Explicit Syntactic Guidance for Neural Text Generation

Yafu Li[♠♣*], **Leyang Cui**[♡†], **Jianhao Yan**[♠♣], **Yongjng Yin**[♠♣]
Wei Bi[♡], **Shuming Shi**[♡], **Yue Zhang**[♣◇†]
[♠] Zhejiang University [♡] Tencent AI lab
[♣] School of Engineering, Westlake University
[◇] Institute of Advanced Technology, Westlake Institute for Advanced Study
yafuly@gmail.com
{leyangcui,victoriabi,shumingshi}@tencent.com
{yanjianhao,yinyongjing,zhangyue}@westlake.edu.cn

Abstract

Most existing text generation models follow the sequence-to-sequence paradigm. *Generative Grammar* suggests that humans generate natural language texts by learning language grammar. We propose a syntax-guided generation schema, which generates the sequence guided by a constituency parse tree in a top-down direction. The decoding process can be decomposed into two parts: (1) predicting the infilling texts for each constituent in the lexicalized syntax context given the source sentence; (2) mapping and expanding each constituent to construct the next-level syntax context. Accordingly, we propose a structural beam search method to find possible syntax structures hierarchically. Experiments on paraphrase generation and machine translation show that the proposed method outperforms autoregressive baselines, while also demonstrating effectiveness in terms of interpretability, controllability, and diversity.

1 Introduction

Natural language generation (NLG), such as paraphrase generation (Sun et al., 2021), text summarization (Lin et al., 2018), machine translation (Vaswani et al., 2017; Edunov et al., 2018), and language models (Brown et al., 2020; OpenAI, 2023), have shown remarkable progress in the past few years. Most of the highest-performing NLG models train the model based on source-target correspondence and conduct autoregressive inference, which achieves competitive empirical performances yet deviates from a range of desirable attributes of human language generation, e.g., lack of interpretability (Alvarez-Melis and Jaakkola, 2017; He et al., 2019; Li and Yao, 2021).

It has been shown that humans generate language by learning and manipulating language grammar (Zholkovskii and Mel'chuk, 1965; Montague, 1974), which generative grammar (Chomsky, 1965) considers as a finite rule set that combines words to form grammatical sentences, thereby avoiding enumeration of surface sequences, which can significantly increase data sparsity and reducing learning efficiency (Li et al., 2021; Dankers et al., 2022). In this process, syntax plays a crucial role, imposing constraints on how to construct sentences. Syntax knowledge has been found *implicitly* contained by deep neural models (Kovaleva et al., 2019; Clark et al., 2019) and also useful for NLG tasks (Yang et al., 2020a; Sun et al., 2021; Xie et al., 2021). However, relatively little recent work has considered *explict* syntax in NLG (Wang et al., 2018).

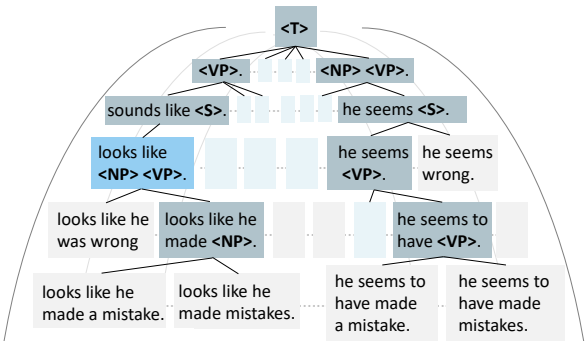

Figure 1: Syntax-guided generation: searching the hypotheses hierarchically throughout the syntax tree in a top-down direction, starting from the root node "<T>". The green blocks denote the possible syntax structures at different tree depths, the blue one denotes the external modification, whereas the gray ones denote the finalized hypotheses, marking the end of search paths.

Inspired by the above psycholinguistic observation, we propose a syntax-guided generation scheme, which generates text by following a well-defined grammar. As shown in Figure 1, instead of sequential generation, the model generates the sentence in a hierarchically top-down manner guided by the constituency parse tree, starting with the root node <T>. Syntactic categories such as noun phrases <NP> and verb phrases <VP> are integrated with tokens in the generation process, and

[*] Work was done during the internship at Tencent AI lab.
[†] Corresponding authors.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 14095–14112
July 9-14, 2023 ©2023 Association for Computational Linguistics

the model simultaneously considers multiple syntax structures at each tree depth, hierarchically exploring the syntax tree for reasonable hypotheses.

Intuitively, such a generation paradigm has the following advantages compared with autoregressive generation. First, akin to the language learning process of human beings, grammar learning breaks down non-enumerable surface sequences into finite pieces, acting as a training curriculum. Second, it provides an effective and interpretable pathway to probe into the generation process. Consequently, generation errors can be traced back to specific constituent expansion at the respective tree depth. Third, one can manipulate the generation process by exerting versatile control at arbitrary depths, e.g., modifying the translation of a verb phrase and constraining the paraphrase style with syntax templates. Forth, diverse sequences can be generated by exploring various syntax structures hierarchically throughout the syntax tree.

We implement the above process on Transformer (Vaswani et al., 2017). As shown in Figure 1, the generation process proceeds under the guidance of syntactic grammar. Starting from the root node "<T>", the model recursively generates the infilling texts (e.g., "he" and "seems <S>") for each constituent in the current lexicalized syntax context (e.g, "<NP> <VP>."), and infills each one accordingly to construct the next-level lexicalized syntax context (e.g., "he seems <S>."). The generation proceeds until there is no remaining constituent. The infilling texts are predicted by a Transformer-based model, which is trained by maximizing the likelihood of infilling texts for each constituent in the syntax context based on the source input. To explore more syntactically diverse and reasonable hypotheses during inference, we propose *structural beam search*, which searches promising syntax structures over the entire syntax tree in a top-down manner, as shown in Figure 1.

To isolate the effect of syntax and avoid the influence of other transformation factors, we conduct experiments on two sequence-to-sequence (seq2seq) tasks *with semantic equivalence* between the source and target sequences: paraphrase generation and machine translation. Empirical results demonstrate that our method can generate sequences with higher quality than the seq2seq baselines. Quantitative analysis demonstrates that the generation process can be interpreted effectively. In addition, our method demonstrates the capability of executing control from both syntax templates and fine-grained manual modifications. Finally, we show the diversity advantage through both automatic evaluation and human evaluation. We release the code on https://github.com/yafuly/SyntacticGen.

2 Related Work

Syntax as Extra Input. A line of work incorporates syntax knowledge as extra input to boost task performance. In paraphrase generation, Iyyer et al. (2018), Chen et al. (2019), Kumar et al. (2020) and (Sun et al., 2021) additionally encode a constituency tree to produce controllable paraphrases. For machine translation, researchers utilize syntactic information to boost the neural machine translation system using syntactic encoders (Li et al., 2017; Ma et al., 2018; Eriguchi et al., 2019; Ma et al., 2020; Yang et al., 2020a), position encoding (Ma et al., 2019; Xie et al., 2021), attention mechanism (Chen et al., 2018; Peng et al., 2019), and auxiliary training objectives (Ma et al., 2019).

Syntax for Generation Guidance. Different from the above work, we focus on guiding generation explicitly following syntactic grammar. Typically, Aharoni and Goldberg (2017) and Le et al. (2017) learn the mapping from sequences to linearized constituency trees to improve machine translation. Eriguchi et al. (2017) proposes a hybrid decoder with RNNG (Dyer et al., 2016) to jointly learn parse actions and word predictions. Wu et al. (2017) and Wang et al. (2018) design a syntactic tree decoder based on LSTM (Hochreiter and Schmidhuber, 1997), with an extra rule decoder. Yang et al. (2020b) introduce a syntax-guided soft target template as extra prompts in Transformer. Different from their work, our method leverages Transformer strengths and breaks down the sequence-to-sequence generation process into a hierarchically top-down generation guided by the syntax tree.

3 Method

3.1 Baseline Transformer

Transformer models the correspondence between the source sequence $\mathbf{x} = \{x_1, \ldots, x_{|\mathbf{x}|}\}$ and the target sequence $\mathbf{y} = \{y_1, \ldots, y_{|\mathbf{y}|}\}$ in an end-to-end fashion. The Transformer encoder transforms the discrete source sequence $\mathbf{x}$ into a continuous representation, which the Transformer decoder utilizes

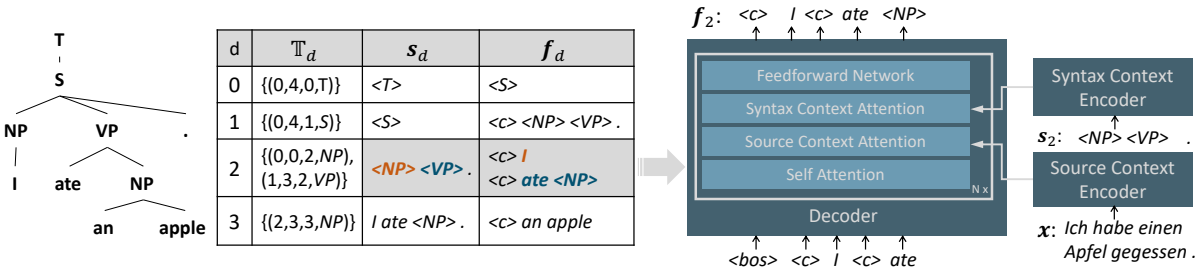

Figure 2: Method illustration: the left part demonstrates the construction of the training triplet (i.e., $(\mathbf{x}, \boldsymbol{s}_d, \boldsymbol{f}_d)$) based on the constituency parse tree; the right part denotes the architecture of the neural decoder, which takes in the German source sentence $\mathbf{x}$ and the syntax context $\boldsymbol{s}_2$ as input, and predicts the infilling text $\boldsymbol{f}_2$.

to generate the target sequence. The conditional probability $p(\mathbf{y}|\mathbf{x})$ can be factorized in an autoregressive way:

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p_{\boldsymbol{\theta}}(y_t|\mathbf{x}, y_{1:t-1}), \quad (1)$$

where $\boldsymbol{\theta}$ denotes the model parameters.

Given a source-target training set $\mathcal{D} = \{\mathbf{x}^i, \mathbf{y}^i\}|_{i=1}^{|\mathcal{D}|}$, the model is optimized by minimizing the cross-entropy (CE) loss:

$$\mathcal{L}_{ce}^{\mathcal{D}} = -\sum_{i=1}^{|\mathcal{D}|}\sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}(y_t^i|\mathbf{x}^i, y_{1:t-1}^i). \quad (2)$$

3.2 Syntax-guided Generation

In this section, we introduce syntax-guided generation, which generates texts by hierarchically expanding constituents in syntax contexts throughout the syntax tree, while also leveraging the strengths of Transformer. In general, the generation process can be decomposed into two stages: (1) **neural generation**: the neural decoder (Section 3.2.2) generates the infilling sequences based on the source sequence and the syntax context; (2) **constituent expansion**: predicted infilling sequences are mapped and filled into each constituent in the syntax context accordingly (Section 3.2.3), forming the next-level syntax context. To facilitate parallelism during training, we decompose the sequence-to-sequence dataset to a triplet set, where the neural decoder is optimized to maximize the probability of the infilled sequence (e.g., "<c> I <c> ate <NP> .") given the lexicalized syntax context (e.g., "<NP> <VP> ."), as shown in Figure 2.

3.2.1 Triplet Construction

Given a target sequence $\mathbf{y}$, the corresponding constituency parse tree of depth $|\mathbb{T}|$ can be composed by a set of labeled spans $\mathbb{T}$:

$$\mathbb{T} = \{\mathbb{T}_d\}|_{d=1}^{|\mathbb{T}|} = \{\{(a_k, b_k, d, l_k)\}|_{k=1}^{|\mathbb{T}_d|}\}|_{d=1}^{|\mathbb{T}|}, \quad (3)$$

where a_k and b_k represent the k-th constituent span's fencepost positions at depth d, and l_k represents the constituent label. Our model is optimized to predict the next-level span sets $\mathbb{T}_d$ given the previous one and the source input, i.e., $p_{\boldsymbol{\theta}}(\mathbb{T}_d|\mathbb{T}_{d-1}, \mathbf{x})$.

Given the set of labeled spans at depth d, i.e., $\mathbb{T}_d$, we transform the target sequence into a lexicalized syntax sequence of length $|\boldsymbol{s}_d|$: $\boldsymbol{s}_d = \{s_{d;1}, s_{d;2}, \ldots, s_{d;|\boldsymbol{s}_d|}\}$, by keeping the lexical tokens and replacing the constituent spans with corresponding labels. For instance, the sequence "I ate an apple ." is transformed to $\boldsymbol{s}_2$ ={<NP>,<VP>,.} at depth 2, and is transformed to $\boldsymbol{s}_3$ ={I,ate,<NP>,.} at depth 3, as shown in Figure 2. The alignment between $\boldsymbol{s}_2$ and $\boldsymbol{s}_3$ can be modeled as a text-infilling task. For example, the {<NP>}, {<VP>} and at depth 2 are replaced by {I} and {ate <NP>} at depth 3, respectively. To generate the whole $\boldsymbol{s}_3$ based on $\boldsymbol{s}_2$ in one pass, we concatenate all the infilling texts with a special token "<c>", yielding an infilling sequence $\boldsymbol{f}_2$ = {<c>,I,<c>,ate,<NP>}.

Similarly for each syntax context $\boldsymbol{s}_d$, we collect the respective infilling texts for each constituent in the lexicalized sequence at depth $d+1$, and concatenate them to construct the target infilling sequence of length $|\boldsymbol{f}_d|$: $\boldsymbol{f}_d = \{f_{d;1}, f_{d;2}, \ldots, f_{d;|\boldsymbol{f}_d|}\}$. In this way, a triplet is constructed for a source-target sequence pair at depth d: $\{(\mathbf{x}, \boldsymbol{s}_d, \boldsymbol{f}_d)\}$. We traverse the target syntax tree in level-order to obtain the full set Φ of training triplets for a training instance:

$$\Phi = \{\Phi_d\}|_{d=1}^{|\mathbb{T}|-1} = \{(\mathbf{x}, \boldsymbol{s}_d, \boldsymbol{f}_d)\}|_{d=1}^{|\mathbb{T}|-1}. \quad (4)$$

Given a sequence-to-sequence training set $\mathcal{D}$ =

$\{\mathbf{x}^i, \mathbf{y}^i\}|_{i=1}^{|\mathcal{D}|}$, we go through the full training set to construct the complete triplet set Ψ:

$$\Psi = \{\Phi^i\}|_{i=1}^{|\mathcal{D}|} = \{(\mathbf{x}^j, \boldsymbol{s}^j, \boldsymbol{f}^j)\}|_{j=1}^{\sum_{i=1}^{|\mathcal{D}|}|\Phi^i|}. \quad (5)$$

3.2.2 Neural Decoder

Given a triplet instance Ψ^j, we construct the **neural decoder** based on Transformer to model the generative probability $p_{\boldsymbol{\theta}}(\boldsymbol{f}^j|\mathbf{x}^j, \boldsymbol{s}^j)$. The neural decoder takes the source sequence and the lexicalized syntax context as input and generates the corresponding infilling texts, as shown in Figure 2.

Besides the encoder that encodes source context, we introduce an extra Transformer encoder, i.e., syntax context encoder, to encode the lexicalized syntax context into a representation. On top of self-attention and source context attention, we insert an extra attention layer (syntax context attention) into each decoder layer to incorporate syntax contexts, as shown in the right part of Figure 2.

Similarly, the probability of the infilling sequence can be factorized as:

$$p_{\boldsymbol{\theta}}(\boldsymbol{f}|\mathbf{x}, \boldsymbol{s}) = \prod_{t=1}^{|\boldsymbol{f}|} p_{\boldsymbol{\theta}}(f_t|\mathbf{x}, \boldsymbol{s}, f_{1:t-1}). \quad (6)$$

We define the scoring function for an infilling sequence as the sum of the log probabilities:

$$\text{score}(\mathbf{x}, \boldsymbol{s}, \boldsymbol{f}) = \sum_{t=1}^{|\boldsymbol{f}|} \log p_{\boldsymbol{\theta}}(f_t|\mathbf{x}, \boldsymbol{s}, f_{1:t-1}). \quad (7)$$

We adopt the standard cross-entropy loss (CE loss) to optimize our model, where the loss for the j-th triplet in the training set Ψ can be written as:

$$\mathcal{L}_{ce}^j = -\sum_{t=1}^{|\boldsymbol{f}^j|} \log p_{\boldsymbol{\theta}}(f_t^j|\mathbf{x}^j, \boldsymbol{s}^j, f_{1:t-1}^j), \quad (8)$$

and the CE loss across the whole triple set Ψ becomes:

$$\mathcal{L}_{ce}^{\Psi} = \sum_{j=1}^{|\Psi|} \mathcal{L}_{ce}^j. \quad (9)$$

3.2.3 Generation Process

Given a source sequence, our model generates the target sequence in a top-down manner which is grounded on syntactic grammar rules. As shown in Figure 2, the neural decoder first encodes the source sequence $\mathbf{x}$ into the source context representation $\mathbf{h}_{src}$, which remains fixed and can be reused throughout the generation process. Initially, the neural decoder generates the infilling sequences $\mathbf{t}_0$ given $\boldsymbol{x}$ and $\boldsymbol{s}_0$ ={<T>}, based on Equation 6. Then the model proceeds with the generation process via iteratively generating infilling texts and expanding constituents.

At each iteration step (i.e., tree depth), the neural decoder generates the infilling sequence $\boldsymbol{f}_d$ for the syntax context $\boldsymbol{s}_d$:

$$\boldsymbol{f}_d = \arg\max_{\boldsymbol{f'}} p_{\boldsymbol{\theta}}(\boldsymbol{f'}|\mathbf{x}, \boldsymbol{s}_d) \quad (10)$$

Then the constituent expansion function yields the next-level syntax context given the syntax context and the infilling sequences predicted by the neural decoder:

$$\boldsymbol{s}_{d+1} = \text{expand}(\boldsymbol{s}_d, \boldsymbol{f}_d). \quad (11)$$

Specifically, we first separate the infilling sequences by the special separator "<C>" into a group of infilling texts, e.g., spliting $\boldsymbol{f}_2$ ={{<C>,I,<C>,ate,<NP>}} to {{I},{ate <NP>}}. Then we fill in each of the infilling texts into the corresponding constituent in the syntax context $\boldsymbol{s}_2$ to obtain the syntax context at the following level, e.g., $\boldsymbol{s}_3$={I,ate,<NP>,.}. The syntax context encoder encodes the updated syntax context $\boldsymbol{s}_{d+1}$ and starts the next iteration. The remaining decoding process loops between these two stages, until there is no constituent label in the syntax context, or a maximum tree depth is reached, as shown in Figure 2.

As the model behavior on expanding constituents over the entire syntax tree is completely accessible, the generation process can be effectively interpreted, as shown in Section 6.2. Moreover, manual modifications can be directly incorporated into the expansion process for each constituent throughout the syntax tree (Section 6.3). Finally, more than one syntax structure can be considered simultaneously at each tree depth, enabling searching for hypotheses of better syntactical diversity(Section 6.4).

3.2.4 Structural Beam Search

By default, our model selects the best infilling texts greedily in each iteration. We introduce **structural beam search** to explore the hypothesis space for a more accurate and diverse generation. Similar to standard beam search (Sutskever et al., 2014), structural beam search maintains a beam width of

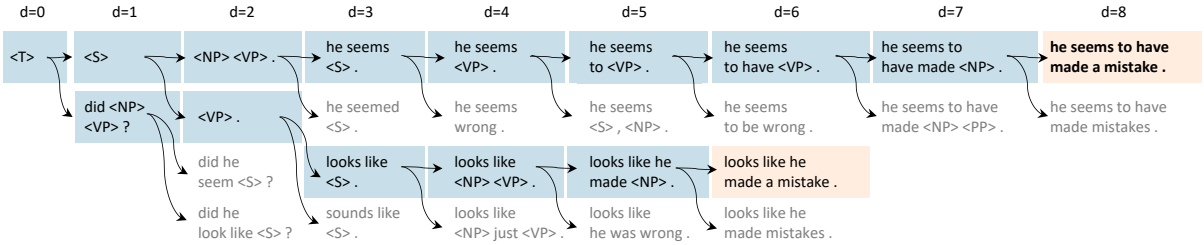

Figure 3: A real example of our model generating a paraphrase given the source sequence "it seems like he has made a mistake.", under the structural beam search of width 2. Diverse syntax structures are explored during the generation, e..g, "<VP>.", "<NP> <VP>.", and "did <NP> <VP>?".

candidates at each iteration. Thanks to explicitly traversing the constituency parse tree during inference, our method is able to search promising syntax structures throughout the syntax tree in a top-down manner. We show a real example of our model generating a paraphrase in Figure 3.

At each level, we apply standard beam search for neural generation and keep top k infilling texts along with their scores, computed by Equation 7. Taking previous predictions into consideration, we introduce a moving average mechanism to trade off confidence between the predictions from lower levels and the current-level prediction. Specifically, suppose $\boldsymbol{s}_i$ is the i-th syntax context in the k-width beam at the current depth, with an accumulated score of $\delta_{\boldsymbol{s}_i}$; and $\boldsymbol{f}_{j;\boldsymbol{s}_i}$ is the j-th infilling sequence candidate from the neural generation beam given the syntax context $\boldsymbol{s}_i$, with a score of $\delta_{\boldsymbol{f}_{j;\boldsymbol{s}_i}}$. A beam of next-level syntax contexts is constructed, by filling in the current syntax context with the corresponding infilling sequences:

$$\boldsymbol{s}_{ik+j} = \text{expand}(\boldsymbol{s}_i, \boldsymbol{f}_{j;\boldsymbol{s}_i}). \tag{12}$$

The updated score for each of the next-level syntax contexts in the beam is given by:

$$\delta_{ik+j} = \alpha\delta_{\boldsymbol{s}_i} + (1-\alpha)\delta_{\boldsymbol{f}_{j;\boldsymbol{s}_i}}, \tag{13}$$

where α is a hyper-parameter (**accumulation weight**) that determines how much weight is put on predictions at lower levels. Then the beam is further pruned by their updated scores to maintain the beam width. For example, the first two candidate syntax contexts are selected at depth 2 in Figure 3. Algorithm implementation details can be referred to in Appendix A.

4 Experiment Setup

Datasets For paraphrase generation, we experiment on ParaNMT-small (Chen et al., 2019), which contains 500K sentence-paraphrase pairs for training, 500 for validation, and 800 for testing. Both validation and test sets are provided with human-annotated sentence exemplars from which syntax information can be extracted for controlling paraphrase generation. For machine translation, we use NIST Chinese-English (Zh-En), WMT'16 Romanian-English (Ro-En), WMT'14 English-German (De-En), and WMT'14 English-German (En-De). For WMT datasets, we follow the official split for validation and testing. For NIST Zh-En, we use MT06 as the validation set and choose MT02, MT03, MT04, MT05, and MT08 as the test sets. For all datasets, we use Berkeley Parser (Kitaev and Klein, 2018; Kitaev et al., 2019) to obtain constituency parse trees and use the most frequent constituents (e.g., <NP>, <VP>, <PP> and <S>) for syntactic guidance.

Model Settings For Transformer baselines, we adopt the Transformer_Base configuration which consists of a 6-layer encoder and decoder. For our model, we keep the 6-layer source context encoder, and set the number of layers for both the syntax context encoder and the decoder as 3, resulting in a similar model size with Transformer_Base. The accumulation weight α is as 0.8 for structural beam search based on validation experiments. For machine translation, we adopt sequence-level distillation (Kim and Rush, 2016) for both our model and the corresponding baseline Transformer. More details are shown in Appendix B.

Evaluation We use the BLEU score (Papineni et al., 2002) to evaluate machine translation performance. For paraphrase generation, we also adopt ROUGE (Lin, 2004) and METEOR (Banerjee and Lavie, 2005) as reference-based metrics. Besides,

Model	BLEU↑ / self-BLEU↓ / iBLEU↑	METEOR↑	ROUGE-1/2/L↑	$\mathbf{D}_{lex}$↑	$\mathbf{D}_{syn}$ ↑
Copy	18.5 / 100 / -17.1	28.8	50.6 / 23.2 / 47.7	0.0	0.0
Gold	100.0 / 18.6 / 64.4	100.0	100.0 /100.0 / 100.0	20.7	32.6
without Syntax Control					
SCPN (Iyyer et al., 2018)	12.1 / - / -	23.3	35.7 / 15.1 / 32.9	-	-
AESOP (Sun et al., 2021)	15.0 / - / -	26.1	47.0 / 21.3 / 47.3	-	-
Transformer (beam 1)	15.2 / 28.2 / 2.2	29.5	49.8 / 23.6 / 49.2	17.4	19.8
Our Method (beam 1)	18.6 / **15.2** / 8.5	30.8	51.1 / 26.3 / 51.3	**21.6**	24.4
Transformer (beam 5)	17.6 / 33.8 / 2.2	31.1	**51.9** / 26.0 / 51.0	16.2	18.1
Our Method (beam 5)	**19.3** / 16.4 / **8.6**	**31.5**	51.8 / **27.0** / **52.2**	21.5	**25.1**
with Human-annotated Syntax Control					
CGEN (Chen et al., 2019)	13.6 /- /-	24.8	44.8 / 21.0 / 48.3	-	-
SGCP-F (Kumar et al., 2020)	15.3 / - / -	25.9	46.6 / 21.8 / 49.7	-	-
SGCP-R (Kumar et al., 2020)	16.4 / - / -	28.8	49.4 / 22.9 / 50.3	-	-
AESOP-F (Sun et al., 2021)	20.4 / - /-	30.0	52.0 / 27.8 / **55.3**	-	-
Our Method	**20.9** / **10.5** / **13.0**	**33.3**	**54.1** / **29.7** / **55.3**	**22.6**	**27.7**

Table 1: Experimental results on paraphrase generation (ParaNMT-small).

Model	NIST Zh-En						WMT16	WMT14	
	MT02	MT03	MT04	MT05	MT08	avg	Ro-En	En-De	De-En
Transformer (beam 1)	48.9	49.2	50.7	49.3	41.4	47.9	33.9	27.9	30.7
Our Method (beam 1)	50.8	51.8	51.9	51.7	42.2	49.7	34.4	28.6	31.8
Transformer (beam 5)	49.8	50.1	51.1	50.1	42.3	48.7	34.1	28.3	31.3
Our Method (beam 5)	**51.1**	**52.4**	**52.4**	**52.1**	**43.1**	**50.2**	**34.9**	**28.7**	**32.2**

Table 2: Experimental results (BLEU score) on machine translation benchmark datasets. The result of our method is statistically significant compared to the corresponding Transformer baseline with $p < 0.05$ (Koehn, 2004).

Model	iBLEU↑	$\mathbf{D}_{lex}$↑	$\mathbf{D}_{syn}$↑
BART	4.4	19.6	24.4
BART + Our Method	**8.8**	**21.3**	**24.7**

Table 3: Results for training on BART, compared with sequence-to-sequence BART for paraphrase generation.

we report iBLEU (Sun and Zhou, 2012):

$$\text{iBLEU} = r \cdot \text{BLEU}(\text{hypothesis}, \text{reference}) - (1-r) \cdot \text{BLEU}(\text{hypothesis}, \text{source}),$$

which evaluates the generation fidelity with novelty to the source sentence considered*. Following Bandel et al. (2022), we consider two reference-free metrics: (1) lexical diversity score, i.e., $\mathbf{D}_{lex}$, which is the normalized character-level minima edit distance between the bag-of-words; and (2) syntax diversity score, i.e., $\mathbf{D}_{syn}$, which is the normalized tree edit distance. Both scores measure generated paraphrases with the source sequences unless specified.

5 Results

Paraphrase We compare our method with the baselines and previous work on syntax-control paraphrase generation. Another two baselines are also listed, i.e., copy the source input and use the reference as the output. The results are shown in Table 1. For paraphrase generation **without syntax control** (the center section in Table 1), our method achieves higher performance than the seq2seq Transformer, in both greedy and beam search settings. Typically, our method under greedy decoding obtains comparable results with the Transformer under beam search, and even outperforms under some metrics. The advantage of our method becomes larger for metrics such as iBLEU, $\mathbf{D}_{lex}$, and $\mathbf{D}_{syn}$, which consider generation novelty compared with the source input. For example, compared with Transformer (beam 5), our method (beam 5) gives a much lower self-BLEU score (**16.4** v.s. **33.8**) and higher diversity scores (**21.5** v.s. **16.2** for lexical diversity and **25.1** v.s. **18.1** for syntax diversity), indicating better generation diversity and contributing to a significant improvement on iBLEU (**8.6** v.s. **2.2**). **With annotated exemplars** (the lower section in Table 1), our model obtains further improvement over the non-exemplar setting and achieves better performance compared to previous work which utilizes full syntactic parse.

We extend our method to the **pre-trained language model** (PLM) setting and present the result in Table 3 (Details in Appendix A). It can be seen from the table that the utilization of BART (Lewis

*r is set as 0.7.

et al., 2019) improves the generation diversity for the sequence-to-sequence model significantly. Despite the narrowed gap, our model outperforms the seq2seq counterpart in terms of iBLEU and lexical diversity by a considerable margin.

Machine Translation As shown in Table 2, our method achieves consistent performance (BLEU score) improvement over the Transformer baseline. The improvement is larger for the greedy setting (+1.5 BLEU scores on average), compared with the beam search setting (+1.2). This indicates that using syntax to guide and constrain generation yields more reasonable and high-quality hypotheses than the greedy autoregressive generation, and thus relies less on search algorithms (e.g., beam search). Note that compared with the English-oriented datasets, our model obtains a smaller performance improvement on WMT'14 En-De. This can be because the German parser is less accurate than the English one (92.1 v.s. 96.3 for F1 score), resulting in a training set with lower quality.

6 Analysis

We first discuss the influence of grammar quality, then we understand the potential advantages of our method from three perspectives, i.e., interpretability, controllability, and diversity.

6.1 The Influence of Grammar Quality

Intuitively, learning syntactic grammar of higher quality results in better generation performance, e.g., the advantage of our method on English-oriented datasets is larger than the German-oriented one. To further explore the influence of grammar quality, we randomly replace a certain ratio of the constituent labels with a random one to simulate a less accurate parser. We conduct experiments on the WMT'16 Ro-En dataset. By injecting noise of ratios of **0.2** and **0.4**, the model performance deteriorates from 34.9 to **34.6** and **32.3** accordingly, indicating the quality of syntactic grammar exerts a large influence on model's generation performance.

6.2 Interpretability

We evaluate the model's interpretability based on its capability of providing explanations in understandable terms to a human (Doshi-Velez and Kim, 2017), i.e., whether it generates texts following language grammar. We trace each constituent expansion during generation and compare the model-induced tree with the tree parsed by a benchmark parser, e.g., Berkeley Parser. Specifically, we use the Berkeley parser to parse the same generated hypotheses by our model and treat the corresponding parsing results as golden parses. Quantitative results (Figure 4) show that our model achieves an average F1 score of **94.6** , which demonstrates the generation process highly corresponds to the syntactic grammar and thus can be effectively interpreted. Note that the score for WMT'14 En-De is lower (89.0), possibly due to the less accurate German parser for constructing the syntactic grammar, as discussed in Section 6.1.

Dataset	Precision	Recall	F1 Score
ParaNMT-small	96.0%	98.4 %	97.2%
NIST Zh-En	96.6%	96.8%	96.7%
WMT'16 Ro-En	93.5%	94.2%	93.9%
WMT'14 De-En	95.7%	96.3%	96.0%
WMT'14 En-De	84.4%	95.4%	89.6%

Table 4: The quantitative evaluation of the models' interpretability.

Dataset	BLEU ↑		$\mathbf{D}_{syn}^{ref}$ ↓	
	w/o	w	w/o	w
ParaNMT-small	19.3	24.9(+5.6)	25.7	17.2(-8.5)
NIST (ref-0)	28.0	30.3(+2.3)	25.1	19.2(-5.9)
NIST (ref-1)	27.3	29.3(+2.0)	25.5	20.1(-5.4)
NIST (ref-2)	25.7	28.5(+2.8)	25.4	18.3(-7.1)
NIST (ref-3)	26.1	28.1(+2.0)	25.7	20.1(-5.6)
WMT'16 Ro-En	35.0	35.8(+0.8)	18.3	15.9(-2.4)
WMT'14 De-En	32.2	35.3(+3.1)	19.6	14.0(-5.6)
WMT'14 En-De	28.7	30.6(+1.9)	28.9	26.3(-2.6)

Table 5: Controllable generation using golden syntax exemplars. NIST (ref-i) denotes the merged test sets with the i-th reference. A lower $\mathbf{D}_{syn}^{ref}$ denotes higher syntactic similarity with the reference.

6.3 Controllability

Control with Complete Syntax Template To leverage control signals from delexicalized syntax templates (e.g., "(S (NP) (VP (NP)))" for the sequence "I ate an apple."), we introduce a reward γ into Equation 13:

$$\delta_{ik+j} = \alpha\delta_{\boldsymbol{s}_i} + (1-\alpha)\delta_{\boldsymbol{f}_{j;\boldsymbol{s}_i}} + \gamma. \qquad (14)$$

If the updated syntax context $\boldsymbol{s}_{ik+j}$ matches the corresponding template pattern at depth $d+1$, the γ is a positive value otherwise 0. For example, the syntax context "<NP> <VP>" in Figure 3 matches the pattern "((NP)(VP))" at depth 2. Intuitively, the reward encourages the model to favor beam candidates that match the syntax template. We set the reward value as 0.32 based on validation results

(Appendix F). The testset of ParaNMT-small is provided with human-annotated exemplars and we use it to control generation, with results shown in Table 1. More generally, golden templates can be derived by parsing the reference sentences for each dataset with a parser (e.g., the Berkeley Parser). We present the results in Table 5. Guided by the reference syntax template, our model obtains consistent improvement in terms of hypothesis similarity with references, which is reflected by the decreased syntax edit distance to the references, i.e., $\mathbf{D}_{syn}^{ref}$. For the multi-reference dataset NIST Zh-En, our model can generate translations of different styles which are prompted by alternative syntax templates from multiple references.

Control with Partial Syntax Template We further explore whether the model can handle fine-grained arbitrary controls. Specifically, we ask three annotators to modify the intermediate syntax contexts output by the model, based on the source input. 100 instances are randomly selected from the NIST Zh-En test set and each annotator gives different modifications for each instance. The modified contexts are fed to the model to predict the infilling texts. We then ask the annotators to evaluate whether their controls (i.e., modifications) are safely responded to by the model. We show some of the control examples in Appendix G. The average control success rate is 81%, which demonstrates the capability of our model to handle arbitrary fine-grained controls.

6.4 Diversity

Beam Diversity We expect the model to generate diverse hypotheses under beam search, while also maintaining generation quality. To this end, we measure the model's beam diversity by computing two average scores: (1) the average of the mutual diversity scores of every two of the beam candidates, i.e., $\mathbf{D}_{lex}^{beam}$ and $\mathbf{D}_{syn}^{beam}$; (2) the average generation quality of the beam candidates, measured by BLEU scores. The results for paraphrase generation are shown in Table 6. In terms of generation quality, our model generates consistently better beam candidates on average than the baseline model. Besides, we can see that structural beam search can yield more diverse beam candidates, indicated by the higher mutual diversity (i.e., $\mathbf{D}_{lex}^{beam}$ and $\mathbf{D}_{syn}^{beam}$) among beam candidates.

Effects of Accumulation Weight A larger accumulation weight (α in Eq. 13) indicates a larger weight on previous decisions when re-ranking the newly updated beam candidates. As a result, early determined syntax structures are less likely to be surpassed throughout the whole structural beam search. On the contrary, a smaller α encourages the model to explore promising candidates at higher levels, and can therefore find more diverse hypotheses. We explore the effects of α with results shown in Figure 4. As the weight grows smaller, the model generates sequences of better syntactic diversity, i.e., $\mathbf{D}_{syn}$. However, an overly small weight deteriorates generation quality (iBLEU), which can be caused by the model's overconfidence in local predictions without considering the predictions of syntax contexts at lower levels. Such deterioration is also seen for overly large weights (>0.95), due to limited exploration at higher levels.

Model	ParaNMT-small avg BLEU/iBLEU	$\mathbf{D}_{lex}^{beam}\uparrow$	$\mathbf{D}_{syn}^{beam}\uparrow$
Transformer	15.0/1.6	12.6	11.2
Our Method	**16.9/7.1**	**15.0**	**12.6**

Table 6: Beam diversity measured by the average generation quality and the average mutual diversity among the beam candidates.

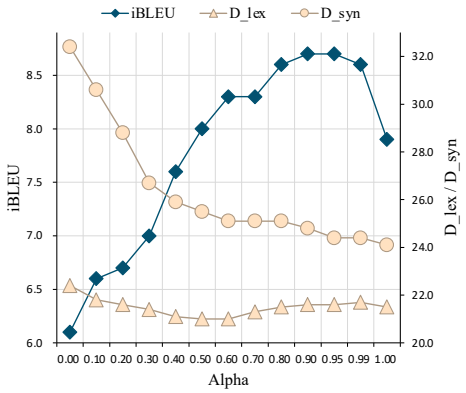

Figure 4: Effects of accumulation weights.

Human Evaluation We further conduct a human evaluation to evaluate generation quality and diversity on paraphrase generation. We ask three annotators to vote for one of the two candidates: hypotheses from the seq2seq baseline and our method. The annotators are required to decide, which one is better by considering *Fidelity*, *Novelty*, and *Diversity* (See Appendix H for details). The results are shown in Table 7. As can be seen from the table, our method achieves much better generation novelty and beam diversity compared with the baseline, while maintaining semantic fidelity, which further

Model	Fidelity	Novelty	Diversity
Transformer	50.2%	29.6 %	29.0%
Our Method	49.8%	**70.4%**	**71.0%**

Table 7: Human evaluation on paraphrase generation.

validates the results of the automatic evaluation.

7 Conclusion

We proposed a syntax-guided generation paradigm, which leverages the strengths of Transformer and generates sequences by hierarchically expanding constituents in the lexicalized syntax contexts throughout the syntax tree. The neural decoder was trained by maximizing the likelihood of the infilling texts for each constituent in the syntax contexts given the source sequence. Moreover, we proposed the structural beam search to better explore the hypothesis space. Empirical results demonstrated the advantage of generation quality over the seq2seq baseline, and also the effectiveness in terms of interpretability, controllability, and diversity.

Our method can be seen as a step towards explicit modelling of psycholinguistic structures during neural text generation , helping the model to have a degree of control over what it intends to generate, which can potentially address salient issues of current neural NLG, such as hallucination (Guerreiro et al., 2023; Dziri et al., 2022) and ethical issues (Sheng et al., 2019, 2021; Weidinger et al., 2021), if semantics, pragmatics, and other factors are also integrated.

Limitations

Despite the competitive performance, there are several limitations of this work: (1) As discussed in Section 6.1, the generation performance relies on the parser performance, which is strong enough for English but still less satisfactory for other languages. Dedicated methods need to be considered to compensate for the weak parser performance if we want to extend our method to more languages. (2) In this work, we consider two NLG tasks with semantic equivalence to testify if the proposed method can convey the source semantics accurately by following the target syntactic grammar. Other tasks such as summarization and dialogue generation can also be tested, where the semantics are not equivalent between the source and target. (3) To train the neural decoder parallelly, we break down the source-target dataset into a triple set. However, the global dependency of the syntax parse tree is not considered, which can deteriorate generation performance. (4) Due to the recursive encoding of the syntax contexts, our model's inference speed is approximately half that of the seq2seq counterpart (Appendix E). (5) Future work should include experiments on large language models (Brown et al., 2020; OpenAI, 2023; Zeng et al., 2022; Touvron et al., 2023; Taori et al., 2023). to further demonstrate the effectiveness of our method beyond pretrained language models.

Ethics Statement

We honor the ACL Code of Ethics. No private data or non-public information is used in this work. For human annotation (Section 6.3 and Section 6.4), we recruited our annotators from the linguistics departments of local universities through public advertisement with a specified pay rate. All of our annotators are senior undergraduate students or graduate students in linguistic majors who took this annotation as a part-time job. We pay them 60 CNY an hour. The local minimum salary in the year 2022 is 25.3 CNY per hour for part-time jobs. The annotation does not involve any personally sensitive information. The annotated is required to rank the system output and label factual information (i.e., syntactic annotation).

Acknowledgement

We would like to thank all reviewers for their insightful comments and suggestions to help improve the paper. We thank Deng Cai and Xinting Huang for their insightful suggestions. This work is funded by the Ministry of Science and Technology of China (grant No. 2022YFE020038).

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140, Vancouver, Canada. Association for Computational Linguistics.

David Alvarez-Melis and Tommi Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 412–421, Copenhagen, Denmark. Association for Computational Linguistics.

Elron Bandel, Ranit Aharonov, Michal Shmueli-Scheuer, Ilya Shnayderman, Noam Slonim, and Liat Ein-Dor. 2022. Quality controlled paraphrase generation. *CoRR*, abs/2203.10940.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2018. Syntax-directed attention for neural machine translation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy. Association for Computational Linguistics.

Noam Chomsky. 1965. Aspects of the theory of syntax.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bert look at? an analysis of bert's attention. In *BlackBoxNLP@ACL*.

Verna Dankers, Elia Bruni, and Dieuwke Hupkes. 2022. The paradox of the compositionality of natural language: A neural machine translation case study. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 4154–4175. Association for Computational Linguistics.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv: Machine Learning*.

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.

Nouha Dziri, Sivan Milton, Mo Yu, Osmar R. Zaïane, and Siva Reddy. 2022. On the origin of hallucinations in conversational models: Is it the datasets or the models? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5271–5285. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2019. Incorporating source-side phrase structures into neural machine translation. *Computational Linguistics*, 45(2):267–292.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. *arXiv preprint arXiv:1702.03525*.

Nuno Miguel Guerreiro, Elena Voita, and André F. T. Martins. 2023. Looking for a needle in a haystack: A comprehensive study of hallucinations in neural machine translation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 1059–1075. Association for Computational Linguistics.

Shilin He, Zhaopeng Tu, Xing Wang, Longyue Wang, Michael Lyu, and Shuming Shi. 2019. Towards understanding neural machine translation with word importance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 953–962, Hong Kong, China. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1317–1327. The Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 388–395. ACL.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. 2020. Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics*, 8:329–345.

An Nguyen Le, Ander Martinez, Akifumi Yoshimoto, and Yuji Matsumoto. 2017. Improving sequence to sequence neural machine translation by utilizing syntactic dependency information. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 21–29.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697, Vancouver, Canada. 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4767–4780. Association for Computational Linguistics.

Yangming Li and Kaisheng Yao. 2021. Interpretable NLG for task-oriented dialogue systems with heterogeneous rendering machines. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13306–13314. AAAI Press.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 163–169. Association for Computational Linguistics.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2019. Improving neural machine translation with neural syntactic distance. 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 2032–2037.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2020. Syntax-based transformer for neural machine translation. *Journal of Natural Language Processing*, 27(2):445–466.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Tiejun Zhao, and Eiichiro Sumita. 2018. Forest-based neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1253–1263.

Richard Montague. 1974. Universal grammar. In Richmond H. Thomason, editor, *Formal Philosophy: Selected Papers of Richard Montague*, 222–247. Yale University Press, New Haven, London.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Ru Peng, Zhitao Chen, Tianyong Hao, and Yi Fang. 2019. Neural machine translation with attention based on a new syntactic branch distance. In *China Conference on Machine Translation*, pages 47–57. Springer.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Emily Sheng, Kai-Wei Chang, Prem Natarajan, and Nanyun Peng. 2021. Societal biases in language generation: Progress and challenges. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4275–4293. Association for Computational Linguistics.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3405–3410. Association for Computational Linguistics.

Hong Sun and Ming Zhou. 2012. Joint learning of a dual SMT system for paraphrase generation. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers*, pages 38–42. The Association for Computer Linguistics.

Jiao Sun, Xuezhe Ma, and Nanyun Peng. 2021. AESOP: Paraphrase generation with adaptive syntactic control. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5176–5189, Online and Punta Cana, Dominican Republic. 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.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. 2018. A tree-based decoder for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4772–4777, Brussels, Belgium. Association for Computational Linguistics.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2021. Ethical and social risks of harm from language models. *CoRR*, abs/2112.04359.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 698–707, Vancouver, Canada. Association for Computational Linguistics.

Yikuan Xie, Wenyong Wang, Mingqian Du, and Qing He. 2021. Transformer with syntactic position encoding for machine translation. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1536–1544.

Baosong Yang, Derek F Wong, Lidia S Chao, and Min Zhang. 2020a. Improving tree-based neural machine translation with dynamic lexicalized dependency encoding. *Knowledge-Based Systems*, 188:105042.

Jian Yang, Shuming Ma, Dongdong Zhang, Zhoujun Li, and Ming Zhou. 2020b. Improving neural machine translation with soft template prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5979–5989.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Peng Zhang, Yuxiao Dong, and Jie Tang. 2022. Glm-130b: An open bilingual pre-trained model.

AK Zholkovskii and IA Mel'chuk. 1965. On a possible method and instrument for semantic synthesis. *Nauchno-tekhnicheskaya informatsiya,(6)*.

Algorithm 1 Structural beam search

Setup: k: beam size
α: accumulation weight
d_{max}: maximum tree depth
ENCODER$(\cdot)$: source context encoder
terminated$(\cdot)$: termination examination function
expand$(\cdot,\cdot)$: constituent expansion function
beam_search$(\cdot,\cdot)$: standard beam search algorithm
Input: $\mathbf{x}$: source sequence

$d \leftarrow 0$
$\mathbf{h}_{src} \leftarrow$ ENCODER$(\mathbf{x})$
$B_0 \leftarrow \{(0, \langle T \rangle)\}$
while $d < d_{max}$ **do**
  $B \leftarrow \varnothing$
  for $(\delta_{\boldsymbol{s}}, \boldsymbol{s}) \in B_{d-1}$ **do**
    if terminated$(\boldsymbol{s})$ **then**
      B.add$((\delta_{\boldsymbol{s}}, \boldsymbol{s}))$
      continue
    end if
    $\mathcal{F} \leftarrow$ beam_search$(\boldsymbol{s}, \mathbf{h}_{\text{src}})$
    for $(\delta_{\boldsymbol{f}}, \boldsymbol{f}) \in \mathcal{F}$ **do**
      $\hat{\delta} \leftarrow \alpha\delta_{\boldsymbol{s}} + (1-\alpha)\delta_{\boldsymbol{f}}$
      $\hat{\boldsymbol{s}} \leftarrow$ expand$(\boldsymbol{s}, \boldsymbol{f})$
      B.add$((\hat{\delta}, \hat{\boldsymbol{s}}))$
    end for
  end for
  $B_d \leftarrow B$.top(k)
  $d \leftarrow d + 1$
end while
return $B_{d_{max}}$

Algorithm 2 Beam search

Setup: k: beam size
t_{max}: maximum hypothesis length
$\mathcal{V}$: target tokens set
score$(\cdot,\cdot,\cdot)$: scoring function (Eq. 15)
Input: $\boldsymbol{s}$: syntax context
$\mathbf{h}_{src}$: source context representations

$t \leftarrow 0$
$B_0 \leftarrow \{(0, \langle bos \rangle)\}$
while $t < t_{max}$ **do**
  $B \leftarrow \varnothing$
  for $(\delta, \boldsymbol{f}) \in B_{t-1}$ **do**
    if $\boldsymbol{f}.last() = \langle eos \rangle$ **then**
      B.add$((\delta, \boldsymbol{f}))$
      continue
    end if
    for $f \in \mathcal{V}$ **do**
      $\delta \leftarrow$ score$(\mathbf{h}_{src}, \boldsymbol{s}, \boldsymbol{f} \circ f)$
      B.add$((\delta, \boldsymbol{f}))$
    end for
  end for
  $B_t \leftarrow B$.top(k)
  $t \leftarrow t + 1$
end while
return $B_{t_{max}}$

A Algorithms

The scoring algorithm 7 can be rewritten with the source context $\mathbf{x}$ encoded into $\mathbf{h}_{src}$:

$$\text{score}(\mathbf{h}_{src}, \boldsymbol{s}, \boldsymbol{f}) = \sum_{t=0}^{|\boldsymbol{f}|} logp_{\boldsymbol{\theta}}(f_t | \mathbf{h}_{src}, \boldsymbol{s}, f_{1:t-1}) \tag{15}$$

The algorithm of **structural beam search** is demonstrated in Algorithm 1, which employs the standard beam search for autoregressive generation, depicted in Algorithm 2. The termination function in Algorithm 1 (i.e., terminated$(\cdot)$) returns true if the there is no remaining constituent in the input sequence.

B Experiment Details

For NIST Zh-En, we use parts of the bitext provided within NIST'12 OpenMT[†] and the final train set consists of about 1.8M sentence pairs. We apply BPE (Sennrich et al., 2016) on all datasets: the number of BPE operations is 6K for ParaNMT-small, and 40K for the other datasets. We implement our model using Fairseq (Ott et al., 2019).

[†]LDC2005T06, LDC2004T07, LDC2003E07, LDC2000T46, LDC2000T47, LDC2000T50, LDC2003E14, LDC2005T10, LDC2002E18, LDC2007T09, LDC2004T08

We train the model using Adam (Kingma and Ba, 2015) optimizer. The learning rate increases to $7 \cdot 10^{-4}$ in the first 10K steps and then anneals exponentially. We set the weight decay as 0.01 and label smoothing as 0.1. The dropout is 0.3 for ParaNMT-small, and 0.1 for the other datasets. The batch size is 64K tokens for ParaNMT-small, 256K for WMT'16 Ro-En and NIST Zh-En, and 512K for WMT'14 De↔En. All models are trained for a maximum update of 300K steps unless early stopped. We train the model using 4 V100s and increase gradient accumulation steps for large batch sizes. We choose the 5 best checkpoints based on validation sets and average them for inference. We set the beam width as 5 for beam search. For machine translation, the teacher models for knowledge distillation are Transformer_Base for NIST Zh-En and WMT'16 Ro-en, and Transformer_Big for WMT'14 De↔En.

C Model Architecture

We conduct experiments to compare different model architectures to incorporate syntax context on the WMT'16 Ro-En validation set. We consider the following settings:

- *Concat*: concatenate the syntax context with the source sequence, with the vanilla Transformer unmodified.
- *Extra-attention*: reuse the source encoder for encoding syntax context and insert an extra at-

Model	BLEU↑ / self-BLEU↓ / iBLEU↑	METEOR↑	ROUGE-1/2/L↑	D_{lex}↑	D_{syn}↑
BART Seq2seq (beam 1)	15.8 / 26.9 / 3.0	27.3	50.1 / 23.1 / 50.0	19.5	23.8
BART + Our Method (beam 1)	18.3 / 15.5 / 8.2	31.0	52.1 / 26.7 / 52.1	21.1	24.0
BART Seq2seq (beam 5)	17.9 / 27.0 / 4.4	28.4	51.4 / 24.8 / 51.5	19.6	24.4
BART + Our Method (beam 5)	**19.0 / 15.1 / 8.8**	**31.3**	**52.3 / 27.0 / 52.5**	**21.3**	**24.7**

Table 8: Experimental results on paraphrase generation (ParaNMT-small) based on BART.

Architecture	# params	BLEU	Speed
Concat	64.2M	34.5	1.0x
Extra-attention	70.5M	34.7	0.9x
Extra-encoder	64.2M	**35.3**	**1.1x**

Table 9: Model architectures for encoding previous syntax contexts.

tention layer, i.e., the syntax context attention, into each decoder layer.

- *Extra-encoder*: introduce an additional encoder for encoding syntax context and also uses the syntax context attention.

Empirical results are shown in Table 9. Based on validation results, we adopt the *Extra-encoder* model in all experiments except for training on BART (Table 3), where we adopt the *Concat* model.

D Experiments on PLM

In this section, we introduce our experiment settings of PLM. Following previous work (Sun et al., 2021), we use BART-base (Lewis et al., 2019) as our base model. All models are finetuned for 10 epochs with a batch size of 64k tokens. The learning rate is 3e-5 and the linear decay schedule, as recommended in BART's official repository[‡].

We use the *Concat* (Appendix C) model architecture for extending our method to BART. The source text and the syntax context are concatenated with a special token "<sep>", e.g., "I ate an apple . <sep> <NP> <VP> .". To effectively employ our method with BART, whose inputs are tokenized sequences byte-level, as same as Radford et al., we make several modifications. In the pre-processing, we make sure our special tokens (e.g., <sep>, <c>, <NP>, <VP>) are not split and add extra byte-level spaces before and after the special token. Thanks to the unused tokens in BART embeddings, we do not need to modify the embedding matrix. Instead, we assign our special tokens to unused token indexes.

[‡]https://github.com/facebookresearch/fairseq/tree/main/examples/bart

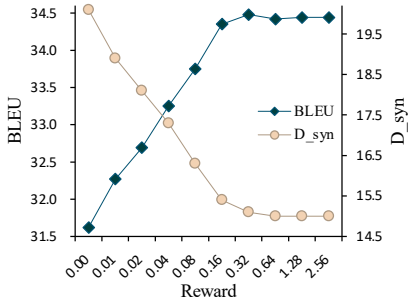

Figure 5: Effects of reward ratio on the WMT14'De-En validation set.

Finally, in the inference stage, we find the constituency expansion causes a discrepancy between inputs of train and test. Thus, we first detokenize each layer's outputs and then tokenize them back with the same procedure in the preprocessing to avoid such a gap.

E Generating Linearized Trees Directly

A baseline method to induce grammar simultaneously during generation is generating linearized parse trees directly, i.e., training a seq2seq model which takes in source sequences and outputs linearized parse trees. We compare it with our method on WMT'16 Ro-En. Specifically, the BLEU score for WMT'16 Ro-En is only **27.6** compared to the seq2seq baseline (**34.1**) and our method (**34.9**). This can be because the additional parentheses and constituency tags in linearized trees may deteriorate sequence coherence, making learning more difficult. Our method, on the other hand, breaks down syntax trees into level pieces to create a better learning curriculum. Furthermore, Generating linearized parse trees is much slower than the seq2seq counterpart, since the average sequence length of linearized tree sequences is longer (152.3 vs 28.4). As a result, the average speed for generating linearized parse trees is only 0.8 sentences/s compared to 3.6 sentences/s for the seq2seq baseline. Our method achieves an inference speed of 1.7 sentences/s under the same computing condition (V100 GPU). Additionally, generating a linearized parse tree is not easily interpretable or controllable, due to the black-box nature of the sequence-to-

Source	Human Control	Infilling Text	Final Hypothesis
巴基斯坦 政府 和 人民 对 死难者 的 家属 表示 深切 的 慰问 。 (English: The Government and people of Pakistan express their deep sympathy to the bereaved families.)	<NP> and <NP> expressed their deep sympathy <PP> .	<c> the pakistani government <c> the pakistani people <c> for <NP>	the pakistani government and the pakistani people expressed their deep sympathy for the families of the victims .
	<S> <VP> to the bereaved family .	<c> the pakistani government and people <c> expressed <NP>	the pakistani government and people expressed their deep sympathy and solicitude to the bereaved family .
	the government and people of pakistan <VP> .	<c> expressed <NP>	the government and people of pakistan expressed their deep sympathy and solicitude for the families of the victims .
老实 说 , 我 认为 自己 要 比 36 岁 年轻 许多 。 (English: To be honest, I consider myself much younger than 36.)	<PP> , <S> .	<c> in <NP> <c> <NP> <VP>	in an honest way , i think i am much younger than 36 .
	to be honest , I consider <S> .	<c> <NP> much younger <PP>	to be honest , i consider myself much younger than 36 .
	to be honest , <S> .	<c> <NP> <VP>	to be honest , i think i am much younger than 36 .
然而 , 这 并 不 妨碍 哈马斯 作 出 灵活 的 策略 调整 , 推选 独立 人士 便是 折中 之 策 。 (English: That, however, does not prevent Hamas from manoeuvring nimbly. Voting for an independent would be a compromise.)	that , however , does not <VP> , voting for an independent would be a compromise .	<c> prevent <NP> <PP>	that , however , does not prevent hamas from making flexible strategic adjustments , voting for an independent would be a compromise .
	that , however , <VP> . <VP> would be a compromise .	<c> does not <VP> <c> electing <NP>	that , however , does not prevent hamas from making flexible strategic adjustments . electing an independent person would be a compromise .
	<S> , <S>	<c> however , <NP> <VP> <c> <S> <VP>	however , this does not prevent hamas from making flexible strategic adjustments , choosing an independent person is a compromise

Figure 6: Samples cases for fine-grained manual controls: the 4 columns denote the source Chinese sentence, the human-annotated control, the model's predicted infilling texts, and the final English translation.

sequence paradigm.

F Effects of Control Reward

The magnitude of the reward γ determines how much priority is given to beam candidates that match the syntax exemplar. We experiment with different reward values to give a quantitative demonstration, shown in Figure 5. It can be seen that the control effectiveness grows with the increase of the reward value until 0.64, which suggests that all possible matched beam candidates are re-ranked to the top in the search space.

G Control with Partial Syntax Template

We present 3 sample cases to demonstrate fine-grained controls over the generation process, shown in Figure 6. Each Chinese source sentence is paired with 3 manual controls from three annotators. The model takes in the annotated syntax context and proceeds to obtain the respective translations.

H Human Evaluation for Paraphrase Generation

We ask three annotators to conduct side-by-side human evaluations and report averaged results of their annotations. For each instance, the annotators vote for one of the two outputs by the baseline and our model. The outputs contain top-5 beam candidates under beam search. The annotators are asked to evaluate both the best candidate and the beam results as a whole, based on the following three aspects:

- Fidelity: Whether the best candidate is semantics-equivalent with the input.
- Novelty: Whether the best candidate modifies the input sentence structure.
- Diversity: Whether the generated five candidates are different from each other given the input.

ACL 2023 Responsible NLP Checklist

A For every submission:

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

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

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

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

B ☐ Did you use or create scientific artifacts?

Not applicable. Left blank.

☑ B1. Did you cite the creators of artifacts you used?
Section 4.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ 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)?
Not applicable. Left blank.

☐ 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?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ 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.
Section 4.

C ☑ Did you run computational experiments?

Section 4 & 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?
Appendix B & C.

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?
Section 4; Appendix B & C.

☐ 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?
Not applicable. Left blank.

☑ 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.)?
Section 5.

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

Section 6.4 & 6.4.

☑ 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.?
Appendix G & H.

☑ 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)?
Section Ethics Consideration.

☑ 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?
Appendix G & H.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☑ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Section Ethics Consideration.