Hierarchical Phrase-based Sequence-to-Sequence Learning

Bailin Wang
MIT
bailinw@mit.edu

Ivan Titov
University of Edinburgh
University of Amsterdam
ititov@inf.ed.ac.uk

Jacob Andreas
MIT
jda@mit.edu

Yoon Kim
MIT
yoonkim@mit.edu

Abstract

We describe a neural transducer that maintains the flexibility of standard sequence-to-sequence (seq2seq) models while incorporating hierarchical phrases as a source of inductive bias during training and as explicit constraints during inference. Our approach trains two models: a discriminative parser based on a bracketing transduction grammar whose derivation tree hierarchically aligns source and target phrases, and a neural seq2seq model that learns to translate the aligned phrases one-by-one. We use the same seq2seq model to translate at all phrase scales, which results in two inference modes: one mode in which the parser is discarded and only the seq2seq component is used at the sequence-level, and another in which the parser is combined with the seq2seq model. Decoding in the latter mode is done with the cube-pruned CKY algorithm, which is more involved but can make use of new translation rules during inference. We formalize our model as a source-conditioned synchronous grammar and develop an efficient variational inference algorithm for training. When applied on top of both randomly initialized and pretrained seq2seq models, we find that both inference modes performs well compared to baselines on small scale machine translation benchmarks.

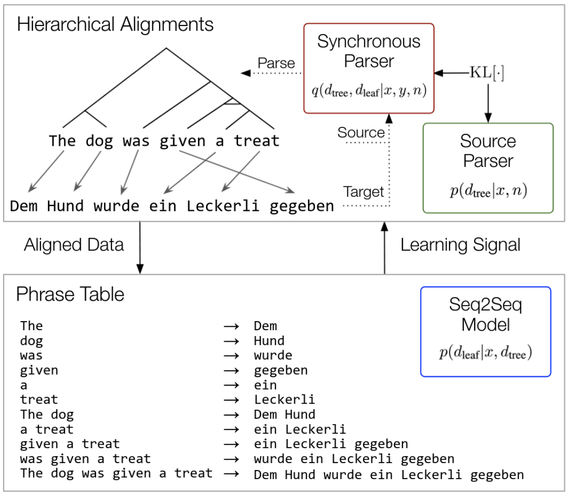

Figure 1: An overview of our approach. During training a variational synchronous parser hierarchically segments and aligns source-target phrases (top), which provides training pairs for a seq2seq model (bottom). The seq2seq loss provides learning signal to the synchronous parser which subsequently provides signal to (and is regularized by) a monolingual source parser. After training, there are two inference models: one mode in which both parsers are discarded, which results in a regular seq2seq model that has been regularized by latent hierarchical phrase pairs; another mode in which the seq2seq model can be combined with the source parser to derive a neural synchronous grammar. Inference in this mode is done via translation-as-parsing with cube-pruned CKY.

1 Introduction

Despite the improvements in performance and data-efficiency enabled by recent advances in pre-training, standard neural sequence-to-sequence (seq2seq) models can still fail to model the hierarchical structure of sentences and be brittle with respect to to novel syntactic structures (Lake and Baroni, 2018; Kim and Linzen, 2020; Weißenhorn et al., 2022). It has moreover not been clear how to incorporate explicit constraints such as translation rules (e.g., for translating idioms) into these black-box models without changing the underlying architecture. Classic grammar- and automaton-based approaches are well-suited for capturing hierarchical structure and can readily incorporate new rules, but have trouble with examples that do not conform exactly to the symbolic specifications. This paper describes a neural transducer that maintains the flexibility of neural seq2seq models but still models hierarchical phrase alignments between source and target sentences, which have been shown to be useful for transduction tasks (Chiang, 2005; Wong and Mooney, 2007, *inter alia*). This transducer bottoms out in an ordinary neural seq2seq model, and can thus take advantage of pre-training schemes like BART (Lewis et al., 2020; Liu et al., 2020) and T5 (Raffel et al., 2020; Xue et al., 2021).

The transducer consists of two components learned end-to-end: a discriminative parser based on a bracketing transduction grammar (BTG; Wu, 1997) whose derivation tree hierarchically aligns

Code available at: https://github.com/berlino/btg-seq2seq.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 8211–8229
December 7-11, 2022 ©2022 Association for Computational Linguistics

source and target phrases, and a neural seq2seq model that learns to translate the aligned phrases one-by-one. Importantly, a single seq2seq model is trained to translate phrases at *all scales*, including at the sentence-level, which results in two inference modes. In the first mode, we discard the parser and simply decode with the resulting seq2seq model at the sequence level, which maintains fast inference but still makes use of a seq2seq model that has been exposed to (and regularized by) latent hierarchical phrase alignments. This approach can be seen a form of *learned* data augmentation (Akyürek et al., 2021), wherein a model is guided away from wrong generalizations by additionally being trained on crafted data that coarsely express the desired inductive biases. In the second mode, we view the combined parser and seq2seq model as a source-conditioned neural synchronous grammar to derive a set of synchronous grammar rules along with their scores (a "neural phrase table"), and decode with a modified version of the classic cube-pruned CKY algorithm (Huang and Chiang, 2005). While more involved, this decoding scheme can incorporate explicit constraints and new translation rules during inference.

We formulate our approach as a latent variable model and use variational learning to efficiently maximize a lower bound on the log marginal likelihood. This results in an intuitive training scheme wherein the seq2seq component is trained on phrase tables derived from hierarchical alignments sampled from a variational posterior synchronous parser, as shown in Figure 1. When applied on top of both randomly initialized and pretrained seq2seq models across various small-scale machine translation benchmarks, we find that both modes improve upon baseline seq2seq models.

2 Approach

Notation. We use x, y to denote the source/target strings, $x_{i:k}, y_{a:c}$ to denote the source/target spans, and x_i to denote the token with index i.

2.1 A Seq2Seq-Based Synchronous Grammar

The inversion transduction grammar (ITG), introduced by Wu (1997), defines a synchronous grammar that hierarchically generates source and target strings in tandem. The bracketing transduction grammar (BTG) is a simplified version of ITG with only one nonterminal, with the primary goal of modeling alignments between source and target words. We propose to use a variant of BTG which defines a stochastic monolingual grammar over target strings conditioned on source strings and can use a neural seq2seq model to generate phrases.

As a motivating example, consider the following English-Chinese example from Shao et al. (2018):[1]

他们 在 桥上 谈笑 风生 →

They were laughing it up on the bridge

A translator for this sentence is faced with several problems: first, they must identify the appropriate unit of translation (number of phrases); two, they must reorder phrases if necessary; finally, they must be able translate phrase-by-phrase, taking into account phenomena such as idioms whose translations cannot be obtained by stitching together translations of subparts (e.g., 谈笑 风生 → *laughing it up*), while also making sure the final translation is fluent as a whole. We model this process with a *source-conditioned* synchronous grammar whose target derivation probabilistically reifies the above process. Informally, this grammar first samples the number of segments to generate on the target (n), then segments and reorders source phrases (d_{tree}), and finally translates phrase-by-phrase while still conditioning on the full source sentence for fluent translations (d_{leaf}). This is shown in Figure 2.

More formally, this monolingual grammar over the target can be represented by a tuple

$$G[x] = \left(Root, \{T^n, S^n, I^n\}_{n=1}^{|x|}, C, \Sigma, \mathcal{R}[x]\right),$$

where $Root$ is the distinguished start symbol, $\{T^n, S^n, I^n\}_{n=1}^{|x|}$ are nonterminals where n denotes the number of segments associated with the nonterminal (to be explained below), $|x|$ is the source sentence length, C is a special nonterminal that emits phrases, and Σ is the target alphabet. The rule set $\mathcal{R}[x]$ is given by a fixed set of context-free production rules. For the root node we have $\forall n \in \{1, 2 \ldots |x|\}$,

$$Root \to T^n, \quad T^n \to S^n[x], \quad T^n \to I^n[x],$$

where S, I represent the Straight and Inverted nonterminals. Binary nonterminal rules are given by,

$$\forall i, j, k \in \{0, 1 \ldots |x|\} \qquad \text{s.t. } i < j < k$$
$$S^n[x_{i:k}] \to I^l[x_{i:j}]S^r[x_{j:k}],$$
$$S^n[x_{i:k}] \to I^l[x_{i:j}]I^r[x_{j:k}],$$
$$I^n[x_{i:k}] \to S^l[x_{j:k}]I^r[x_{i:j}],$$
$$I^n[x_{i:k}] \to S^l[x_{j:k}]S^r[x_{i:j}],$$

where each nonterminal is indexed by (i.e., aligned

[1]The phrase-level translations are: 他们 → *They*, 在 桥上 → *were on the bridge*, 谈笑 风生 → *laughing it up*.

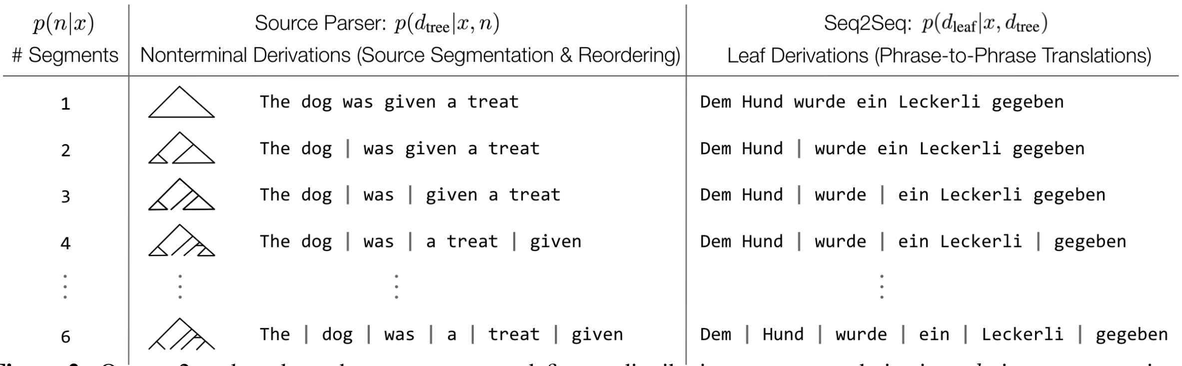

Figure 2: Our seq2seq-based synchronous grammar defines a distribution over target derivations d given source string x by decomposing $p(d|x)$ into three components, $p(d|x) = p(n|x)p(d_{\text{tree}}|x, n)p(d_{\text{leaf}}|x, d_{\text{tree}})$. We first sample the number of segments in the target n according to $p(n|x)$. Then we sample the tree topology $d_{\text{tree}} \sim p(d_{\text{tree}}|x, n)$, which gives a segmentation and reordering of the source sequence. Finally, the leaf derivations are given by translating the segmented and reordered source phrases one-by-one with a seq2seq model $p(d_{\text{leaf}}|x, d_{\text{tree}})$, where the phrase-by-phrase translations are contextualized against the entire source sentence. We show some example trees d_{tree} and leaf derivations d_{leaf} for various values of n. Note that each n could have multiple segmentations and reorderings.

to) a source span $x_{i:k}$, and the superscript n refers to the number of segments to generate.[2] We force $l + r = n$ to ensure that the number of segments generated by the $l$eft and $r$ight nonterminal is equal to n. Finally, if S^1 or I^1 is generated (i.e., there is only one segment to be generated), we move to the leaf nonterminal C and generate target phrases,

$$S^1[x_{i:k}] \to C[x_{i:k}], \qquad I^1[x_{i:k}] \to C[x_{i:k}],$$
$$C[x_{i:k}] \to v, \qquad v \in \Sigma^+.$$

We use a probabilistic version of the above grammar where each rule is weighted via $\pi : \mathcal{R} \to \mathbb{R}_{\geq 0}$. Importantly, for the leaf nonterminal we use a (potentially pretrained) neural seq2seq model, i.e.,

$$\pi(C[x_{i:k}] \to v) = p_{\text{seq2seq}}(v|x_{i:k}).$$

Foreshadowing, if we force $Root \to T^1$ in the beginning of derivation then this corresponds to the usual seq2seq approach which generates the full target sequence conditioned on the source sequence.

2.2 Parameterization

By assigning scores to rules conditioned on x with function π, we can obtain a distribution $p(d|x)$ over all possible derivations d licensed by $G[x]$. Note that d is fully characterized by: (1) n, the total number of target segments, (2) d_{tree}, the set of derivation rules excluding the leaf rules (i.e., rules with $C[x_{i:k}]$ on the left hand side), and (3) d_{leaf}, the set of leaf derivation rules. We hence decompose the derivation into three distributions, $p(d|x) = p(n|x)p(d_{\text{tree}}|x, n)p(d_{\text{leaf}}|x, d_{\text{tree}})$. Intuitively, d_{tree} encodes the segmentations and reorderings of x, while d_{leaf} encodes the phrase translations which implicitly segment y.

[2]Rules of the form $S \to SI$ and $I \to IS$ are disallowed. This resolves ambiguities where certain reorderings can be encoded by multiple derivations.

$p(n|x)$. For the distribution over the number of target segments we use a geometric distribution,

$$p(n|x) = \lambda(1 - \lambda)^{n-1},$$

where $0 < \lambda < 1$, $1 \leq n < |x|$. This sets the probability of the upper bound $n = |x|$ to be $(1 - \lambda)^{|x|-1}$. This distribution is implemented by the following configuration of the grammar:

$$\pi(Root \to T^n) = p(n|x).$$

The geometric distribution favors fewer target segments, which aligns with one of our goals which is to be able to use the resulting model as a regular seq2seq system (i.e., forcing $Root \to T^1$).

$p(d_{\text{tree}}|x, n)$. For the distribution over source segmentations and reorderings, we use a discriminative CRF parser whose scores are given by neural features over spans. The score for rule $R \in \mathcal{R}$, e.g., $S^n[x_{i:k}] \to I^l[x_{i:j}]A^r[x_{j:k}]$, is given by,

$$\pi(R) = \exp\left(e_A^\top f(\tilde{\mathbf{h}}_{i:j}, \tilde{\mathbf{h}}_{j:k})\right),$$

where $A \in \{S, I\}$, $\tilde{\mathbf{h}}_{i:j}$ and $\tilde{\mathbf{h}}_{j:k}$ are the span features derived from the contextualized word representations of tokens at i, j, k from a Transformer encoder as in (Kitaev and Klein, 2018), f is an MLP that takes into two span features, and e_A is a one-hot vector that extracts the corresponding scalar score. (Hence we use the same score all n). These scores are globally normalized, i.e.,

$$p(d_{\text{tree}}|x, n) \propto \prod_{R \in d_{\text{tree}}} \pi(R),$$

to obtain a distribution over d_{tree}, where R are the rules in d_{tree}. Algorithms 1, 2 in appendix C give the dynamic programs for normalization/sampling.

$p(d_{\text{leaf}}|x, d_{\text{tree}})$. Finally, for the distribution over the leaf translations, the phrase transduction rule probabilities $p(C[x_{i:k}] \rightarrow v)$ are parameterized with a (potentially pretrained) neural seq2seq model. For the seq2seq component, instead of just conditioning on $x_{i:k}$ to parameterize $p_{\text{seq2seq}}(v|x_{i:k})$, in practice we use the contextualized representation of $x_{i:k}$ from a Transformer encoder that conditions on the full source sentence. Hence, our approach can be seen learning *contextualized* phrase-to-phrase translation rules. Specifically, for a length-m target phrase we have,

$$p(d_{\text{leaf}}|x, d_{\text{tree}}) = \prod_{R \in d_{\text{leaf}}} \pi(R)$$

$$\pi(C[x_{i:k}] \rightarrow v) = \prod_{t=1}^{m} p_{\text{seq2seq}}(v_t \mid v_{<t}, \mathbf{h}_{i:k}),$$

$$\mathbf{h}_{i:k} = \text{Encoder}(x)_{i:k},$$

where $\mathbf{h}_{i:k}$ refers to the contextualized representation of $x_{i:k}$ that is obtained after passing it through a Transformer encoder. The decoder attends over $\mathbf{h}_{i:k}$ and $v_{<t}$ to produce a distribution over v_t. The two "exit" rules are assigned a score $\pi(A^1[x_{i:k}] \rightarrow C[x_{i:k}]) = 1$ where $A \in \{S, I\}$.

Remark. This grammar is a mixture of both locally and globally normalized models. In particular, $p(n|x)$ and $p(d_{\text{leaf}}|x, d_{\text{tree}})$ are locally normalized as the scores for these rules are already probabilities. In contrast, $p(d_{\text{tree}}|x, n)$ is obtained by globally normalizing the score of d_{tree} with respect to a normalization constant $Z_{\text{tree}}(x, n)$.[3] In the appendix we show how to convert the unnormalized scores π in the globally normalized model into normalized probabilities $\tilde{\pi}$ (Algorithm 1), which will be used for CKY decoding in Section 2.4.

We also share the parameters between components whenever possible. For example, the encoder of the seq2seq component is the same encoder that is used to derive span features for d_{tree}. The decoder of the seq2seq component is used as part of the variational distribution when deriving span features for the variational parser. As we will primarily be working with small datasets, such parameter sharing will provide implicit regularization to each of the components of our grammar and militate against overfitting. We provide the full parameterization of each component in appendix A.

2.3 Learning

Our model defines a distribution over target trees $d = (n, d_{\text{tree}}, d_{\text{leaf}})$ (and by marginalization, target strings y) conditioned on a source string x,

$$p(y|x) = \sum_{d \in D(x,y)} p(d|x)$$

$$= \sum_{d \in D(x,y)} p(n|x)p(d_{\text{tree}}|x, n)p(d_{\text{leaf}}|x, d_{\text{tree}}),$$

where $D(x, y)$ is the set of trees such that $\text{yield}(D(x, y)) = (x, y)$. While this marginalization can theoretically be done in $O(L^7)$ time where $L = \min(|x|, |y|)$,[4] we found it impractical in practice. We instead to use variational approximations to the posteriors of n, d_{tree}, d_{leaf} and optimize a lower bound on the log marginal likelihood whose gradient estimator can be obtained in $O(L^3)$ time. Each of the three variational distributions q is analogous to the three distributions p introduced previously, but additionally conditions on y.

$\boldsymbol{q(n|x, y)}$. Similar to $p(n|x)$, we use a geometric distribution to model $q(n|x, y)$. The only difference is that we have $1 \leq n \leq \min(|x|, |y|)$.

$\boldsymbol{q(d_{\text{tree}}|x, y, n)}$. This is parameterized with neural span scores over x in the same way as in $p(d_{\text{tree}}|x, n)$, except that the span features also condition on the target y. That is, $\tilde{\mathbf{h}}_{i:j}$, $\tilde{\mathbf{h}}_{j:k}$ are now the difference features of the *decoder*'s contextualized representations over x from a Transformer encoder-decoder, where y is given to the encoder.[5] The dynamic programs for normalization/sampling in this CRF is the same as in $p(d_{\text{tree}}|x, n)$.

$\boldsymbol{q(d_{\text{leaf}}|x, y, d_{\text{tree}})}$. Observe that conditioned on (x, y, d_{tree}), the only source of uncertainty in d_{leaf} is $C[x_{i:k}] \rightarrow y_{a:c}$ for spans $y_{a:c}$. That is, $q(d_{\text{leaf}}|x, y, d_{\text{tree}})$ is effectively a hierarchical segmentation model with a fixed topology given by d_{tree}. The segmentation model can be described by a probabilistic parser over the following grammar,

$$G[y, d_{\text{tree}}] = (\mathit{Root}, \{T^s\}_{s=1}^n, \mathcal{R}[y, d_{\text{tree}}]),$$

where rule set $\mathcal{R}[y, d_{\text{tree}}]$ includes (here $m = l + r$),

$$\mathit{Root} \rightarrow T^n[y], \; T^m[y_{a:c}] \rightarrow T^l[y_{a:b}]T^r[y_{b:c}]$$

where the second rule exists only when the rule like $A^m \rightarrow B^l C^r$ is in d_{tree}. We use span features over

[3] Locally normalized parameterization of d_{tree} would require π to take into account the number of segments n (and l, r), which would lead to a more complex rule-scoring function. In our parameterization $\pi(R)$ does not depend on the number of segments, and n, l, r only affect $p(d_{\text{tree}}|x, n)$ via the normalization $Z_{\text{tree}}(x, n)$.

[4] For each $n \in [L]$, we must run the $O(n^6)$ bitext inside algorithm, so the total runtime is $\sum_{n=1}^{L} O(n^6) = O(L^7)$.

[5] We use this parameterization (instead of, e.g., a bidirectional Transformer encoder over the concatenation of x and y) in order to take advantage of pretrained multilingual encoder-decoder models such as mBART.

y for parameterization similarly to $p(d_{\text{tree}}|x, n)$. Algorithms 3, 4 in appendix C give the normalization/sampling dynamic programs for this CRF.

Optimization. With the variational distributions in hand, we are now ready to give the following evidence lower bound (ignoring constant terms),

$$\log p(y|x) \geq \mathbb{E}_{q(n|x,y)}\Big\{\mathbb{E}_{q(d_{\text{tree}}|x,y,n)}\Big[\mathbb{E}_{q(d_{\text{leaf}}|x,y,d_{\text{tree}})}[\log p(d_{\text{leaf}}|x, d_{\text{tree}})] + \mathbb{H}[q(d_{\text{leaf}}|x, y, d_{\text{tree}})]\Big] - \mathrm{KL}[q(d_{\text{tree}}|x, y, n) \,\|\, p(d_{\text{tree}}|x, n)]\Big\}.$$

(The KL between $q(n|x, y)$ and $p(n|x)$ is a constant and hence omitted.) See appendix B for the derivation. While the above objective seems involved, in practice it results in an intuitive training scheme. Let θ be the grammar parameters and ϕ be the variational parameters.[6] Training proceeds as follows:

1. Sample number of target segments,
$$n \sim q(n|x, y).$$
2. Sample trees from variational parsers,
$$d'_{\text{tree}} \sim q(d_{\text{tree}}|x, y, n),$$
$$d'_{\text{leaf}} \sim q(d_{\text{leaf}}|x, y, d'_{\text{tree}}),$$
and obtain the set of hierarchical phrase alignments (i.e., phrase table) A implied by $(d'_{\text{tree}}, d'_{\text{leaf}})$.
3. Train seq2seq model by backpropagating the leaf likelihood $\log p(d'_{\text{leaf}}|x, d'_{\text{tree}})$ to the seq2seq model,
$$\sum_{(x_{i:k}, y_{a:c}) \in A} \nabla_\theta \log p_{\text{seq2seq}}(y_{a:c}|x_{i:k}).$$
4. Train variational parsers by backpropagating the score function and KL objectives with respect to $q(d_{\text{tree}}|x, y, n)$,
$$\big(\log p(d'_{\text{leaf}}|x, d'_{\text{tree}}) + \mathbb{H}[q(d_{d_{\text{leaf}}}|x, y, d'_{\text{tree}})]\big) \times \nabla_\phi \log q(d'_{\text{tree}}|x, y, n) - \nabla_\phi \mathrm{KL}[\cdot],$$
and also with respect to $q(d_{\text{leaf}}|x, y, n)$,
$$\log p(d'_{\text{leaf}}|x, d'_{\text{tree}}) \times \nabla_\phi \log q(d'_{\text{leaf}}|x, y, d'_{\text{tree}}) + \nabla_\phi \mathbb{H}[q(d_{\text{leaf}}|x, y, d'_{\text{tree}})].$$
5. Train source parser by backpropagating $-\nabla_\theta \mathrm{KL}[\cdot]$ to $p(d_{\text{tree}}|x, n)$.

For the score function gradient estimators we also employ a self-critical control-variate (Rennie et al., 2017) with argmax trees.The KL and entropy metrics are calculated exactly with the inside algorithm with the appropriate semirings (Li and Eisner, 2009). This training scheme is shown in Figure 1. See Algorithms 5, 6 in appendix C for the entropy-/KL dynamic programs.

Complexity. The above training scheme reduces time complexity from $O(L^7)$ to $O(L^3)$ since we do not marginalize over all possible number of segments and instead sample n, and further decompose the derivation tree into $(d_{\text{tree}}, d_{\text{leaf}})$ and sample from CRF parsers, which takes $O(L^3)$. The KL/entropy calculations are also $O(L^3)$.

2.4 Inference: Two Decoding Modes

During inference, we can use the seq2seq to either directly decode at the sentence-level (i.e., setting $n = 1$ and choosing $Root \to T^1$), or generate phrase-level translations and compose them together. The first decoding strategy maintains the efficiency of seq2seq, and variational learning can be seen as a structured training algorithm that regularizes the overly flexible seq2seq model with latent phrase alignments. The second decoding strategy takes into account translations at all scales and can further incorporate constraints, such as new translation rules during decoding.

Seq2Seq decoding. Setting $n = 1$, decoding in $G[x]$ this case is reduced to the standard sentence-level seq2seq decoding for the terminal rules $C[x_{0,|x|}] \to v$ with beam search. We call this variant of the model **BTG-**1 **Seq2Seq**.

CKY decoding. Here we aim to find y with maximal marginal probability $\operatorname{argmax}_y \sum_{d \in T(x,y)} p(d|x)$. This requires exploring all possible derivations licensed by $G[x]$. We employ CKY decoding algorithm based on the following recursion:

$$\mathbb{C}_{i:k}[v_1 v_2, S^n] = \sum_{l=1, r=n-l}^{n-1} \Big\{ \mathbb{C}_{i:j}[v_1, I^l] \cdot \mathbb{C}_{j:k}[v_2, S^r] \cdot \tilde{\pi}(S^n \to I^l S^r) + \mathbb{C}_{i:j}[v_1, I^l] \cdot \mathbb{C}_{j:k}[v_2, I^r] \cdot \tilde{\pi}(S^n \to I^l I^r) \Big\}$$

$$\mathbb{C}_{i:k}[v_1 v_2, I^n] = \sum_{l=1, r=n-l}^{n-1} \Big\{ \mathbb{C}_{j:k}[v_1, S^l] \cdot \mathbb{C}_{i:j}[v_2, I^r] \cdot \tilde{\pi}(I^n \to S^l I^r) + \mathbb{C}_{j:k}[v_1, S^l] \cdot \mathbb{C}_{i:j}[v_2, S^r] \cdot \tilde{\pi}(I^n \to S^l S^r) \Big\}$$

[6]We share the Transformer encoder/decoder between $p_\theta(\cdot)$ and $q_\phi(\cdot)$, and therefore the only parameters that are different between θ and ϕ are the MLPs' parameters for scoring rules in the CRF parsers.

where $\mathbb{C}_{i:k}[v, A]$ stores the score of generating target phrase v based on source phrase $x_{i:k}$ with nonterminal A. In the above we abbreviate $\tilde{\pi}(S^n \to I^l S^r)$ to mean $\tilde{\pi}(S^n[x_{i:k}] \to I^l[x_{i:j}] S^r[x_{j:k}])$ since i, j, k are implied by the $\mathbb{C}$ terms that this quantity is multiplied with. Here v, v_1, v_2 denotes target phrases, and $v_1 v_2$ refers to the concatenation of two phrases. The chart is initialized by phrase pairs that are generated based on seq2seq,

$$\mathbb{C}_{i:k}[v, S^1] = p_{\text{seq2seq}}(v|x_{i:k}) \cdot \pi(S^1 \to C),$$
$$\mathbb{C}_{i:k}[v, I^1] = p_{\text{seq2seq}}(v|x_{i:k}) \cdot \pi(I^1 \to C)$$

The final prediction is based on the chart item for the root node computed via:

$$\mathbb{C}[v, T^n] = \mathbb{C}_{0,|x|}[v, S^n] \cdot \tilde{\pi}(T^n \to S^n) + \mathbb{C}_{0,|x|}[v, I^n] \cdot \tilde{\pi}(T^n \to I^n)$$
$$\mathbb{C}[v, Root] = \sum_{n=1}^{|x|} \mathbb{C}[v, T^n] \cdot \pi(Root \to T^n).$$

Cube pruning. Exact marginalization is still intractable since there are theoretically infinitely many phrases that can be generated by a seq2seq model (e.g., there is $\mathbb{C}_{i:k}[v, S^1]$ for each possible phrase $v \in \Sigma^+$). We follow the cube pruning strategy from Huang and Chiang (2005) and only store the top-K target phrases within each chart item, and discard the rest. During the bottom-up construction of the chart, for $A \in \{S, I\}$, $\mathbb{C}_{i:k}[v, A^1]$ only stores the K phrases generated by beam search with size K, and $\mathbb{C}_{i:k}[v, A^n]$ constructs at most $2 \cdot (n-1) \cdot K \cdot K$ target phrase translations and we only keep the top-K phrases. We call this decoding scheme **BTG-N SeqSeq**. We generally found it to perform better than BTG-1 Seq2Seq but incur additional cost in decoding.[7]

External Translation Rules. An important feature of CKY decoding is that we can constrain the prediction of BTG-Seq2Seq to incorporate new translation rules during inference. When we have access to new translation rules $\langle x_{i:k} \to v \rangle$ during inference (e.g., for idioms or proper nouns), we can enforce this by adding a rule $C[x_{i:k}] \to v$ and setting the score $\pi(C[x_{i:k}] \to v) = 1$ (or some high value) during CKY decoding.

[7]Merging two sorted lists requires $O(K \log K)$ if implemented by priority queue (Huang and Chiang, 2005). Compared to the standard CKY algorithm, we also need to enumerate the number of segments n. Hence, the best time complexity of CKY decoding is $O\left((K \log K)|x|^4\right)$. In practice, we set a maximum number of segments N and treat it as a hyperparameter to tune.

3 Experimental Setup

We apply BTG-Seq2Seq to a spectrum sequence-to-sequence learning tasks, starting from diagnostic toy datasets and then gradually moving towards real machine translation tasks. Additional details (dataset statistics, model sizes, etc.) are given in appendix E.

Baselines. We use the standard Transformer-based seq2seq models (Vaswani et al., 2017) as the backbone module for all experiments. We consider three baselines: (1) a seq2seq model trained in a standard way, (2) a seq2seq model pretrained with phrase pairs extracted with an off-the-shelf tool, Fast-Align (Dyer et al., 2013),[8] and (3) a trivial baseline in which the model is pretrained on phrase pairs generated by randomly splitting a source-target sentence pair into phrase pairs. The latter two approaches study the importance of using learned alignments.

BTG-1 Seq2Seq and BTG-N Seq2Seq. We focus on evaluating the two decoding modes of our model: CKY decoding where phrase-level translations are structurally combined to form a sentence-level prediction (BTG-N Seq2Seq), and standard "flat" Seq2Seq decoding where sentence-level translations are generated directly by forcing $Root \to T^1$ (BTG-1 Seq2Seq). In this way we test the effectiveness of hierarchical alignments both as a structured translation model and as a regularizer on standard Seq2Seq learning. BTG-N Seq2Seq is sometimes referred simply as BTG-Seq2Seq in cases where CKY decoding is activated by default (e.g., constrained inference).

Backbone Transformers. In each experiment, we use the same set of hyperparameters across models for fair comparison. One crucial hyperparameter for the backbone Transformers of BTG-Seq2Seq is to use relative positional encodings (as opposed to absolute ones) which are better at handling translation at phrase levels.Hence we use relative positional encodings whenever possible except when utilizing pretrained Transformers that have been pretrained with absolute positional embeddings.

4 Main Results

4.1 Toy SVO-SOV Translation

We start with a diagnostic translation task in which a model has to transform a synthetic sentence from

[8]We extract phrase pairs that respect consistency constraints (Koehn, 2009) from Fast-Align word-level alignments.

Task	SVO→SOV		EN→ZH			DE→EN			CHR↔EN	
Model	Novel-P	Few-Shot	VP	NP	PP	500	1000	4978	CHR→EN	EN→CHR
Seq2Seq	1.2	22.1	49.0	45.2	41.5	22.9	25.7	29.7	9.1	7.8
+ Trivial-Align	-	-	34.8	35.5	32.9	17.8	19.1	23.0	7.9	6.3
+ Fast-Align	-	-	42.0	**64.4**	37.6	21.1	24.0	29.1	10.3	7.5
BTG-1 Seq2Seq	**100**	96.0	55.9	60.8	48.5	24.7	26.8	29.6	11.2	9.3
BTG-N Seq2Seq	**100**	**100**	**57.2**	61.3	**49.8**	**25.1**	**29.0**	**31.0**	**13.6**	**11.0**

Table 1: Accuracies on the toy SVO-SOV task, and BLEU scores on the few-shot English-Chinese (EN→ZH) translation, low-data German-English translation (DE→EN), and the low-resource Cherokee-English (CHR↔EN) translation tasks. Trivial-Align and Fast-Align refer to the baseline seq2seq models that are pre-trained with phrase pairs produced by the simplistic uniform-splitting heuristic and fast-align, respectively. For ZH→EN, the columns (VP, NP, PP) denote different few-shot splits for pairing phrases with new contexts. For DE→EN, the columns (500, 1000, 4978) refer to the number of training pairs. We use a pretrained mBART to initialize the seq2seq component for the DE→EN experiments.

SVO to SOV structure. The dataset is generated with the following synchronous grammar,

$$\begin{aligned}&\mathit{Root} \rightarrow \langle \text{S V O},\ \text{S O V}\rangle,\\ &\text{S} \rightarrow \text{NP},\ \ \text{O} \rightarrow \text{NP},\ \ \text{V} \rightarrow \text{VP},\\ &\text{NP} \rightarrow \langle np_s,\ np_t\rangle,\ \ \text{VP} \rightarrow \langle vp_s,\ vp_t\rangle,\end{aligned}$$

where the noun phrase pairs $\langle np_s, np_t\rangle$ and the verb phrase pairs $\langle vp_s,\ vp_t\rangle$ are uniformly sampled from two nonoverlapping sets of synthetic phrase pairs with varying phrase lengths between 1 to 8. We create two generalization train-test splits.

Novel-Position. For this split, a subset of np_s, np_t phrase pairs are only observed in one syntactic position during training (i.e., only as a subject or only as an object). At test time, they appear in a new syntactic role. Each remaining np_s, np_t pairs appears in both positions in training.

Few-Shot. In this setting, each subject (or object) phrase is observed in three examples in three different examples. During the evaluation, each subject phrase appears with context words (i.e. a predicate and an object phrase) which are found in the training set but not with this subject phrase.

Results. In Table 1, we observe that BTG-N Seq2Seq perfectly solves these two tasks. We also observed that BTG-N Seq2Seq correctly recovers the latent trees when $N = 3$. Interestingly, even without structured CKY decoding, BTG-1 Seq2Seq implicitly captures the underlying transduction rules, achieving near-perfect accuracy. In contrast, standard seq2seq models fail in both settings.

4.2 English-Chinese: Phrasal Few-shot MT

Next, we test our models on the English-Chinese translation dataset from Li et al. (2021), which was designed to test for compositional generalization in MT. We use a subset of the original dataset and create three few-shot splits similar to the few-shot setting in the previous task. For example in the NP split noun phrases are paired with new contexts during evaluation. Compared with the few-shot setting for the SVO-SOV task, this setup is more challenging due to a larger vocabulary and fewer training examples (approx. 800 data points).

Results. As shown in Table 1, both BTG-1 Seq2Seq and BTG-N Seq2Seq improve on the standard Seq2Seq by a significant margin. BTG-N Seq2Seq also outperforms the Fast-Align baseline[9] on VP and PP splits and slightly lags behind on NP, highlighting the benefits of jointly learning alignment along with the Seq2Seq model.

Analysis. In Figure 3 we visualize the induced phrasal alignments for an example translation by extracting the most probable derivation $\text{argmax}_d\, p(d|x)$ for $n = 3$, which encodes the phrase-level correspondence between x and y. In this example "at a restaurant" is segmented and reordered before "busy night" according to the Chinese word order. We also give example translations in Table 2, which shows two failure modes of the standard seq2seq approach. The first failure mode is producing hallucinations. In the first example, the source phrase "hang out with the empty car he liked" is translated to a Chinese phrase with a completely different meaning. This Chinese translation exactly matches a Chinese training example, likely triggered by the shared word 空车 ("empty car"). The second failure mode is under-translation. In the second example, standard seqsSeq completely ignores the last phrase "it was so dark" in its Chinese translation, likely because this phrase never co-occurs with the current context. In contrast BTG-Seq2Seq is able to cover all source phrases, and

[9]We pretrain on Fast-Align phrases and fine-tune on the original training set. This was found to perform better than jointly training on the combined dataset.

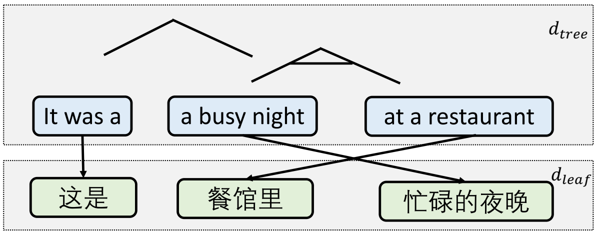

Figure 3: Induced hierarchical alignment by BTG-Seq2Seq on an English-Chinese example.

Source:	she knew this but continued to *hang out with the empty car he liked* .
Seq2Seq:	她知道这一点，但还是继续他喜欢的空车都散发出一股难闻的气味。
BTG Seq2Seq:	她知道这一点，但继续他喜欢的空车在一起 。
Reference:	她知道这一点，但继续和他喜欢的空车在一起。
Source:	he went inside the airplane on the floor *and it was so dark* .
Seq2Seq:	他进了地板上的飞机里。
BTG Seq2Seq:	他走进地板上的飞机里面一片漆黑 。
Reference:	他走进地板上的飞机里面，里面一片漆黑。

Table 2: Chinese translations from a baseline seq2seq model and our BTG-Seq2Seq approach. Source phrases that are translated incorrectly by standard seq2seq are highlighted in italics, and target translations are delineated with dashed underlines.

in this case, predicts the corresponding Chinese phrase for "it was so dark".

4.3 German-English: Low-Data MT

Next, we move onto more a realistic dataset and consider a low data German-English translation task , following Sennrich and Zhang (2019). The original dataset contains TED talks from the IWSLT 2014 DE-EN translation task (Cettolo et al., 2014). Instead of training the seq2seq component from scratch, we use pre-trained mBART (Liu et al., 2020) to initialize the seq2seq component of our grammar (and the baseline seq2seq). This was found to perform much better than random initialization, and highlights an important feature of our approach which can work with arbitrary neural seq2seq models. (We explore this capability further in section 4.5.) The use of pre-trained mBART lets us consider extremely low-data settings, with 500 and 1000 sentence pairs, in addition to the 4978 setup from the original paper. We observe in Table 1 that even the standard seq2seq achieves surprisingly reasonable BLEU scores in this low-data setting—much higher that scores of seq2seq models trained from scratch (Sennrich and Zhang, 2019).[10] BTG-N Seq2Seq and BTG-1 Seq2Seq again outperform the baseline seq2seq model.

Pretraining on Fast-Align harms performance, potentially due to it's being unable to induce meaningful phrase pairs in such low data regimes. In contrast, all our aligners (i.e., parsers) can also leverage pretrained features from mBART (we use

[10]Their best seq2seq model with 4978 sentences obtains 16.57 BLEU, while phrase-based SMT obtains 15.87.

Input:	Er hat das Unternehmen *von Grund auf* aufgebaut
New translation rule:	von Grund auf → from scratch
Original Prediction:	He built the company from the ground up
New Prediction:	He built the company from scratch
Reference:	He built the company from scratch
Google Translate:	He built the company from the ground up
Input:	Die europäische Krise *schließt den Kreis*
New translation rule:	schließt den Kreis → is coming full circle
Original Prediction:	The euro crisis closes the loop
New Prediction:	The European crisis is coming full circle
Reference:	The European crisis is coming full circle
Google Translate:	The European crisis closes the circle
Input:	Die ARD strahlte gestern um 20:15 "Aus dem Nichts" aus
New translation rule:	"Aus dem Nichts" → "In the Fade", 20:15 → 8.15 pm
Original Prediction:	The ARD aired yesterday at 20:15 "Out of Nowhere"
New Prediction:	The ARD aired yesterday 8.15 pm "In the Fade"
Reference:	ARD broadcast "In the Fade" yesterday at 8.15 pm
Google Translate:	Yesterday at 8:15 p.m., ARD broadcast "Out of Nowhere".

Table 3: Experiments on injecting new translation rules during inference. Examples are with the mBART model finetuned on the IWSLT 2014 German-to-English dataset. We manually provide new translation rules to BTG-SeqSeq, and decode with the constrained CKY algorithm. Injected phrases in the new predictions are highlighted with dashed underlines. We also provide results from Google Translate.

the same mBART model to give span scores for all our parsers), which can potentially improve structure induction and have a regularizing effect.

Injecting New Translation Rules. We investigate whether we can control the prediction of BTG-Seq2Seq by incorporating new translation rules during CKY inference. In particular, we give new translation rules to mBART-based BTG-Seq2Seq for DE-EN translation. Since the model has been trained only on 4978 pairs, the translations are far from perfect. But as shown in Table 3 we observe that BTG-Seq2Seq can translate proper nouns, time expression, and even idioms to an extent. The idiom case is particularly interesting as pure seq2seq systems are known to be *too* compositional when translating idioms (Dankers et al., 2022).

4.4 Cherokee-English: Low-Resource MT

Our next experiments consider a true low-resource translation task with Cherokee-English, using the dataset from Zhang et al. (2020). This dataset contains around 12k/1k/1k examples for train/dev/test, respectively. Unlike German-English, Cherokee was not part of the mBART pretraining set, and thus we train all our models from scratch. In Table 1 we see that BTG-N Seq2Seq and BTG-1 Seq2Seq again outperform the standard seq2seq.[11]

4.5 Low Resource MT with Pretrained Models

The current trend in low resource MT is to first pretrain a single multilingual model on a large number

[11]However standard SMT is a strong baseline for Cherokee-English: Zhang et al. (2020) obtain 14.5 (Chr→En) and 9.8 (En→Chr) with Moses (Koehn et al., 2007).

Models	Mongolian (MN→EN)	Marathi (MR→EN)	Azerbaijani (AZ→EN)	Bengali (BN→EN)
Finetune mT5	8.6	8.5	10.2	8.4
BTG-1 mT5	10.7	9.6	11.3	9.6
BTG-N mT5	9.4	9.5	11.5	10.3
Finetune mBART50	11.2	**11.6**	15.5	**13.6**
BTG-1 mBART50	11.5	11.2	15.8	12.8
BTG-N mBART50	**12.1**	10.9	**15.9**	12.8

Table 4: BLEU scores on four low-resource machine translation tasks. The backbone seq2seq models of our BTG models are intialized with either mT5 (Xue et al., 2021) or mBART50 (Tang et al., 2020). Numbers in grey background are taken from Tang et al. (2020).

of language pairs (Tang et al., 2020; Fan et al., 2021; Costa-jussà et al., 2022) and then optionally finetune it on specific language pairs. In our final experiment, we explore whether we can improve upon models that have already pretrained on bitext data. We focus mBART50-large which has been pretrained on massive monolingual *and* bilingual corpora. We randomly pick four low-resource languages from the ML50 benchmark (Tang et al., 2020), each of which has fewer than $10K$ bitext pairs. We also experiment with mT5-base (Xue et al., 2021) to see if we can plug-and-play other pretrained seq2seq models.

From Table 4, we see that our BTG models outperform the standard finetuning baseline in four languages for mT5, and achieve slight improvement in two languages for mBART50. We conjecture that this is potentially due to (1) mBART50's being already trained on a large amount of bitext data and (2) mT5's use of relative positional encodings, which is more natural for translating at all phrase scales. Finally, although we focus on pretrained seq2seq models in this work, an interesting extension would be to consider *prompt*-based hierarchical phrase-based MT where $p_{\text{seq2seq}}(v|x_{i:k})$ is replaced with a large language model that has been appropriately prompted to translate phrases.[12]

5 Related Work

Synchronous grammars Classic synchronous grammars and transducers have been widely explored in NLP for many applications (Shieber and Schabes, 1990; Wu, 1997; Eisner, 2003; Ding and Palmer, 2005; Nesson et al., 2006; Huang et al., 2006; Wong and Mooney, 2007; Wang et al., 2007; Graehl et al., 2008; Blunsom et al., 2008, 2009; Cohn and Lapata, 2009, *inter alia*). In this work, we focus on the formalism of bracketing transduction grammars, which have been used for modeling reorderings in SMT systems (Nakagawa, 2015; Neubig et al., 2012). Recent work has explored the coupling bracketing transduction grammars with with neural parameterization as a way to inject structural biases into neural sequence models (Wang et al., 2021). Our work is closely related to the neural QCFG (Kim, 2021), a neural parameterization of a quasi-synchronous grammars (Smith and Eisner, 2006), which also defines a monolingual grammar conditioned on the source. However they do not experiment with embedding a neural seq2seq model within their grammar. More generally our approach extends the recent line of work on neural parameterizations of classic grammars (Jiang et al., 2016; Han et al., 2017, 2019; Kim et al., 2019; Jin et al., 2019; Zhu et al., 2020; Yang et al., 2021b,a; Zhao and Titov, 2020, *inter alia*), although unlike in these works we focus on the transduction setting.

Data Augmentation Our work is also related to the line of work on utilizing grammatical or alignment structures to guide flexible neural seq2seq models via data augmentation (Jia and Liang, 2016; Fadaee et al., 2017; Andreas, 2020; Akyürek et al., 2021; Shi et al., 2021; Yang et al., 2022; Qiu et al., 2022) or auxiliary supervision (Cohn et al., 2016; Mi et al., 2016; Liu et al., 2016; Yin et al., 2021). In contrast to these works our data augmentation module has stronger inductive biases for hierarchical structure due to explicit use of latent tree-based alignments.

6 Conclusion

We presented a neural transducer that maintains the flexibility and seq2seq models but still incorporates hierarchical phrases both as a source of inductive bias during training and as explicit constraints during inference. We formalized our model as a synchronous grammar and developed an efficient variational inference algorithm for training. Our model performs well compared to baselines on small MT benchmarks both as a regularized seq2seq model and as a synchronous grammar.

[12]The source parser distribution could be set to uniform or learned on a small amount of labeled data.

7 Limitations

Our work has several limitations. While variational inference enables more efficient training than full marginalization ($O(L^3)$ vs. $O(L^7)$), it is still much more expensive compute- and memory-wise than regular seq2seq training. This currently precludes our approach as a viable alternative on large-scale machine translation benchmarks, though advances in GPU-optimized dynamic programming algorithms could improve efficiency (Rush, 2020). (However, as we show in our experiments it is possible to finetune a pretrained seq2seq MT system using our approach.) Cube-pruned CKY decoding is also much more expensive than regular seq2seq decoding. In general, grammar- and automaton-based models enable a greater degree of model introspection and interpretability. However our use of black-box seq2seq systems within the grammar still makes our approach not as interpretable as classic transducers.

Acknowledgements

This study was supported by MIT-IBM Watson AI Lab.

References

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

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 200–208, Columbus, Ohio. Association for Computational Linguistics.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2009. Bayesian Synchronous Grammar Induction. In *Proceedings of NeurIPS*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 2–17, Lake Tahoe, California.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan. Association for Computational Linguistics.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 876–885, San Diego, California. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2009. Sentence Compression as Tree Transduction. *Journal of Artificial Intelligence Research*, 34(1):637–674.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Verna Dankers, Christopher G Lucas, and Ivan Titov. 2022. Can transformer be too compositional? analysing idiom processing in neural machine translation. *arXiv preprint arXiv:2205.15301*.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 541–548, Ann Arbor, Michigan. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 205–208, Sapporo, Japan. Association for Computational Linguistics.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data Augmentation for Low-Resource Neural Machine Translation. In *Proceedings of ACL*.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Michael Auli, and Armand Joulin. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48.

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

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

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

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64, Vancouver, British Columbia. Association for Computational Linguistics.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. 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*, pages 1–8, New York City, New York. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas. Association for Computational Linguistics.

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

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Yoon Kim. 2021. Sequence-to-Sequence Learning with Latent Neural Grammars. In *Proceedings of NeurIPS*.

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

Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of ACL*.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

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 M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of ICML*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yafu Li, Yongjing Yin, Yulong Chen, and Yue Zhang. 2021. On compositional generalization of neural machine translation. 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 4767–4780, Online. Association for Computational Linguistics.

Zhifei Li and Jason Eisner. 2009. First- and Second-Order Expectation Semirings with Applications to Minimum-Risk Training on Translation Forests. In *Proceedings of EMNLP*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3093–3102, Osaka, Japan. The COLING 2016 Organizing Committee.

Runjing Liu, Jeffrey Regier, Nilesh Tripuraneni, Michael Jordan, and Jon Mcauliffe. 2019. Rao-blackwellized stochastic gradients for discrete distributions. In *International Conference on Machine Learning*, pages 4023–4031. PMLR.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages

2283–2288, Austin, Texas. Association for Computational Linguistics.

Tetsuji Nakagawa. 2015. Efficient top-down BTG parsing for machine translation preordering. 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 208–218, Beijing, China. Association for Computational Linguistics.

Rebecca Nesson, Stuart Shieber, and Alexander Rush. 2006. Induction of Probabilistic Synchronous Tree-insertion Grammars for Machine Translation. In *Proceedings of AMTA*.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 843–853, Jeju Island, Korea. Association for Computational Linguistics.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2022. Improving compositional generalization with latent structure and data augmentation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4341–4362, Seattle, United States. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Wei Li Yanqi Zhou, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21.

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

Alexander M. Rush. 2020. Torch-Struct: Deep Structured Prediction Library. In *Proceedings of ACL (System Demonstrations)*.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Yutong Shao, Rico Sennrich, Bonnie Webber, and Federico Fancellu. 2018. Evaluating machine translation performance on Chinese idioms with a blacklist method. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2021. Substructure substitution: Structured data augmentation for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3494–3508, Online. Association for Computational Linguistics.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous Tree-Adjoining Grammars. In *COLING 1990 Volume 3: Papers presented to the 13th 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.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401*.

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

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021. Structured reordering for modeling latent alignments in sequence transduction. *Advances in Neural Information Processing Systems*, 34:13378–13391.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic. Association for Computational Linguistics.

Pia Weißenhorn, Yuekun Yao, Lucia Donatelli, and Alexander Koller. 2022. Compositional Generalization Requires Compositional Parsers. *arXiv:2202.11937*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic. Association for Computational Linguistics.

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

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Jingfeng Yang, Le Zhang, and Diyi Yang. 2022. SUBS: Subtree substitution for compositional semantic parsing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 169–174, Seattle, United States. Association for Computational Linguistics.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021a. Neural Bi-Lexicalized PCFG Induction. In *Proceedings of ACL.*

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

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 2021. Compositional generalization for neural semantic parsing via span-level supervised attention. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2810–2823, Online. Association for Computational Linguistics.

Shiyue Zhang, Benjamin Frey, and Mohit Bansal. 2020. ChrEn: Cherokee-English machine translation for endangered language revitalization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 577–595, Online. Association for Computational Linguistics.

Yanpeng Zhao and Ivan Titov. 2020. Visually Grounded Compound PCFGs. In *Proceedings of EMNLP*.

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

A Parameterization of BTG-Seq2Seq

The basic strategy for our parameterization is to take as much advantage of the backbone Transformer seq2seq models as possible, which is the main source of rich neural representations (especially when working with pretrained models).

Seq2Seq $p(d_{\text{leaf}}|x, d_{\text{tree}})$. The leaf nodes of d_{tree} are the production rules of $C[x_{i:k}] \rightarrow v$, which encode the correspondence between source segment $x_{i:k}$ and target segment v. The probability of such a rule is modeled by a conventional Transformer-based seq2seq model (Vaswani et al., 2017). In BTG-Seq2Seq, the underlying seq2seq is expected to handle segments at all levels, including sentence- and phrase-level translations. We found it beneficial to use two sets of special tokens to indicate the beginning/end of sentences, one for sentence-level translations and the other for phrase-level translations. Concretely, in addition to the typical 'BOS' and 'EOS' which are used to mark the beginning and the end of a sentence, we additionally use 'Seg-BOS' and 'Seg-EOS' to mark the beginning and the end of a segment.

Source parser $p(d_{\text{tree}}|x, n)$. To obtain a probabilistic parser, we need to assign a score to each rule involved in d_{tree}. For the following rules,

$$\forall n \in \{1, 2 \dots |x|\},$$
$$T^n \rightarrow S^n[x], \quad T^n \rightarrow I^n[x],$$

we assign them a weight of 1 for simplicity, as we find that using trainable weights does not help during our initial experiments.

We use neural features to parameterize the score of the following rules:

$$\begin{aligned}
&\forall i, j, k \in \{0, 1 \dots |x|\} \qquad\qquad \text{s.t. } i < j < k \\
&\quad S^n[x_{i:k}] \rightarrow I^l[x_{i:j}] S^r[x_{j:k}] \\
&\quad S^n[x_{i:k}] \rightarrow I^l[x_{i:j}] I^r[x_{j:k}] \\
&\quad I^n[x_{i:k}] \rightarrow S^l[x_{j:k}] I^r[x_{i:j}] \\
&\quad I^n[x_{i:k}] \rightarrow S^l[x_{j:k}] S^r[x_{i:j}].
\end{aligned}$$

For example the score

$$\pi(S^n[x_{i:k}] \rightarrow I^l[x_{i:j}] S^r[x_{j:k}]) = \exp\left(e_S^\top f(\tilde{\mathbf{h}}_{i:j}, \tilde{\mathbf{h}}_{j:k})\right)$$

relies on the difference feature $\tilde{\mathbf{h}}_{i:j}$ and $\tilde{\mathbf{h}}_{j:k}$, following Kitaev and Klein (2018). In our parameterization, rules that have the same right-hand spans will have the same score regardless of their nonterminals. That is, the score of $I^n[x_{i:k}] \rightarrow S^l[x_{j:k}] I^r[x_{i:j}]$ is identical to

$I^n[x_{i:k}] \rightarrow S^l[x_{j:k}]S^r[x_{i:j}]$. These difference features are calculated using

$$\tilde{\mathbf{h}}_{i:j} = \text{Concat}[\overrightarrow{\mathbf{h}}_j - \overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_{j+1} - \overleftarrow{\mathbf{h}}_{i+1}]$$

where $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ are the "forward" and "backward" representations from Kitaev and Klein (2018).[13]

The score of a derivation tree is the sum of the scores of the production rules that form the tree, and we normalize the score to obtain the distribution over all possible derivations,

$$p(d_{\text{tree}}|x, n) = \left(\prod_{R \in d_{\text{tree}}} \pi(R) \right) / Z_{\text{tree}}(x, n),$$

where $Z_{d_{\text{tree}}}(x, n) = \sum_{d_{\text{tree}}} \left(\prod_{R \in d_{\text{tree}}} \pi(R) \right)$ is the partition function. The dynamic program for computing the partition function is given by Algorithm 1.

Variational parser $q(d_{\text{tree}}, d_{\text{leaf}}|x, y, n)$. We factorize the variational synchronous parser as

$$q(d_{\text{tree}}, d_{\text{leaf}}|x, y, n) = q(d_{\text{tree}}|x, y, n) \times q(d_{\text{leaf}}|x, y, d_{\text{tree}}),$$

which makes it possible to sample hierarchical alignments in $O(L^3)$ where $L = \min(|x|, |y|)$. The parameterization of $q(d_{\text{tree}}|x, y, n)$ is very similar to the parameterization of $p(d_{\text{tree}}|x, n)$, except that $\mathbf{h}_i$ (which are used to compute the difference features) are the hidden states of a Transformer *decoder*, with the backbone Transformer seq2seq running from y to x. This choice makes it possible to condition the contextualized representations of x on the target y. Calculating the partition function uses the same dynamic program as in the prior parser (i.e., Algorithms 1). Algorithm 2 shows the dynamic program for sampling d_{tree}.

Given x, y and d_{tree}, the only rules that remain unknown for the variational parser are the target segments $y_{a:c}$ in $C[x_{i:k}] \rightarrow y_{a:c}$. Thus $q(d_{\text{leaf}}|x, y, d_{\text{tree}})$ is effectively a hierarchical segmentation model. We use a grammar with the following rules,

$$Root \rightarrow T^n[y], \; T^m[y_{a:c}] \rightarrow T^l[y_{a:b}]T^r[y_{b:c}],$$

to model the segmentations. Since the segmentation is conditioned on d_{tree}, we need to ensure that the tree structures of d_{leaf} and d_{tree} is identical. (Note that by conditioning on d_{tree}, we only utilize the tree topology of d_{tree} for efficiency.)

We use the following function to score the rules:

$$\pi\big([T^m[y_{a:c}] \rightarrow T^l[y_{a:b}]T^r[y_{b:c}]\big) = \exp\left(f(\tilde{\mathbf{h}}_{a:b}, \tilde{\mathbf{h}}_{b:c})\right),$$

where $\tilde{\mathbf{h}}_{a:b}$, $\tilde{\mathbf{h}}_{b:c}$ are the difference features based on the contextualized representations of y, and f is an MLP that emits a scalar score. We can then obtain a distribution over d_{leaf} via,

$$q(d_{\text{leaf}}|x, n, d_{\text{tree}}) = \left(\prod_{R \in d_{\text{leaf}}} \pi(R) \right) / Z_{\text{leaf}}(d_{\text{tree}}, y)$$

where $Z_{d_{\text{leaf}}}(d_{\text{tree}}, y) = \sum_{d_{\text{leaf}}} \left(\prod_{R \in d_{\text{leaf}}} \pi(R) \right)$ is the partition function. Algorithm 3 shows the dynamic program for calculating the partition function above, and Algorithm 4 gives the dynamic program for sampling from $q(d_{\text{leaf}}|x, n, d_{\text{tree}})$. These dynamic programs modify the standard inside algorithm to only sum over chart entries that are valid under d_{tree}.

Since we share the encoder and decoder across all components (i.e., prior/variational parsers and the seq2seq model), the only additional parameters for BTG-Seq2Seq are the parameters of the MLP layers for the three CRF parsers.

Number of segments $p(n|x)$, $q(n|x, y)$. We use a truncated geometric distribution of the following form,

$$p(n) = \begin{cases} \lambda(1-\lambda)^{n-1}, & n \in \{1, 2, \dots N-1\} \\ (1-\lambda)^{n-1}, & n = N \end{cases}$$

where N is the maximum number of segments. For the prior $p(n|x)$, we have $N = |x|$. For the variational posterior $q(n|x, y)$, we have $N = \min(|x|, |y|)$. In practice, we set an additional upper bound (chosen from $[3, 4 \dots 8]$) for efficient training. λ is chosen from $\{0.5, 0.6, 0.7, 0.8, 0.9\}$.

B Evidence Lower Bound

Figure 4 shows the lower bound derivation.

C Dynamic Programs for the Parsers

Sampling Algorithms

The algorithms for sampling from $p(d_{\text{tree}}|x, n)$ and $q(d_{\text{leaf}}|x, d_{\text{tree}})$ are provided in Algorithm 2 and 4. Sampling from $q(d_{\text{tree}}|x, y, n)$ is omitted, as it is the same algorithm as Algorithm 2.

[13]Originally, the forward and backward representations were defined based on the forward/backward hidden states from a bidirectional LSTM (Stern et al., 2017). Kitaev and Klein (2018) extend it to the Transformer architecture by splitting the hidden representation $\mathbf{h}_i$ from the Transformer encoder into halves and treating them as the forward and backward representations, i.e., $\mathbf{h}_i = \text{Concat}[\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$.

	SVO→SOV		EN→ZH			DE→EN			CHR↔EN	
	Novel-P	Few-Shot	VP	NP	PP	500	1000	4978	chr-en	en-chr
Train	20000	2000	914	812	906	500	1000	4978	11638	11638
Dev	5000	500	200	200	200	7584	7584	7584	1000	1000
Test	5000	500	200	200	200	6759	6759	6759	1000	1000
Transformer Encoder Layer	2	2	6	6	6	12	12	12	6	6
Transformer Encoder Size	1024	1024	512	512	512	1024	1024	1024	512	512
Transformer Decoder Layer	2	2	6	6	6	12	12	12	6	6
Transformer attention head	8	8	8	8	8	16	16	16	8	8
Positional embedding type	relative	relative	relative	relative	relative	absolute	absolute	absolute	relative	relative

Table 5: Dataset statistics and model hyperparameters used for experiments. Apart from the standard absolute positional encodings used by mBART, we use relative positional encodings whenever possible.

$$
\begin{aligned}
\log p(y|x) = \log \sum_{d \in D(x,y)} p(d|x) &= \log \sum_{d \in D(x,y)} p(n|x) p(d_{\text{tree}}|x,n) p(d_{\text{leaf}}|x, d_{\text{tree}}) \\
&\geq \mathbb{E}_{q(n|x,y)} \left\{ \boxed{\log \sum_{d_{\text{tree}}, d_{\text{leaf}}} p(d_{\text{tree}}|x,n) p(d_{\text{leaf}}|x, d_{\text{tree}})} \right\} - \mathrm{KL}[q(n|x,y) \| p(n|x)] \\
&\geq \mathbb{E}_{q(d_{\text{tree}}|x,y,n)} \left\{ \boxed{\log \sum_{d_{\text{leaf}}} p(d_{\text{leaf}}|x, d_{\text{tree}})} - \mathrm{KL}[q(d_{\text{tree}}|x,y,n) \| p(d_{\text{tree}}|x,n)] \right\} \\
&\geq \mathbb{E}_{q(d_{\text{leaf}}|x,y,d_{\text{tree}})} [\log p(d_{\text{leaf}}|x, d_{\text{tree}})] + \mathbb{H}[q(d_{\text{leaf}}|x,y,d_{\text{tree}})]
\end{aligned}
$$

$$
\begin{aligned}
\log p(y|x) \geq \mathbb{E}_{q(n|x,y)} \Big\{ & \mathbb{E}_{q(d_{\text{tree}}|x,y,n)} \Big[\mathbb{E}_{q(d_{\text{leaf}}|x,y,d_{\text{tree}})} [\log p(d_{\text{leaf}}|x, d_{\text{tree}})] + \mathbb{H}[q(d_{\text{leaf}}|x,y,d_{\text{tree}})] \Big] \\
& - \mathrm{KL}[q(d_{\text{tree}}|x,y,n) \| p(d_{\text{tree}}|x,n)] \Big\} - \mathrm{KL}[q(n|x,y) \| p(n|x)]
\end{aligned}
$$

Figure 4: Derivation of the evidence lower bound used for training. (Top) Here, line 3 lower bounds the boxed term in line 2, while line 4 lower bounds the boxed term in line 3. (Bottom) Taking these steps together, we can obtain the final lower bound.

	Mongolian (MN)	Marathi (MR)	Azerbaijani (AZ)	Bengali (BN)
Train	7607	9840	5946	4649
Dev	372	767	671	896
Test	414	1090	903	216

Table 6: Dataset sizes for four low-resource languages we choose from ML50 benchmark (Tang et al., 2020).

Computing the Entropy and KL

We provide the dynamic program to compute $\mathbb{H}[q(d_{\text{leaf}}|x,y,d_{\text{tree}})]$ in Algorithm 5, and $\mathrm{KL}[q(d_{\text{tree}}|x,y,n) \,||\, p(d_{\text{tree}}|x,n)]$ in Algorithm 6. Note that $\mathrm{KL}[q(n|x,y) \| p(n|x)]$ is a constant, and we can ignore it during optimization. In practice the algorithms are implemented in log space for numerical stability.

Gradient Estimation

We use policy gradients (Williams, 1992) to optimize the lower bound. Two techniques are additionally employed to reduce variance. First, we use a self-critical control variable (Rennie et al., 2017) where the scores from argmax trees are used as a control variates. Second, we utilize a sum-and-sample strategy (Liu et al., 2019) where we always train on $n = 1$, i.e., sentence-level translations, and sample another $n > 1$ to explore segment-level translations.

D Recipe for Using BTG-Seq2Seq

Although it is expensive to train our BTG-Seq2Seq from scratch due to $O(L^3)$ time complexity, we find that a pretrain-then-finetune strategy is a practical way to use BTG-Seq2Seq. During the pretraining stage, we train a Transformer seq2seq in the standard way until convergence, or simply use public model checkpoints. Then during the finetuning stage, we use the pretrained model as the backbone seq2seq for our BTG-SeqSeq. Usually we use a relatively smaller learning rate to finetune the backbone seq2seq during finetuning.

E Dataset Statistics and Hyperparameters

We show the statistics of the datasets and the hyperparameters of the model in Tables 5 and 6.

Algorithm 1 Inside algorithm for $p(d_{\text{tree}}|x, n)$. Rules are highlighted in blue

Input: x: source sentence, n: number of segments

function INSIDE-TREE(x, n)

Initialize $\beta_{...}[S^1] = 0$, $\beta_{...}[I^1] = 0$

for $m = 2$ **to** n **do** ▷ number of segments

  for $w = 1$ **to** $|x|$ **do** ▷ width of spans

    for $i = 0$ **to** $|x| - w$ **do** ▷ start point

      $k = i + w$ ▷ end point

      for $j = i + 1$ **to** $k - 1$ **do**

        for $l = 1$ **to** $m - 1$ **do** ▷ number of segments in the left part

          $r = m - l$ ▷ number of segments in the right part

          $\beta_{i:k}[S^m] \mathrel{+}= \pi(S^m[x_{i:k}] \to I^l[x_{i:j}] S^r[x_{j:k}]) \cdot \beta_{i:j}[I^l] \cdot \beta_{j:k}[S^r]$

          $\beta_{i:k}[S^m] \mathrel{+}= \pi(S^m[x_{i:k}] \to I^l[x_{i:j}] I^r[x_{j:k}]) \cdot \beta_{i:j}[I^l] \cdot \beta_{j:k}[I^r]$

          $\beta_{i:k}[I^m] \mathrel{+}= \pi(I^m[x_{i:k}] \to S^l[x_{j:k}] I^r[x_{i:j}]) \cdot \beta_{j:k}[S^l] \cdot \beta_{i:j}[I^r]$

          $\beta_{i:k}[I^m] \mathrel{+}= \pi(I^m[x_{i:k}] \to S^l[x_{j:k}] S^r[x_{i:j}]) \cdot \beta_{j:k}[S^l] \cdot \beta_{i:j}[S^r]$

▷ Compute locally-normalized scores $\tilde{\pi}$

for $m = 2$ **to** n **do** ▷ number of segments

  for $w = 1$ **to** $|x|$ **do** ▷ width of spans

    for $i = 0$ **to** $|x| - w$ **do** ▷ start point

      $k = i + w$ ▷ end point

      for $j = i + 1$ **to** $k - 1$ **do**

        for $l = 1$ **to** $m - 1$ **do** ▷ number of segments in the left part

          $r = m - l$ ▷ number of segments in the right part

          ▷ Denote $S^m[x_{i:k}] \to I^l[x_{i:j}] S^r[x_{j:k}]$ as R_1

          $\tilde{\pi}(R_1) = \left(\pi(R_1) \cdot \beta_{i:j}[I^l] \cdot \beta_{j:k}[S^r]\right) / \beta_{i:k}[S^m]$

          ▷ Denote $S^m[x_{i:k}] \to I^l[x_{i:j}] I^r[x_{j:k}]$ as R_2

          $\tilde{\pi}(R_2) = \left(\pi(R_2) \cdot \beta_{i:j}[I^l] \cdot \beta_{j:k}[I^r]\right) / \beta_{i:k}[S^m]$

          ▷ Denote $I^m[x_{i:k}] \to S^l[x_{j:k}] I^r[x_{i:j}]$ as R_3

          $\tilde{\pi}(R_3) = \left(\pi(R_3) \cdot \beta_{j:k}[S^l] \cdot \beta_{i:j}[I^r]\right) / \beta_{i:k}[I^m]$

          ▷ Denote $I^m[x_{i:k}] \to S^l[x_{j:k}] S^r[x_{i:j}]$ as R_4

          $\tilde{\pi}(R_4) = \left(\pi(R_4) \cdot \beta_{j:k}[S^l] \cdot \beta_{i:j}[S^r]\right) / \beta_{i:k}[I^m]$

▷ T rules are assigned a trivial score: $\pi(T^n \to S^n[x]) = 1$, $\pi(T^n \to I^n[x]) = 1$

$\beta[T^n] = \beta_{0:|x|}[S^n] + \beta_{0:|x|}[I^n]$, $Z_{\text{tree}}(x, n) = \beta[T^n]$ ▷ partition function

$\tilde{\pi}(T^n \to S^n[x]) = \beta_{0:|x|}[S^n] / \beta[T^n]$, $\tilde{\pi}(T^n \to I^n[x]) = \beta_{0:|x|}[I^n] / \beta[T^n]$

return $Z_{\text{tree}}(x, n), \beta, \tilde{\pi}$

Algorithm 2 Top-down sampling d_{tree} from $p(d_{\text{tree}}|x, n)$. Rules are highlighted in blue

Input: $\tilde{\pi}$: normalized scores obtained from INSIDE-TREE($\cdot$), n: number of segments

function SAMPLE-TREE($\tilde{\pi}, n$)
  Initialize an empty tree d_{tree}
  Sample $A \in \{S, I\}$ wrt. $\tilde{\pi}(T^n \to A^n[x])$
  RECUR-SAMPLE-TREE(0, $|x|$, n, A, $\tilde{\pi}$, d_{tree})
  return d_{tree}

▷ Arguments: i: start point, k: end point, n: number of segments, A: nonterminal)
function RECUR-SAMPLE-TREE(i, k, n, A, $\tilde{\pi}$, d_{tree})
  if $n = 1$ **then**
    Add rule $A^1[x_{i:k}] \to C[x_{i:k}]$ to d_{tree}
    return
  if $A = S$ **then**
    ▷ $1 \leq l < n, A, B \in \{S, I\}, i < j < k$ are the random variables, $r = n - l$
    Sample rule $S^n[x_{i:k}] \to A^l[x_{i:j}]B^r[x_{j:k}]$ according to $\tilde{\pi}(\cdot)$
    Add $S^n[x_{i:k}] \to A^l[x_{i:j}]B^r[x_{j:k}]$ to d_{tree}
    RECUR-SAMPLE-TREE(i, j, l, A, $\tilde{\pi}$, d_{tree})
    RECUR-SAMPLE-TREE(j, k, r, B, $\tilde{\pi}$, d_{tree})
  else ▷ $A = I$
    ▷ $1 \leq l < n, A, B \in \{S, I\}, i < j < k$ are the random variables, $r = n - l$
    Sample rule $I^n[x_{i:k}] \to A^l[x_{j:k}]B^r[x_{i:j}]$ according to $\tilde{\pi}(\cdot)$
    Add $I^n[x_{i:k}] \to A^l[x_{j:k}]B^r[x_{i:j}]$ to d_{tree}
    RECUR-SAMPLE-TREE(j, k, l, A, $\tilde{\pi}$, d_{tree})
    RECUR-SAMPLE-TREE(i, j, r, B, $\tilde{\pi}$, d_{tree})

Algorithm 3 Inside algorithm for $q(d_{\text{leaf}}|x, y, d_{\text{tree}})$.

Input: y: target sentence, d_{tree}: derivation tree

function INSIDE-LEAF(y, d_{tree})
  Infer number of segments n from d_{tree}
  Initialize $\beta_{...}[\ldots] = 0$
  for $m = 2$ **to** N **do** ▷ number of segments
    for $w = 1$ **to** $|x|$ **do** ▷ width of spans
      for $i = 0$ **to** $|x| - w$ **do** ▷ start point
        $k = i + w$ ▷ end point
        for $j = i + 1$ **to** $k - 1$ **do**
          for $l = 1$ **to** $m - 1$ **do** ▷ number of segments in the left part
            $r = m - l$ ▷ number of segments in the right part
            if there exist a rule like $A^m \to B^l C^r$ in d_{tree} **then**
              $\beta_{i:k}[T^m] \mathrel{+}= \pi(T^m[y_{i:k}] \to T^l[y_{i:j}]T^r[y_{j:k}]) \cdot \beta_{i:j}[T^l] \cdot \beta_{j:k}[T^r]$
  return β

Algorithm 4 Top-down sampling d_{leaf} from $q(d_{\text{leaf}}|x, y, d_{\text{tree}})$.

Input: β: inside scores returned from INSIDE-LEAF($\cdot$), d_{tree}: derivation tree, y: target sentence

function SAMPLE-LEAF(β, d_{tree}, y)
  Infer the number of segments n from d_{tree}
  Initialize an empty tree d_{leaf}
  RECUR-SAMPLE-LEAF(0, $|y|$, n, β, d_{tree}, d_{leaf})
  return d_{leaf}

▷ Arguments: i: start point, k: end point, n: number of segments
function RECUR-SAMPLE-LEAF(i, k, n, β, d_{leaf}, d_{tree})
  if $n = 1$ **then**
    Add rule $T^1[y_{i:k}]\to y_{i:k}$ to d_{leaf}
    return

  Infer l, r based on d_{tree} and n
  Compute $Z = \sum_j(\beta_{i:j}[T^l] \cdot \beta_{j:k}[T^r])$ ▷ normalization term
  for $j = i + 1$ **to** $k - 1$ **do**
    $\tilde{\pi}(T^n[y_{i:k}]\to T^l[y_{i:j}]T^r[y_{j:k}]) = (\beta_{i:j}[T^l] \cdot \beta_{j:k}[T^r])/Z$

  Sample rule $T^n[y_{i:k}]\to T^l[y_{i:j}]T^r[y_{j:k}]$, and add it to d_{leaf} ▷ the rule only encodes split point j
  RECUR-SAMPLE-LEAF(i, j, l, β, d_{tree}, d_{leaf})
  RECUR-SAMPLE-LEAF(j, k, r, β, d_{tree}, d_{leaf})

Algorithm 5 Calculating the entropy of $\mathbb{H}[q(d_{\text{leaf}}|x, y, d_{\text{tree}})]$

Input: β: inside scores returned by INSIDE-LEAF($\cdot$), d_{tree}: derivation tree, y: target sentence

function COMPUTE-LEAF-ENTROPY(β, d_{tree}, y)
  Find the number of segments n from d_{tree}
  return RECUR-COMPUTE-LEAF-ENTROPY(0, $|y|$, n, β, d_{tree})

▷ Arguments: i: start point, k: end point, n: number of segments
function RECUR-COMPUTE-LEAF-ENTROPY(i, k, n, β, d_{tree})
  if $n = 1$ **then**
    return 0

  Infer l, r based on d_{tree} and n
  Compute $Z = \sum_j(\beta_{i:j}[T^l] \cdot \beta_{j:k}[T^r])$ ▷ normalization term
  for $j = i + 1$ **to** $k - 1$ **do**
    $\tilde{\pi}(T^n[y_{i:k}]\to T^l[y_{i:j}]T^r[y_{j:k}]) = (\beta_{i:j}[T^l] \cdot \beta_{j:k}[T^r])/Z$

  $H = 0$
  for $j = i + 1$ **to** $k - 1$ **do**
    H_l = RECUR-COMPUTE-LEAF-ENTROPY($i, j, l, \beta, d_{\text{tree}}$)
    H_r = RECUR-COMPUTE-LEAF-ENTROPY($j, k, r, \beta, d_{\text{tree}}$)
    ▷ Denote rule $T^n[y_{i:k}]\to T^l[y_{i:j}]T^r[y_{j:k}]$ as R
    $H \mathrel{+}= (H_l + H_r - \log[\tilde{\pi}(R)]) \cdot \tilde{\pi}(R)$
  return H ▷ return the entropy

Algorithm 6 Computing the KL-divergence of $\mathrm{KL}[q(d_{\text{tree}}|x,y,n) \parallel p(d_{\text{tree}}|x,n)]$

Input: $\tilde{\pi}_q$: normalized rule scores returned by INSIDE-TREE$(\cdot)$ for $q(d_{\text{tree}}|x,y,n)$, $\tilde{\pi}_p$: rule scores obtained similarly by calling INSIDE-TREE$(\cdot)$ for $p(d_{\text{tree}}|x,n)$, n: number of segments, x: source sentence

function COMPUTE-TREE-KL($\tilde{\pi}_q, \tilde{\pi}_p, n, x$)
  Initialize $\mathrm{KL}_{\dots}[\cdot] = 0$
  for $m = 2$ **to** n **do** ▷ number of segments
    for $w = 1$ **to** $|x|$ **do** ▷ width of spans
      for $i = 0$ **to** $|x| - w$ **do** ▷ start point
        $k = i + w$ ▷ end point
        for $j = i + 1$ **to** $k - 1$ **do**
          for $l = 1$ **to** $m - 1$ **do** ▷ number of segments in the left part
            $r = m - l$ ▷ number of segments in the right part
            ▷ Denote $S^m[x_{i:k}] \to I^l[x_{i:j}]S^r[x_{j:k}]$ as R_1
            $\mathrm{KL}_{i,k}[S^m] \mathrel{+}= \big(\mathrm{KL}_{i:j}[I^l] + \mathrm{KL}_{j:k}[S^r] + \log[\tilde{\pi}_q(R_1)] - \log[\tilde{\pi}_p(R_1)]\big) \cdot \tilde{\pi}_q(R_1)$
            ▷ Denote $S^m[x_{i:k}] \to I^l[x_{i:j}]I^r[x_{j:k}]$ as R_2
            $\mathrm{KL}_{i,k}[S^m] \mathrel{+}= \big(\mathrm{KL}_{i:j}[I^l] + \mathrm{KL}_{j:k}[I^r] + \log[\tilde{\pi}_q(R_2)] - \log[\tilde{\pi}_p(R_2)]\big) \cdot \tilde{\pi}_q(R_2)$
            ▷ Denote $I^m[x_{i:k}] \to S^l[x_{j:k}]I^r[x_{i:j}]$ as R_3
            $\mathrm{KL}_{i,k}[I^m] \mathrel{+}= \big(\mathrm{KL}_{j:k}[S^l] + \mathrm{KL}_{i:j}[I^r] + \log[\tilde{\pi}_q(R_3)] - \log[\tilde{\pi}_p(R_3)]\big) \cdot \tilde{\pi}_q(R_3)$
            ▷ Denote $I^m[x_{i:k}] \to S^l[x_{j:k}]S^r[x_{i:j}]$ as R_4
            $\mathrm{KL}_{i,k}[I^m] \mathrel{+}= \big(\mathrm{KL}_{j:k}[S^l] + \mathrm{KL}_{i:j}[S^r] + \log[\tilde{\pi}_q(R_4)] - \log[\tilde{\pi}_p(R_4)]\big) \cdot \tilde{\pi}_q(R_4)$
  ▷ Denote $T^n \to S^n[x]$, $T^n \to I^n[x]$ as R_5, R_6 respectively
  $\mathrm{KL}[T^n] \mathrel{+}= \big(\mathrm{KL}_{0:|x|}[S^n] + \log[\tilde{\pi}_q(R_5)] - \log[\tilde{\pi}_p(R_5)]\big) \cdot \tilde{\pi}_q(R_5)$
  $\mathrm{KL}[T^n] \mathrel{+}= \big(\mathrm{KL}_{0:|x|}[I^n] + \log[\tilde{\pi}_q(R_6)] - \log[\tilde{\pi}_p(R_6)]\big) \cdot \tilde{\pi}_q(R_6)$
  return $\mathrm{KL}[T^n]$