil Blunsom. Recurrent Continuous Translation Models. In *Proceedings of EMNLP*, 2013.

[54] Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of EMNLP*, 2020.

[55] Yoon Kim, Chris Dyer, and Alexander M. Rush. Compound Probabilistic Context-Free Grammars for Grammar Induction. In *Proceedings of ACL*, 2019.

[56] Lingpeng Kong, Chris Dyer, and Noah A. Smith. Segmental Recurrent Neural Networks. In *Proceedings of ICLR*, 2016.

[57] Brenden M. Lake. Compositional generalization through meta sequence-to-sequence learning. In *Proceedings of NeurIPS*, 2019.

[58] Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of NeurIPS*, 2018.

[59] Xiang Lisa Li and Alexander M. Rush. Posterior Control of Blackbox Generation. In *Proceedings of ACL*, 2020.

[60] Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. Compositional Generalization for Primitive Substitutions. In *Proceedings of EMNLP*, 2019.

[61] Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. Compositional Generalization by Learning Analytical Expressions. In *Proceedings of NeurIPS*, 2020.

[62] Rune B. Lyngsø and Christian N. S. Pedersen. The consensus string problem and the complexity of comparing hidden markov models. *J. Comput. Syst. Sci.*, 65(3):545–569, 2002.

[63] Yiwei Lyu, Paul Pu Liang, Hai Pham, Eduard Hovy, Barnabás Póczos, Ruslan Salakhutdinov, and Louis-Philippe Morency. StylePTB: A Compositional Benchmark for Fine-grained Controllable Text Style Transfer. In *Proceedings of NAACL*, 2021.

[64] R. Thomas McCoy, Robert Frank, and Tal Linzen. Does Syntax Need to Grow on Trees? Sources of Hierarchical Inductive Bias in Sequence-to-Sequence Networks. *Transactions of the Association for Computational Linguistics*, 8:125–140, 2020.

[65] I. Dan Melamed. Multitext grammars and synchronous parsers. In *Proceedings of NAACL*, 2003.

[66] I. Dan Melamed, Giorgio Satta, and Benjamin Wellington. Generalized Multitext Grammars. In *Proceedings of ACL*, 2004.

[67] Yishu Miao, Edward Grefenstette, and Phil Blunsom. Discovering Discrete Latent Topics with Neural Variational Inference. In *Proceedings of ICML*, 2017.

[68] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of CoNLL*.

[69] Rebecca Nesson, Stuart Shieber, and Alexander Rush. Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *Proceedings of AMTA*, 2006.

[70] Maxwell I. Nye, Armando Solar-Lezama, Joshua B. Tenenbaum, and Brenden M. Lake. Learning Compositional Rules via Neural Program Synthesis. In *Proceedings of NeurIPS*, 2020.

[71] Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. Improving Compositional Generalization in Semantic Parsing. In *Proceedings of EMNLP*, 2020.

[72] Sheena Panthaplackel, Miltiadis Allamanis, and Marc Brockschmidt. Copy that! Editing Sequences by Copying Spans. In *Proceedings of AAAI*, 2021.

[73] Maxim Rabinovich, Mitchell Stern, and Dan Klein. Abstract Syntax Networks for Code Generation and Semantic Parsing. In *Proceedings of ACL*, 2017.

[74] Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. Online and Linear-Time Attention by Enforcing Monotonic Alignments. In *Proceedings of ICML*, 2017.

[75] Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. Weighting finite-state transductions with neural context. In *Proceedings of NAACL*, 2016.

[76] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical Sequence Training for Image Captioning. In *Proceedings of CVPR*, 2017.

[77] Alexander M. Rush. Torch-struct: Deep structured prediction library. In *Proceedings of ACL (System Demonstrations)*, 2020.

[78] Jake Russin, Jason Jo, Randall C. O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv:1904.09708*, 2019.

[79] Abigail See, Peter J. Liu, and Christopher D. Manning. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of ACL*, pages 1073–1083, 2017.

[80] Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. Surprisingly Easy Hard-Attention for Sequence to Sequence Learning. In *Proceedings of EMNLP*, 2018.

[81] Jetic Guand Hassan S. Shavarani and Anoop Sarkar. Top-down Tree Structured Decoding with Syntactic Connections for Neural Machine Translation and Parsing. In *Proceedings of EMNLP*, 2018.

[82] Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. Compositional Generalization and Natural Language Variation: Can a Semantic Parsing Approach Handle Both? *arXiv:2010.12725*, 2020.

[83] Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. Neural data-to-text generation via jointly learning the segmentation and correspondence. In *Proceedings of ACL*, 2020.

[84] Stuart M. Shieber and Yves Schabes. Synchronous Tree-Adjoining Grammars. In *COLING 1990 Volume 3: Papers presented to the 13th International Conference on Computational Linguistics*, 1990.

[85] Khalil Sima'an. Computational Complexity of Probabilistic Disambiguation by means of Tree-Grammars. In *Proceedings of COLING*, 1996.

[86] David Smith and Jason Eisner. Quasi-Synchronous Grammars: Alignment by Soft Projection of Syntactic Dependencies. In *Proceedings on the Workshop on Statistical Machine Translation*, 2006.

[87] David A. Smith and Jason Eisner. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of EMNLP*, Singapore, 2009.

[88] Andreas Stolcke. An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities. *Computational Linguistics*, 21(2), 1995.

[89] Ilya Sutskever, Oriol Vinyals, and Quoc Le. Sequence to Sequence Learning with Neural Networks. In *Proceedings of NeurIPS*, 2014.

[90] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of ACL*, 2015.

[91] Shawn Tan, Yikang Shen, Alessandro Sordoni, Aaron Courville, and Timothy J. O'Donnell. Recursive top-down production for sentence generation with latent trees. In *Findings of the Association for Computational Linguistics: EMNLP*, 2020.

[92] Ke Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. Unsupervised Neural Hidden Markov Models. In *Proceedings of the Workshop on Structured Prediction for NLP*, 2016.

[93] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Proceedings of NeurIPS*, 2017.

[94] Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of EMNLP*, 2007.

[95] Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. Learning Neural Templates for Text Generation. In *Proceedings of EMNLP*, 2018.

[96] Sam Wiseman, Arturs Backurs, and Karl Stratos. Generating (Formulaic) Text by Splicing Together Nearest Neighbors. *arXiv:2101.08248*, 2021.

[97] Yuk Wah Wong and Raymond Mooney. Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus. In *Proceedings of ACL*, 2007.

[98] Kristian Woodsend and Mirella Lapata. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of EMNLP*, 2011.

[99] Kristian Woodsend, Yansong Feng, and Mirella Lapata. Title generation with quasi-synchronous grammar. In *Proceedings of EMNLP*, 2010.

[100] Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, 1997.

[101] Shijie Wu, Pamela Shapiro, and Ryan Cotterell. Hard Non-Monotonic Attention for Character-Level Transduction. In *Proceedings of EMNLP*, 2018.

[102] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of ICML*, 2015.

[103] Kenji Yamada and Kevin Knight. A Syntax-based Statistical Translation Model. In *Proceedings of ACL*, 2001.

[104] Songlin Yang, Yong Jiang, Wenjuan Han, and Kewei Tu. Second-Order Unsupervised Neural Dependency Parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

[105] Songlin Yang, Yanpeng Zhao, and Kewei Tu. PCFGs Can Do Better: Inducing Probabilistic Context-Free Grammars with Many Symbols. In *Proceedings of NAACL*, 2021.

[106] Pengcheng Yin and Graham Neubig. A Syntactic Neural Model for General-Purpose Code Generation. In *Proceedings of ACL*, 2017.

[107] Lei Yu, Jan Buys, and Phil Blunsom. Online Segment to Segment Neural Transduction. In *Proceedings of EMNLP*, 2016.

[108] Songyang Zhang, Linfeng Song, Lifeng Jin, Kun Xu, Dong Yu, and Jiebo Luo. Video-aided Unsupervised Grammar Induction. *arXiv:2104.04369*, 2021.

[109] Yanpeng Zhao and Ivan Titov. Visually grounded compound pcfgs. In *Proceedings of EMNLP*, 2020.

[110] Yanpeng Zhao and Ivan Titov. An Empirical Study of Compound PCFGs. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*. Association for Computational Linguistics, 2021.

[111] Hao Zhu, Yonatan Bisk, and Graham Neubig. The Return of Lexical Dependencies: Neural Lexicalized PCFGs. *Transactions of the Association for Computational Linguistics*, 8:647–661, 2020.

[112] Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Long Short-Term Memory Over Tree Structures. In *Proceedings of ICML*, 2015.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]