Improving Grammar-based Sequence-to-Sequence Modeling with Decomposition and Constraints

Chao Lou, Kewei Tu*
School of Information Science and Technology, ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
{louchao,tukw}@shanghaitech.edu.cn

Abstract

Neural QCFG is a grammar-based sequence-to-sequence (seq2seq) model with strong inductive biases on hierarchical structures. It excels in interpretability and generalization but suffers from expensive inference. In this paper, we study two low-rank variants of Neural QCFG for faster inference with different trade-offs between efficiency and expressiveness. Furthermore, utilizing the symbolic interface provided by the grammar, we introduce two soft constraints over tree hierarchy and source coverage. We experiment with various datasets and find that our models outperform vanilla Neural QCFG in most settings.

1 Introduction

Standard neural seq2seq models are versatile and broadly applicable due to its approach of factoring the output distribution into distributions over the next words based on previously generated words and the input (Sutskever et al., 2014; Gehring et al., 2017; Devlin et al., 2019). Despite showing promise in approximating complex output distributions, these models often fail when it comes to diagnostic tasks involving compositional generalization (Lake and Baroni, 2018; Bahdanau et al., 2019; Loula et al., 2018), possibly attributed to a lack of inductive biases for the hierarchical structures of sequences (e.g., syntactic structures), leading to models overfitting to surface clues.

In contrast to neural seq2seq models, traditional grammar-based models incorporate strong inductive biases to hierarchical structures but suffer from low coverage and the hardness of scaling up (Wong and Mooney, 2006; Bos, 2008). To benefit from both of these approaches, blending traditional methods and neural networks has been studied (Herzig and Berant, 2021; Shaw et al., 2021; Wang et al., 2021, 2022). In particular, Kim (2021) proposes Neural QCFG for seq2seq learning with a quasi-synchronous context-free grammar (QCFG) (Smith and Eisner, 2006) that is parameterized by neural networks. The symbolic nature of Neural QCFG makes it interpretable and easy to impose constraints for stronger inductive bias, which leads to improvements in empirical experiments. However, all these advantages come at the cost of high time complexity and memory requirement, meaning that the model and data size is restricted, which leads to a decrease in text generation performance and limited application scenarios.

In this work, we first study low-rank variants of Neural QCFG for faster inference and lower memory footprint based on tensor rank decomposition (Rabanser et al., 2017), which is inspired by recent work on low-rank structured models (Cohen et al., 2013; Chiu et al., 2021; Yang et al., 2021, 2022). These variants allow us to use more symbols in Neural QCFG, which has been shown to be beneficial for structured latent variable models (Buhai et al., 2020; Chiu and Rush, 2020; Yang et al., 2021, 2022). Specifically, we study two low-rank variants with different trade-off between computation cost and ranges of allowed constraints: the efficient model (E model), following the decomposition method in TN-PCFG (Yang et al., 2021), and the expressive model (P model), newly introduced in this paper. Furthermore, we propose two new constraints for Neural QCFG, including a soft version of the tree hierarchy constraint used by vanilla Neural QCFG, and a coverage constraint which biases models in favour of translating all source tree nodes[1]. We conduct experiments on three datasets and our models outperform vanilla Neural QCFG in most settings. Our code is available at https://github.com/LouChao98/seq2seq_with_qcfg.

*Corresponding Author

[1]Similar topics are discussed in the machine translation literature (Tu et al., 2016; Li et al., 2018, among others).

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 1918–1929

2 Preliminary: Neural QCFG

Let s_1, s_2 be the source and target sequences, and t_1, t_2 be the corresponding constituency parse trees (i.e., sets of labeled spans). Following previous work (Smith and Eisner, 2006; Kim, 2021), we consider QCFG in Chomsky normal form (CNF; Chomsky, 1959) with restricted alignments, which can be denoted as a tuple $G[t_1] = (S, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}[t_1], \theta)$, where S is the start symbol, $\mathcal{N}/\mathcal{P}/\Sigma$ are the sets of nonterminals/preterminals/terminals respectively, $\mathcal{R}[t_1]$ is the set of grammar rules in three forms:

$$\begin{aligned} &S \to A[\alpha_i] \quad \text{where } A \in \mathcal{N},\ \alpha_i \in t_1, \\ &A[\alpha_i] \to B[\alpha_j]C[\alpha_k] \text{ where} \\ &\qquad A \in \mathcal{N},\ B, C \in \mathcal{N} \cup \mathcal{P},\ \alpha_i, \alpha_j, \alpha_k \in t_1, \\ &D[\alpha_i] \to w \quad \text{where } A \in \mathcal{P},\ \alpha_i \in t_1,\ w \in \Sigma, \end{aligned}$$

and θ parameterizes rule probablities $p_\theta(r)$ for each $r \in \mathcal{R}[t_1]$.

Recently, Kim (2021) proposes Neural QCFG for seq2seq learning. He uses a source-side parser to model $p(t_1|s_1)$ and a QCFG to model $p(t_2|t_1)$. The log marginal likelihood of the target sequence s_2 is defined as follows:

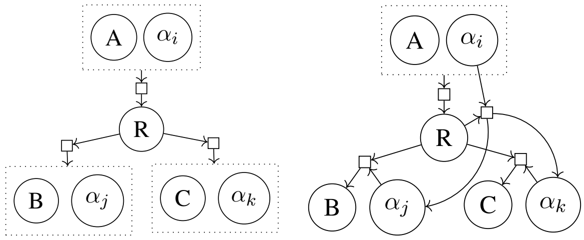
$$\begin{aligned} &\log p_{\theta,\phi}(s_2|s_1) \\ &= \log \sum_{t_1 \in \mathcal{T}(s_1)} p_\theta(s_2|t_1) p_\phi(t_1|s_1) \\ &= \log \sum_{t_1 \in \mathcal{T}(s_1)} \sum_{t_2 \in \mathcal{T}(s_2)} p_\theta(t_2|t_1) p_\phi(t_1|s_1), \end{aligned}$$

where $\mathcal{T}(\cdot)$ denotes the set of possible parse trees for a sequence and θ, ϕ are parameters. Due to the difficulty of marginalizing out t_1 and t_2 simultaneously, Kim (2021) resorts to maximizing the lower bound on the log marginal likelihood,

$$\log p_{\theta,\phi}(s_2|s_1) \geq \mathbb{E}_{t_1 \sim p_\phi(t_1|s_1)} \left[\log p_\theta(s_2|t_1)\right].$$

3 Low-rank Models

Marginalizing t_2 in Neural QCFG has a high time complexity of $O(|\mathcal{N}|(|\mathcal{N}| + |\mathcal{P}|)^2 S^3 T^3)$ where S/T are the source/target sequence lengths. In particular, the number of rules in QCFG contributes to a significant proportion, $O(|\mathcal{N}|(|\mathcal{N}| + |\mathcal{P}|)^2 S^3)$, of the complexity. Below, we try to reduce this complexity by rule decompositions in two ways.



(a) E model (b) P model

Figure 1: Extended factor graph notation of decomposed binary rules (Frey, 2002). Each square □ represents a factor. Arrows indicate conditional probabilities.

3.1 Efficient Model (E Model)

Let $\mathcal{R}$ be a new set of symbols. The E model decomposes binary rules r_b into three parts: $A[\alpha_i] \to R, R \to B[\alpha_j]$ and $R \to C[\alpha_k]$ (Fig. 1a), where $R \in \mathcal{R}$ such that

$$\begin{aligned} p(A[\alpha_i] \to B[\alpha_j]C[\alpha_k]) = \sum_R p(A[\alpha_i] \to R) \\ \times p(R \to B[\alpha_j]) \times p(R \to C[\alpha_k]). \end{aligned}$$

In this way, $|\mathcal{N}|(|\mathcal{N}| + |\mathcal{P}|)^2 S^3$ binary rules are reduced to only $G_E := (3|\mathcal{N}| + 2|\mathcal{P}|)|\mathcal{R}|S$ decomposed rules, resulting in a time complexity of $O(G_E T^3)$[2] for marginalizing t_2. Further, the complexity can be improved to $O(|\mathcal{R}|T^3 + |\mathcal{R}|^2 T^2)$ using rank-space dynamic programming in Yang et al. (2022)[3].

However, constraints that simultaneously involve $\alpha_i, \alpha_j, \alpha_k$ (such as the tree hierarchy constraint in vanilla Neural QCFG and those to be discussed in Sec. 4.1) can no longer be imposed because of two reasons. First, the three nodes are in separate rules and enforcing such constraints would break the separation and consequently undo the reduction of time complexity. Second, the rank-space dynamic programming algorithm prevents us from getting the posterior distribution $p(\alpha_i, \alpha_j, \alpha_k | t_1, s_2)$, which is necessary for many methods of learning with constraints (e.g., Chang et al., 2008; Mann and McCallum, 2007; Ganchev et al., 2010) to work.

[2]Typically, we set $|R| = O(|\mathcal{N}| + |\mathcal{P}|)$.

[3]They describe the algorithm using TN-PCFG (Yang et al., 2021), which decomposes binary rules of PCFG, $A \to BC$, into $A \to R, R \to B$ and $R \to C$. For our case, one can define new symbol sets by coupling nonterminals with source tree nodes: $\mathcal{N}_t = \{(A, \alpha_i)|A \in \mathcal{N}, \alpha_i \in t_1\}$ and $\mathcal{P}_t = \{(A, \alpha_i)|A \in \mathcal{P}, \alpha_i \in t_1\}$. Then our decomposition becomes identical to TN-PCFG and their algorithm can be applied directly.

3.2 Expressive Model (P Model)

In the P model, we reserve the relation among $\alpha_i, \alpha_j, \alpha_k$ and avoid their separation,

$$\begin{aligned} p(A[\alpha_i] \to B[\alpha_j]C[\alpha_k]) = \\ \sum_R p(A[\alpha_i] \to R) \times p(R, \alpha_i \to \alpha_j, \alpha_k) \times \\ p(R, \alpha_j \to B) \times p(R, \alpha_k \to C), \end{aligned}$$

as illustrated in Fig. 1b. The P model is still faster than vanilla Neural QCFG because there are only $G_P := |\mathcal{R}|S^3 + (3|\mathcal{N}| + 2|\mathcal{P}|)|\mathcal{R}|S$ decomposed rules, which is lower than vanilla Neural QCFG but higher than the E model. However, unlike the E model, the P model cannot benefit from rank-space dynamic programming[4] and has a complexity of $O(|\mathcal{R}|S^2T^3 + ((2|\mathcal{N}| + |\mathcal{P}|)|\mathcal{R}|S + |\mathcal{R}|S^3)T^2)$ for marginalizing t_2[5].

Rule $R, \alpha_i \to \alpha_j, \alpha_k$ is an interface for designing constraints involving $\alpha_i, \alpha_j, \alpha_k$. For example, by setting $p(R, \alpha_1 \to \alpha_2, \alpha_3) = 0$ for all $R \in \mathcal{R}$ and certain $\alpha_i, \alpha_j, \alpha_k$, we can prohibit the generation $A[\alpha_1] \to B[\alpha_2]C[\alpha_3]$ in the original QCFG. With this interface, the P model can impose all constraints used by vanilla Neural QCFG as well as more advanced constraints introduced next section.

4 Constraints

4.1 Soft Tree Hierarchy Constraint

Denote the distance between two tree nodes[6] as $d(\alpha_i, \alpha_j)$ and define $d(\alpha_i, \alpha_j) = \infty$ if α_j is not a descendant of α_i. Then, the distance of a binary rule is defined as $d(r) = \max(d(\alpha_i, \alpha_j), d(\alpha_i, \alpha_k))$.

Neural QCFG is equipped with two hard hierarchy constraints. For $A[\alpha_i] \to B[\alpha_j]C[\alpha_k]$, α_j, α_k are forced to be either descendants of α_i (i.e., $d(r) < \infty$), or more strictly, distinct direct children of α_i (i.e., $d(r) = 1$). However, we believe the former constraint is too loose and the latter one is too tight. Instead, we propose a soft constraint based on distances: rules with smaller $d(r)$ are considered more plausible. Specifically, we encode the constraint into a reward function of rules, $\zeta(d(r))$, such that $\zeta(1) > \zeta(2) > \dots$ and $\zeta(a)\zeta(b) > \zeta(c)\zeta(d)$ for $a + b = c + d$ and $\max(a, b) < \max(c, d)$. A natural choice of the reward function is $\zeta(d(r)) := d(r)e^{-d(r)}$. We optimize the expected rewards with a maximum entropy regularizer (Williams and Peng, 1991; Mnih et al., 2016), formulated as follows:

$$\log \sum_{t_2 \in \mathcal{T}(s_2)} p_\theta(t_2|t_1)\zeta(t_2) + \tau \mathbb{H}\left(p_\theta(t_2|t_1, s_2)\right),$$

where $\zeta(t_2) = \prod_{r \in t_2} \zeta(d(r))$[7], $p_\theta(t_2|t_1, s_2) = p_\theta(t_2|t_1) / \sum_{t \in \mathcal{T}(s_2)} p_\theta(t|t_1)$, $\mathbb{H}$ represents entropy, and τ is a positive scalar.

4.2 Coverage Constraint

Our experiments on vanilla neural QCFG show that inferred alignments could be heavily imbalanced: some source tree nodes are aligned with multiple target tree nodes, while others are never aligned. This motivates us to limit the number of alignments per source tree node with an upper bound[8], u. Because the total number of alignments is fixed to $|t_2|$, this would distribute alignments from popular source tree nodes to unpopular ones, leading to more balanced source coverage of alignments. We impose this constraint via optimizing the posterior regularization likelihood (Ganchev et al., 2010),

$$\mathbb{E}_{t_1}\left(\log p_\theta(s_2|t_1) + \gamma \min_{q \in \mathcal{Q}} \mathbb{KL}(q(t_2)||p_\theta(t_2|t_1, s_2))\right),$$

where $\mathbb{KL}$ is the Kullback-Leibler divergence (KL), γ is a positive scalar and $\mathcal{Q}$ is the constraint set $\{q(t_2)|\mathbb{E}_{q(t)}\phi(t) \le \xi\}$, i.e., expectation of feature vector ϕ over any distribution in $\mathcal{Q}$ is bounded by constant vector ξ. We define the target tree feature vector $\phi(t_2) \in \mathbb{N}^{|t_1|}$ such that $\phi_i(t_2)$ represents the count of source tree node α_i being aligned by nodes in t_2 and $\xi = u\mathbf{1}$. Ganchev et al. (2010) provide an efficient algorithm for finding the optimum q, which we briefly review in Appx. C. After finding q, the KL term of two tree distributions, q and p_θ, can be efficiently computed using the Torch-Struct library (Rush, 2020).

[4] Below is an intuitive explanation. Assume there is only one nonterminal symbol. Then we can remove A, B, C because they are constants. The decomposition can be simplified to $\alpha_i \to R, R\alpha_i \to \alpha_j\alpha_k$, which is equivalent to $\alpha_i \to \alpha_j\alpha_k$, an undecomposed binary rule. The concept "rank-space" is undefined in an undecomposed PCFG.

[5] It is better than $O(G_P T^3)$ because we can cache some intermediate steps, as demonstrated in Cohen et al. (2013); Yang et al. (2021). Details can be found in Appx. A.

[6] The distance between two tree nodes is the number of edges in the shortest path from one node to another.

[7] $r \in t_2$ means the rule at each generation step of t_2.

[8] We do not set lower bounds, meaning each source tree node should be aligned at least n times, because our source-side parser uses a grammar in CNF, and such trees could contain semantically meaningless nodes, which are not worthing to be aligned. For example, trees of *Harry James Potter* must contain either *Harry James* or *James Potter*.

Approach	Simple	Jump	A. Right	Length
vNQ^1	96.9	96.8	98.7	95.7
E_{Model}	9.01	-	1.2	-
P_{Model}	95.27	97.08	97.63	91.72

Table 1: Accuracy on the SCAN datasets. vNQ^1 is vanilla Neural QCFG from Kim (2021). vNQ^1 and P_{Model} use the hard constraint $d(r) < \infty$.

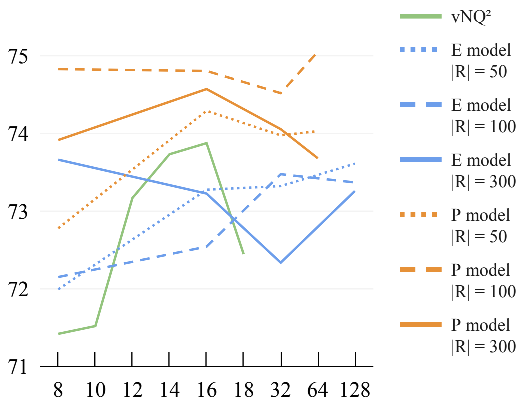


Figure 2: BLEU-4 scores on the ATP task. No constraint is placed. The horizontal axis represents $|\mathcal{N}|(= |\mathcal{P}|)$.

5 Experiments

We conduct experiments on the three datasets used in Kim (2021). Details can be found in Appx. D.1.

5.1 SCAN

We first evaluate our models on four splits of the SCAN dataset (Lake and Baroni, 2018). We report accuracy in Tab. 1. The P model equipped with constraints can achieve almost perfect performance similar to vanilla Neural QCFG, while the E model fails due to a lack of constraints.

5.2 Style Transfer and En-Fr Translation

Next, we evaluate the models on the three hard transfer tasks from the StylePTB dataset (Lyu et al., 2021) and a small-scale En-Fr machine translation dataset (Lake and Baroni, 2018). Tab. 2 shows results of the models with different constraints[9]. Low-rank models generally achieve comparable or better performance and consume much less memory[10]. We can also find that the soft tree hierarchy constraint outperforms hard constraints and is very helpful when it comes to extremely small data (i.e., AEM and VEM). The coverage constraint also improves performance in most cases.

Approach	*nil*	$+\text{H}^1$	$+\text{H}^2$	+S	+C
Active to passive (ATP)					
vNQ^1	–	66.2	–	–	–
vNQ^2	71.42	71.56	–	71.62	73.86
E_{Model}	73.48	×	×	×	74.25
P_{Model}	75.06	69.88	–	73.11	75.44
Adjective Emphasise (AEM)					
vNQ^1	–	31.6	–	–	–
vNQ^2	28.82	31.52	–	36.77	30.81
E_{Model}	28.33	×	×	×	28.67
P_{Model}	31.81	29.14	–	35.91	30.12
Verb Emphasise (VEM)					
vNQ^1	–	31.9	–	–	–
vNQ^2	26.09	29.64	–	30.50	28.50
E_{Model}	25.21	×	×	×	24.67
P_{Model}	27.43	24.77	–	26.81	30.66
En-Fr machine translation					
vNQ^1	–	–	26.8	–	–
vNQ^2	28.63	–	29.10	30.45	31.87
E_{Model}	28.93	×	×	×	29.33
P_{Model}	29.27	–	29.76	30.51	29.69

Table 2: BLEU-4 for tasks from the StylePTB dataset (the top three series) and BLEU for Fr-En machine translation against different models and constraints. vNQ^2 is our reimplementation of Kim (2021). *nil* means that no constraint is placed. H^1 and H^2 is the hard constraint $d(r) < \infty$ and $d(r) = 1$, respectively. S is the soft tree hierarchy constraint. C is the coverage constraint. $\times$ means that the constraint is inapplicable and $-$ means we do not run the experiment or Kim (2021) does not report the score.

5.3 Analysis

We study how the number of nonterminals affects performance. On our computer[11], we can use at most 18/64/128 nonterminals in vanilla Neural QCFG/the P model/the E model, showing that our low-rank models are more memory-friendly than vanilla Neural QCFG. We report results in Fig. 2. There is an overall trend of improved performance with more nonterminals (with some notable exceptions). When the numbers of nonterminals are

[9]Following Kim (2021), we calculate the metrics for tasks from the StylePTB dataset using the nlg-eval library (Sharma et al. (2017); `https://github.com/Maluuba/nlg-eval`) and calculate BLEU for En-Fr MT using the multi-bleu script (Koehn et al. (2007); `https://github.com/moses-smt/mosesdecoder`).

[10]We report speed and memory usage briefly in Sec 5.4 and in detail in Appx. D.3.

[11]One NVIDIA TITIAN RTX with 24 GB memory.

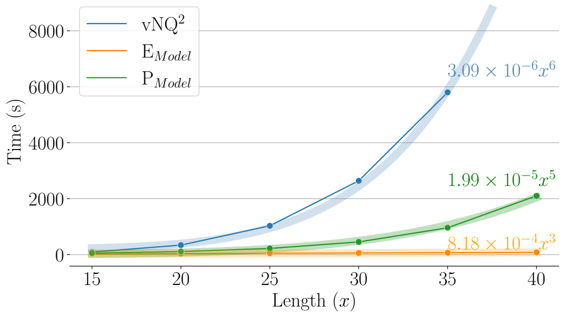


Figure 3: The duration required to train one epoch on synthetic datasets with different length ($x = S = T$). Thick and shallow lines are fitted curves based on time complexities of vNQ^2, E$_{Model}$ and P$_{Model}$, i.e., $O(x^6), O(x^3)$ and $O(x^5)$.

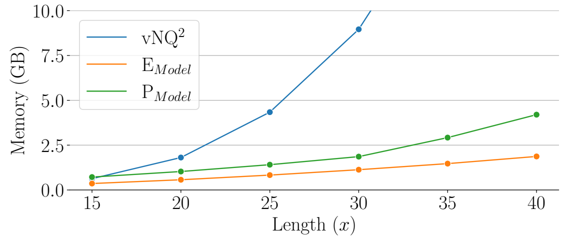


Figure 4: Memory usage for training with batch size 1 on synthetic datasets with different length ($x = S = T$).

the same, the P model outperforms vanilla Neural QCFG consistently, showing its superior parameter efficiency. In contrast, the E model is defeated by vanilla QCFG and the P model in many cases, showing the potential harm of separating $\alpha_i, \alpha_j, \alpha_k$.

5.4 Speed Comparison

We benchmark speed and memory usage using synthetic datasets with different sequence lengths. Fig. 3 and 4 illustrate the results. Compared to the standard Neural QCFG, the E model and P model are significantly faster and have a lower memory footprint. This enables them to model longer sequences effectively. For data construction and more results, please refer to Appx. D.3.

6 Conclusion

We have presented two low-rank variants of Neural QCFG based on decomposition for efficiency and two new constraints over tree hierarchy and source coverage. Experiments on three datasets validate the effectiveness and efficiency of our proposed models and constraints.

7 Limitations

First, unlike decoders in neural seq2seq models, which can attend to any previously generated tokens, QCFGs have a strong context-free independence assumption during generation. With this assumption, Neural QCFG cannot model some complex distributions. A potential solution is to use stronger grammars, such as RNNG (Dyer et al., 2016) and Transformer Grammars (TG; Sartran et al., 2022).

Second, we assume that both the grammars used by the source-side parser and QCFG are in CNF. Although it is convenient for discussion and implementation, CNF does not suit for modeling the structure of practical sequences. In semantic representations (e.g., Abstract Meaning Representation (Banarescu et al., 2013)), a predicate could have more than two arguments. Ideally, we should represent n-ary predicates with n-ary rules. However, for grammars in CNF, $n - 1$ unnatural binary rules are required to represent n-ary predicates. In natural language, we will face semantically meaningless spans due to CNF, which is discussed in Sec 4.2.

Third, although using decomposition improves the speed and the memory requirement, our low-rank models still cost much more computation resources than neural seq2seq models for two main reasons. (1) A large amount of nonterminal symbols increase the memory cost significantly. (2) Because finding the most probable string t_2 from $p_\theta(t_2|t_1)$ is NP-hard (Sima'an, 1996; Lyngsø and Pedersen, 2002), we follow Kim (2021) to use a decoding strategy with heavy sampling. For real data, we may need to sample hundreds or thousands of sequences and then rank them, which can be much slower than the decoding of neural seq2seq models.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work was supported by the National Natural Science Foundation of China (61976139).

References

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. 2019. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, pages 277–286. College Publications.

Rares-Darius Buhai, Yoni Halpern, Yoon Kim, Andrej Risteski, and David Sontag. 2020. Empirical study of the benefits of overparameterization in learning latent variable models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1211–1219. PMLR.

Ming-Wei Chang, Lev Ratinov, Nicholas Rizzolo, and Dan Roth. 2008. Learning and inference with constraints. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, page 1513–1518. AAAI Press.

Justin Chiu, Yuntian Deng, and Alexander Rush. 2021. Low-rank constraints for fast inference in structured models. *Advances in Neural Information Processing Systems*, 34:2887–2898.

Justin Chiu and Alexander Rush. 2020. Scaling hidden Markov language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1341–1349, Online. Association for Computational Linguistics.

Noam Chomsky. 1959. On certain formal properties of grammars. *Information and Control*, 2(2):137–167.

Shay B. Cohen, Giorgio Satta, and Michael Collins. 2013. Approximate PCFG parsing using tensor decomposition. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 487–496, Atlanta, Georgia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*.

Brendan J. Frey. 2002. Extending factor graphs so as to unify directed and undirected graphical models. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03, page 257–264, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *J. Mach. Learn. Res.*, 11:2001–2049.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 908–921, Online. Association for Computational Linguistics.

Yoon Kim. 2021. Sequence-to-sequence learning with latent neural grammars. In *Advances in Neural Information Processing Systems*, volume 34, pages 26302–26317. Curran Associates, Inc.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Yanyang Li, Tong Xiao, Yinqiao Li, Qiang Wang, Changming Xu, and Jingbo Zhu. 2018. A simple and effective approach to coverage-aware neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 292–297, Melbourne, Australia. Association for Computational Linguistics.

João Loula, Marco Baroni, and Brenden Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114, Brussels, Belgium. Association for Computational Linguistics.

Rune B. Lyngsø and Christian N.S. Pedersen. 2002. The consensus string problem and the complexity of comparing hidden markov models. *Journal of Computer and System Sciences*, 65(3):545–569. Special Issue on Computational Biology 2002.

Yiwei Lyu, Paul Pu Liang, Hai Pham, Eduard Hovy, Barnabás Póczos, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2021. StylePTB: A compositional benchmark for fine-grained controllable text style transfer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2116–2138, Online. Association for Computational Linguistics.

Gideon S. Mann and Andrew McCallum. 2007. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 593–600, New York, NY, USA. Association for Computing Machinery.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1928–1937. JMLR.org.

Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. 2017. Introduction to tensor decompositions and their applications in machine learning.

Alexander Rush. 2020. Torch-struct: Deep structured prediction library. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 335–342, Online. Association for Computational Linguistics.

Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Miloš Stanojević, Phil Blunsom, and Chris Dyer. 2022. Transformer Grammars: Augmenting Transformer Language Models with Syntactic Inductive Biases at Scale. *Transactions of the Association for Computational Linguistics*, 10:1423–1439.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 922–938, Online. Association for Computational Linguistics.

Khalil Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

David Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 23–30, New York City. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany. Association for Computational Linguistics.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021. Structured reordering for modeling latent alignments in sequence transduction. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Bailin Wang, Ivan Titov, Jacob Andreas, and Yoon Kim. 2022. Hierarchical phrase-based sequence-to-sequence learning. *arXiv preprint arXiv:2211.07906*.

Ronald J. Williams and Jing Peng. 1991. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA. Association for Computational Linguistics.

Songlin Yang, Wei Liu, and Kewei Tu. 2022. Dynamic programming in rank space: Scaling structured inference with low-rank HMMs and PCFGs. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4797–4809, Seattle, United States. Association for Computational Linguistics.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021. PCFGs can do better: Inducing probabilistic context-free grammars with many symbols. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1487–1498, Online. Association for Computational Linguistics.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1604–1612, Lille, France. PMLR.

A Time Complexity of P Model

Let $\beta_{ij}, \beta_{jk} \in \mathbb{R}^{|\mathcal{N}|\times|t_1|}$ be two cells in the chart of the dynamic programming. $\beta_{ij}(x, y)$ denotes indexing into the matrix. Denote $A[\alpha_1] \to B[\alpha_2]C[\alpha_3]$ as r_b. The state transition equation is

$$\beta_{ik}(A, \alpha_1) = \sum_{\substack{j,B,C\\ \alpha_2,\alpha_3}} p(r_b)\beta_{ij}(B, \alpha_2)\beta_{jk}(C, \alpha_3).$$

Let's define following terms:

$$\tilde{\beta}_{ij}(R, \alpha_2) = \sum_B p(R, \alpha_2 \to B)\beta_{ij}(B, \alpha_2)$$
$$\tilde{\beta}_{jk}(R, \alpha_3) = \sum_C p(R, \alpha_3 \to C)\beta_{ij}(C, \alpha_3)$$
$$\hat{p} = p(A[\alpha_1] \to R)p(R, \alpha_1 \to \alpha_2, \alpha_3)$$

Then the state transition equation can be reformulated as:

$$\beta_{ik}(A, \alpha_1) = \sum_{R,\alpha_2,\alpha_3} \hat{p} \underbrace{\sum_j \tilde{\beta}_{ij}(R, \alpha_2)\tilde{\beta}_{jk}(R, \alpha_3)}_{\hat{\beta}_{ik}},$$

where $\hat{\beta}_{ij} \in \mathbb{R}^{|\mathcal{R}|\times|t_1|\times|t_1|}$. We can compute $\tilde{\beta}_{ij}$ in $O((|\mathcal{N}| + |\mathcal{P}|)|\mathcal{R}|S)$ and cache it for composing $\hat{\beta}_{ij}$. Then $\hat{\beta}_{ik}$ can be computed in $O(|\mathcal{R}|S^2T)$. Finally, we can compute β_{ik} in $O(|\mathcal{R}|S^3 + |\mathcal{N}||\mathcal{R}|S)$ by sum out α_2, α_3 first:

$$\beta_{ik}(A, \alpha_1) = \sum_R p(A[\alpha_1] \to R) \sum_{\alpha_2,\alpha_3} p(R, \alpha_1 \to \alpha_2, \alpha_3)\hat{\beta}_{ik}$$

So, summing terms of all the above steps and counting the iteration over i, k, we will get $O(|\mathcal{R}|S^2T^3 + ((2|\mathcal{N}| + |\mathcal{P}|)|\mathcal{R}|S + |\mathcal{R}|S^3)T^2)$.

B Neural Parameterization

We mainly follow (Kim, 2021) to parameterize the new decomposed rules. First, we add embeddings of terms on the same side together. For example, we do two additions $e_{lhs} = e_R + e_{\alpha_i}$ and $e_{rhs} = e_{\alpha_j} + e_{\alpha_k}$ for $R, \alpha_i \to \alpha_j, \alpha_k$, where e_x denotes the embedding of x. Note that we use the same feed-forward layer f as (Kim, 2021) to obtain e_x from some feature h_x. i.e. $e_x = f(h_x)$. Then, we compute the inner products of embeddings obtained in the previous step as unnormalized scores. For example, $p(R, \alpha_i \to \alpha_j, \alpha_k) \propto \exp(e_{lhs}^\top e_{rhs})$.

C Posterior Regularization

The problem $\min_{q\in\mathcal{Q}} \mathbb{KL}(q(t_2)||p(t_2|t_1, s_2))$ has the optimal solution

$$q^* = \frac{1}{Z(\lambda^*)}p(t_2|t_1, s_2)\exp\{-\lambda^*\phi(t_2)\},$$

where

$$Z(\lambda^*) = \sum_{t_2} p(t_2|s_1, t_1)\exp\{-\lambda^*\phi(t_2)\}$$

and λ^* is the solution of the dual problem:

$$\max_{\lambda\geq 0} -b\cdot\lambda - \log Z(\lambda)$$

We can reuse the inside algorithm to compute $Z(\lambda^*)$ efficiently because our $\phi(t)$ can be factored as $p(t_2|t_1, s_2)$:

$$p(t_2|t_1, s_2) = \prod_{r\in t_2} p_\theta(r)$$
$$\phi(t) = \sum_{r\in t_2} \phi(r, t_1),$$

where $\phi(r, t_1) = 1$ if t_1 is in the left-hand side of r and $\phi(r, t_1) = 0$ otherwise. Then, the solution q^* can be written as

$$q^*(t_2) \propto \prod_{r\in t_2} p_\theta(r)\exp\{-\lambda\phi(r, t_1)\}.$$

Recall that we define $\phi(t)$ to be the counts of source nodes being aligned by nodes in t. We can factor $\phi(t)$ in terms of r because each target tree non-leaf node invokes exactly one rule and only occurs on the left-hand side of that rule. So, the sum over r is equivalent to the sum over target tree nodes.

D Experiments

D.1 Experimental Details

We implement vNQ[2], the E model, and the P model using our own codebase. We inherit almost all hyperparameters of Kim (2021) and a basic constraint: the target tree leaves/non-leaf nodes can only be aligned to source tree leaves/non-leaf nodes, and especially, the target tree root can only be aligned to the source tree root. One major difference is that, in our experiments, we do not use early-stopping and run fixed optimization steps, which are much more than the value set in Kim (2021) (i.e., 15). It is because in preliminary experiments[12], we found that the task metric (e.g., BLEU) almost always get improved consistently with the process of training, while the lowest perplexity occurs typically at an early stage (which is the criteria of early-stopping in Kim (2021)), and computing task metric is very expensive for Neural QCFGs. We report metrics on test sets averaged over three runs on all datasets except for SCAN. As mentioned in the code of Kim (2021), we need to run several times to achieve good performance on SCAN. Therefore, we report the maximum accuracy in twenty runs.

SCAN (Lake and Baroni, 2018) is a diagnostic dataset containing translations from English commands to machine actions. We conduct experiments on four splits: We evaluate our models on four splits of the SCAN (Lake and Baroni, 2018) dataset: *simple*, *add primitive (jump)*, *add template (around right)* and *length*. The latter three splits are designed for evaluating compositional generalization. Following (Kim, 2021), we set $|\mathcal{N}| = 10, |\mathcal{P}| = 1$.

StylePTB (Lyu et al., 2021) is a text style transfer dataset built based on Penn Treebank (PTB; Marcus et al., 1993). Following Kim (2021), we conduct experiments on three hard transfer tasks: textitactive to passive (2808 examples), *adjective emphasis* (696 examples) and *verb emphasis* (1201 examples). According to Tab. 2, we set $|\mathcal{N}| = |\mathcal{P}| = 32, |\mathcal{R}| = 100$ for the E model and set $|\mathcal{N}| = |\mathcal{P}| = 64, |\mathcal{R}| = 100$ for the P model.

En-Fr MT (Lake and Baroni, 2018) is a small-scale machine translation dataset. We use the same split as Kim (2021). The size of training/validate/test set is 6073/631/583. We set $|\mathcal{N}| = |\mathcal{P}| = 32, |\mathcal{R}| = 100$ for the E model and $|\mathcal{N}| = |\mathcal{P}| = 32, |\mathcal{R}| = 196$ for the P model.

[12]We run 100 epochs and evaluate task metrics on validation sets every 5 epochs.

D.2 Tune Hyperparameter

We tune hyperparameters according to metrics on validation sets, either manually or with the Bayesian Optimization and Hyperband (BOHB) search algorithm (Falkner et al., 2018) built in the `wandb` library. First, we tune $|\mathcal{N}|$, $|\mathcal{P}|$, $|\mathcal{R}|$ and the learning rate of parameters for parameterizing QCFG. We freeze hyperparameters related to the source-side parser, the contextual encoder (i.e., LSTM), and the TreeLSTM (Tai et al., 2015; Zhu et al., 2015). For the ATP task from StylePTB, we run the grid search to plot Fig. 2 and choose the best hyperparameters. For other tasks, we run about 20 trials according to BOHB for each manually set search range. Typically, the size of a search range is 256 (four choices for each tunable hyperparameter). Next, we tune the strength of the coverage constraint for all models by running with $\gamma = 0.5, 1, 2$.

D.3 Speed and Memory Usage Comparison

Tab. 3 shows the time and memory usage on synthetic datasets. Each synthetic dataset contains 1000 pairs of random sequences with the same length sampled from a vocabulary with size 5000, i.e., $\{(s_1, s_2)_1, \dots (s_1, s_2)_{1000}\}, s_1, s_2 \in \Sigma^v, |\Sigma| = 5000$ where v is the length. We set $|\mathcal{N}| = |\mathcal{P}| = 8$ for vanilla Neural QCFG and $|\mathcal{N}| = |\mathcal{N}| = 50, |\mathcal{R}| = 200$ for others. We train models on a computer with an NVIDIA GeForce RTX3090. Note that we disable the copy mechanism in Kim (2021) because of its complicated effects on memory usage, such that the results differ from Fig. 2 (in which models enable the copy mechanism).

v	Approach	Constraint	Batch size	Time (s)	GPU Memory (GB)
10	vNQ^2	*nil*	8	25.6	1.42
		$+H^1$	8	25.5	1.43
		$+H^2$	8	113.8	7.67
		+S	8	60.5	2.46
		+C	8	132.7	3.08
	E_{Model}	*nil*	8	20.1	1.59
		+C	8	40.4	1.59
	P_{Model}	*nil*	8	30.7	3.78
		$+H^1$	8	31.3	3.79
		$+H^2$	8	64.0	6.41
		+S	8	45.8	4.08
		+C	8	73.9	4.02
20	vNQ^2	*nil*	8	341.2	14.49
		$+H^1$	8	342.4	14.60
		$+H^2$	1	≈16539.4	14.13
		+S	2	≈1734.4	8.93
		+C	2	≈4657.1	12.24
	E_{Model}	*nil*	8	40.0	4.58
		+C	4	173.4	14.48
	P_{Model}	*nil*	8	111.3	8.25
		$+H^1$	8	110.8	8.29
		$+H^2$	4	452.3	9.83
		+S	8	269.8	18.76
		+C	4	643.5	18.20
40	vNQ^2		1	×	×
	E_{Model}	*nil*	8	82.5	14.95
		+C	8	177.0	14.95
	P_{Model}	*nil*	4	≈2102.7	16.78
		$+H^1$	4	≈2097.6	16.96
		$+H^2$	1	×	×
		+S	2	≈2729.3	10.63
		+C	1	×	×

Table 3: Time and memory usage on synthetic datasets. We report statistics with as large as possbile batch size (in $1, 2, 4, 8$). $\times$ represents that we get an out-of-memory error even if we set batch size to 1. $\approx$ represents that the value is estimated using a small portion of the dataset.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
7

☒ A2. Did you discuss any potential risks of your work?
We cannot see any potential risk.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract; 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

5

☑ B1. Did you cite the creators of artifacts you used?
5

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
All of them have been well-known and publicly available for a long time.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
5

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
They have been well-studied. We follow previous work to conduct experiments on them.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Appx. D.1

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appx. D.1

C ☑ Did you run computational experiments?

5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
5; D

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
D

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
D.1

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
5

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.