Parallel Refinements for Lexically Constrained Text Generation with BART

Xingwei He
Department of Electrical and Electronic Engineering,
The University of Hong Kong,
Hong Kong, China
hexingwei15@gmail.com

Abstract

Lexically constrained text generation aims to control the generated text by incorporating some pre-specified keywords into the output. Previous work injects lexical constraints into the output by controlling the decoding process or refining the candidate output iteratively, which tends to generate generic or ungrammatical sentences, and has high computational complexity. To address these challenges, we propose **C**onstrained **BART** (CBART) for lexically constrained text generation. CBART leverages the pre-trained model BART and transfers part of the generation burden from the decoder to the encoder by decomposing this task into two sub-tasks, thereby improving the sentence quality. Concretely, we extend BART by adding a token-level classifier over the encoder, aiming at instructing the decoder where to replace and insert. Guided by the encoder, the decoder refines multiple tokens of the input in one step by inserting tokens before specific positions and re-predicting tokens with low confidence. To further reduce the inference latency, the decoder predicts all tokens in parallel. Experiment results on One-Billion-Word and Yelp show that CBART can generate plausible text with high quality and diversity while significantly accelerating inference.

1 Introduction

Controllable text generation aims to generate text in a controlled way, such as transferring text style (Shen et al., 2017; Fu et al., 2018; Li et al., 2018; Xu et al., 2018) and generating text with control codes (Keskar et al., 2020). Lexically constrained text generation requires that the given keywords must appear in the output, which can be applied to incorporating keywords into a dialog response (Mou et al., 2016), creating a story with keywords (Fan et al., 2018), generating advertisements for products (Miao et al., 2019) and writing a concrete meeting summary based on several key phrases.

Cons	**lovely**, **time**, **forever**, **try**
Step 1	this **lovely** experience **time** and **forever** to **try** .
Step 2	this was **lovely** experience ! **time** and it **forever** to **try** this .
Step 3	this was a **lovely** experience ! first **time** here and it took **forever** to **try** this place .

Table 1: An example from the Yelp test set demonstrates how CBART generates text with lexical **cons**traints. In each refinement step, tokens underlined are newly generated.

To generate sentences with keywords, Mou et al. (2015) proposed a backward and forward language model (B/F-LM). Liu et al. (2019a) applied adversarial learning (Goodfellow et al., 2014) to B/F-LM. Both models are limited to generating text with one keyword. To incorporate multiple keywords into machine translation, Hokamp and Liu (2017) proposed grid beam search (GBS) by adding an additional constrained dimension to beam search. However, GBS does not consider the future lexical constraints when generating previous tokens, thereby degrading the quality of generated sentences.

Recently, Markov Chain Monte Carlo (MCMC) sampling has been applied to text generation (Berglund et al., 2015; Su et al., 2018; Devlin et al., 2019). Compared with GBS, MCMC-based models can iteratively refine tokens based on contexts. CGMH (Miao et al., 2019) uses Metropolis-Hastings sampling to generate constrained sentences with a series of actions, such as insertion, deletion, and replacement. In most cases, refinements conducted by CGMH are invalid because of the randomly chosen actions and positions. To solve this problem, the gradient information (Sha, 2020) and a token-level classifier (He and Li, 2021) are used to determine the position to be edited and the action to be taken. These models are computation-intensive as they can update only one token in each step. POINTER (Zhang et al., 2020b) reduces the inference latency by refining multiple tokens in one step. However, POINTER is based

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 8653–8666
November 7–11, 2021.

on BERT (Devlin et al., 2019) and imposes all the burden of generation on the decoder. Previous work (Lewis et al., 2020) has shown that BART is more suitable than BERT for text generation. Nevertheless, BART can not be directly applied to constrained text generation. If we feed the keywords into the encoder, the decoder will not guarantee that the output contains the given keywords.

To alleviate the above problems, we propose CBART, a parallel refinement model for lexically constrained text generation, shown in Figure 1. CBART benefits from the large-scale pre-trained model, i.e., BART. In addition, CBART shifts part of the decoder's burden to the encoder. Specifically, we put a token-level classifier over BART's encoder, in charge of analyzing the input and providing the decoder with coarse-grained modification information, such as the positions to be refined and the actions to be conducted. The refinement information provided by the encoder enables the decoder to revise multiple tokens of the input in one step to make the sentence more fluent, such as inserting missing tokens before certain positions and replacing inappropriate tokens with other tokens. In addition, the decoder predicts all tokens simultaneously, which further speeds up inference. As shown in Table 1, CBART keeps refining the generated text until completing it during inference.

Our work's main contributions are threefold: (1) We propose CBART[1] for lexically constrained text generation. The proposed model takes advantage of the pre-trained model, BART. Besides, we lighten the generation burden on the decoder by decomposing constrained sentence generation into two sub-tasks. Furthermore, CBART can insert and replace multiple tokens in each refinement step and predict all tokens in parallel. (2) To train CBART, we propose a new method to construct synthetic data. (3) Experiment results on One-Billion-Word and Yelp demonstrate that CBART outperforms previous work in terms of generation quality, generation diversity and inference speed.

2 Problem Definition

Lexically Constrained Text Generation aims to incorporate the given keywords into the generated text. Given a set of lexical constraints, $c_1, c_2, \ldots, c_k$, this task aims to find a fluent text

[1]Our code is available at https://github.com/NLPCode/CBART.

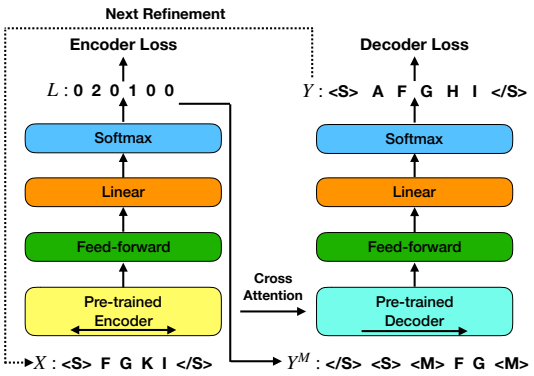

Figure 1: The overview of our proposed model. <S>, </S> and <M> represent the start, end, and mask tokens. The refinement process indicated by the dashed line only appears in inference. During training, the decoder input Y^M is created based on the gold encoder label sequence; while during inference, it is created based on the predicted encoder label sequence.

by maximizing the conditional probability:

$$X^* = \arg\max_X P(X|c_1, c_2, \ldots, c_k), \quad (1)$$

where X is the text containing the given keywords.

3 Methodology

In Section 3.1, we will first introduce the proposed model, CBART. Then, we will show how to create synthetic datasets to train CBART in Section 3.2. Finally, we will introduce several parallel decoding strategies for generating text, the repetition penalty mechanism used to discourage the generation of repetitive tokens, and the termination criterion for inference in Section 3.3.

3.1 Model Architecture and Training

The overview of our proposed model is demonstrated in Figure 1. The proposed model consists of two modules: an encoder E and a decoder D.

Encoder and Action Classifier. We use the pre-trained language model, BART, to initialize the proposed model. The encoder is responsible for providing coarse-grained refinement information for the decoder. In other words, the encoder is expected to instruct the decoder where to replace and insert. To this end, we add a fully connected feed-forward layer over the encoder. More concretely, the encoder takes an incomplete sequence as input and outputs a corresponding label sequence. Based on the label sequence, the decoder can be aware of how to refine the candidate sentence.

The encoder serves as a three-class token-level classifier, where we use labels 0, 1, 2 to refer to

copy, replacement, and insertion actions. The copy action means that the decoder should keep the current token. The replacement action suggests that the decoder should replace the current token with another token to make the text more coherent. Similarly, the insertion action indicates that the decoder should insert a token before the current token to complete the text. We use (X, L, Y^M, Y) to represent a training instance. $X = \{x_1, \ldots, x_n\}$ denotes the incomplete text fed into the encoder and $L = \{l_1, \ldots, l_n\}$ is the encoder label sequence. Then, the cross-entropy loss for the encoder is:

$$L_{encoder} = -\frac{1}{n}\sum_{t=1}^{n} \log p(l_t|x_1, \ldots, x_n). \quad (2)$$

Decoder. The decoder takes Y^M as input and aims to reconstruct the original text Y, where $Y^M = \{y_1^M, \ldots, y_m^M\}$ is constructed based on X and L, and $Y = \{y_1, \ldots, y_m\}$ is the decoder label sequence. Following BART, the decoder will predict the complete output Y rather than the masked tokens of Y^M during training. We optimize the decoder by minimizing the reconstruction loss:

$$L_{decoder} = -\frac{1}{m}\sum_{t=1}^{m} \log p(y_t|X, y_{<t}^M). \quad (3)$$

Joint Training. We jointly optimize the encoder and decoder by minimizing the total loss:

$$L_{total} = L_{encoder} + \alpha L_{decoder}, \quad (4)$$

where α is the trade-off parameter.

3.2 Creating the Synthetic Dataset

Before training CBART, we need to create the synthetic dataset $D = \{(X, L, Y^M, Y)\}$. Each time, we randomly choose a sentence from One-Billion-Word or Yelp to create the synthetic dataset. Suppose the selected text is "*<S> A B C D E F G H I </S>*", where <S> and </S> denote the start and end tokens. Next, we randomly select some tokens from it (e.g., "*<S> F G H I </S>*"). Then, we randomly replace 15% of tokens with other tokens (e.g., replace 'H' with 'K'). Therefore, we obtain the encoder input X ={<S>, F, G, K, I, </S>} and the corresponding encoder label sequence L ={0, 2, 0, 1, 0, 0}, where 2 denotes that a token should be inserted before 'F' and 1 means that 'K' should be replaced with another token. Further, we construct Y^M = {</S>, <S>, <M>, F, G, <M>, I } by inserting a special mask token <M> before 'F', replacing 'K' with <M> and shifting the sequence one position to the right.

The decoder label sequence Y is constructed by replacing the masked tokens of Y^M with gold tokens. It is trivial to decide the gold token for the replacement action. For example, since 'H' is replaced with 'K', the gold label for the second mask token <M> should be 'H'. However, it is challenging to determine the gold token for the insertion action, especially when multiple tokens are missing before a position. Since five tokens ('A','B','C','D' and 'E') before 'F' have been deleted from the original text, we need to decide which token should be inserted before 'F' first, which will be regarded as the gold token for the first mask token <M>.

We test five different ways (*Left*, *Middle*, *Right*, *Random* and *TF-IDF*) to construct synthetic datasets for the insertion action. The *left* method regards the leftmost token 'A' as the first inserted token before 'F', so we get Y ={ <S>, A, F, G, H, I, </S>}. Similarly, the *middle*, *right*, and *random* methods regard the middle token 'C', the rightmost token 'E', or a randomly chosen token 'D' as the first inserted token. We also consider the importance of tokens by computing their TF-IDF scores. Assume token 'B' has the highest TF-IDF value. It will be regarded as the first inserted token. We conduct an experiment to compare the effect of these methods in Section 4.4.

3.3 Inference

We set lexical constraints as the initial input of the encoder, X^0. We start inference by feeding X^0 into the encoder and obtain the predicted label sequence $\hat{L}^0$ with argmax decoding. Next, we construct Y^M based on X^0 and $\hat{L}^0$, and feed it into the decoder. Then, we run a decoding strategy on the decoder to get $\hat{Y}^0$, using $\hat{Y}^0$ as the encoder input of the next refinement step, X^1. We continue to refine the encoder input, until meeting a termination condition. Unlike training, we forbid replacing any keyword with the mask token <M> when constructing Y^M, and the decoder only needs to predict the masked tokens of Y^M to ensure the given keywords appear in the output. In the following, we will introduce greedy decoding for the encoder and four parallel decoding strategies for the decoder: greedy, top-k, top-p, and multiple-sequence decoding. Each decoding strategy allows the decoder to predict all masked tokens of Y^M in parallel.

Greedy Decoding for the Encoder. At the refine-

ment step r, the encoder takes X^r as input, and we choose the label with the highest probability as the predicted label $\hat{l}_t^r$ ($\hat{L}^r = \{\hat{l}_1^r, \ldots, \hat{l}_{n^r}^r\}$):

$$\hat{l}_t^r = \arg\max_{l_t^r} p(l_t^r | X^r). \tag{5}$$

Greedy Decoding. Similar to the method mentioned above, greedy decoding selects the token with the highest probability for position t as the decoder output ($\hat{Y}^r = \{\hat{y}_1^r, \ldots, \hat{y}_{m^r-1}^r, </S>\}$):

$$\hat{y}_t^r = \begin{cases} \arg\max p(y_t^r | X^r, y_{\leq t}^M), & y_{t+1}^M = <M> \\ y_{t+1}^M, & y_{t+1}^M != <M>. \end{cases} \tag{6}$$

Top-k and Top-p Decoding. Since maximization-based decoding, such as greedy decoding and beam search, may cause text degeneration, we use top-k (Fan et al., 2018; Holtzman et al., 2018) and top-p decoding (Holtzman et al., 2020) to alleviate this problem. For each position, top-k decoding samples a token from the k most probable tokens, rather than always choosing the most probable one. Similarly, top-p decoding samples a token from the smallest possible set of tokens, whose cumulative probability exceeds the probability p.

Multiple-sequence Decoding. Top-k or top-p decoding can generate more diverse text but risk producing low-quality sentences. To remedy this, we propose multiple-sequence decoding. When using top-k or top-p decoding to generate sentences, we run the decoding method for N times to get multiple sequences. Because all sequences are mutually independent, they can be decoded simultaneously. After obtaining multiple generated sentences, we resort to the pre-trained language model, the GPT-2 small model (Radford et al., 2019), to rank these sentences and choose the one with the lowest negative log-likelihood (NLL). The ranking operation also runs in a non-autoregressive way, thus avoiding excessive overhead.

Repetition Penalty. Even large well-trained generation models might generate repetitive phrases or sentences, resulting in a lower diversity of the generated text (Holtzman et al., 2020). We find the proposed model suffers from this issue more seriously as the masked tokens are predicted conditionally independently in each refinement step. To alleviate this problem, we resort to the repetition penalty strategy, which discounts the scores of previously generated tokens. Slightly different from Keskar et al. (2020), we discourage the generation of tokens appearing in Y^M instead of previously generated tokens, achieving the repetition penalty without hurting the non-autoregressive property. The probability distribution for the t-th token at the refinement step r is defined as follows:

$$p(\hat{y}_t^r = i) = \frac{exp(h_i / I(i \in Y^M))}{\sum_j exp(h_j / I(j \in Y^M))}, \tag{7}$$

where h_i is the logit for the t-th token. If c is true, $I(c)$ equals θ, otherwise equals 1.

Termination Criterion. During inference, we refine the output token by token. When should we stop refining? One method is to monitor the encoder. If all predicted encoder labels are 0, indicating no revision is required by any token, we will stop refining. However, this criterion is so strict that refinements may not stop in most cases. Therefore, we adopt a relatively loose standard by monitoring the output of the decoder. To be specific, if the decoder output is the same as that of the last refinement step, we will stop the refinement process.

4 Experiments

4.1 Experiment Setup

Datasets and Pre-processing. Following Miao et al. (2019) and Zhang et al. (2020b), we conduct experiments on One-Billion-Word[2] and the Yelp dataset[3]. One-Billion-Word is a public dataset for language modeling produced from the *WMT 2011 News Crawl data*. The Yelp dataset consists of business reviews on Yelp. For each dataset, we filter out sentences with length less than 10 or greater than 40. After preprocessing, we choose $1M$, $0.1M$ sentences from each dataset as the training and validation sets. We also select $1K$ sentences to provide keywords. To be specific, we extract 1-6 keywords from each sentence. Therefore, we construct six kinds of test sets for lexically constrained text generation, and the size of each test set is $1K$.

Baselines. We compare our proposed model with several strong baselines for lexically constrained text generation, including three traditional baselines (sep-B/F, asyn-B/F, and GBS) and three recent models (CGMH, POINTER, and X-MCMC-C). We implement two variants of the backward and forward language model (sep-B/F and asyn-B/F)

[2]http://www.statmt.org/lm-benchmark/

[3]https://www.yelp.com/dataset

(Mou et al., 2015), GBS (Hokamp and Liu, 2017) and CGMH (Miao et al., 2019). For a fair comparison, these baselines are based on the GPT-2 small model ($n_{layer} = 12$, $n_{head} = 12$, $d_{hidden} = 768$, and $117M$ parameters), which has a similar architecture to the decoder of BART-large.

X-MCMC-C benefits from the guidance of the XLNet-based classifier, thus substantially improving the generation quality compared to CGMH. We train X-MCMC-C with the code provided by He and Li (2021), which is based on the XLNet-base-cased model ($n_{layer} = 12$, $n_{head} = 12$, $d_{hidden} = 768$, and $110M$ parameters). We also compare our model with POINTER (Zhang et al., 2020b). Similar to our model, POINTER can insert multiple tokens in each step. We train two different POINTER models, POINTER and POINTER-2, with the code released by Zhang et al. (2020b). Specifically, POINTER is initialized with BERT-large, while POINTER-2 is initialized with the general model, pre-trained on the English Wikipedia dataset. Both models have comparable parameters ($n_{layer} = 24$, $n_{head} = 16$, $d_{hidden} = 1024$, and $336M$ parameters) to CBART.

Training and Inference. For our model, we create synthetic data with the *left* method. For each sentence, we create 10 synthetic data instances. Therefore, for each dataset, the size of the synthetic training and validation sets are $10M$ and $1M$. We initialize our model with the BART-large model ($n_{layer} = 12$, $n_{head} = 16$, $d_{hidden} = 1024$, and $406M$ parameters). We use AdamW (Loshchilov and Hutter, 2019) with an initial learning rate of $1e-5$ and $\alpha = 1$ to update our proposed model for two epochs and choose the checkpoint with the lowest validation loss. Please refer to Table 6 for the effect of the hyper-parameter α.

During inference, we run beam search decoding with beam width = 5 to generate text for sep-B/F, asyn-B/F and GBS. Following He and Li (2021), we run CGMH and X-MCMC-C for 200 refinement steps and select the candidate text with the lowest NLL as output. For POINTER, we use greedy decoding to generate constrained text. For CBART, we use the four parallel decoding methods (see Section 3.3). We apply the repetition penalty to sep-B/F, asyn-B/F, GBS, our models with $\theta = 2$, and POINTER with the default value $\theta = 1.25$.

We implement our model and baselines with HuggingFace (Wolf et al., 2019). Results of fine-tuned language models and well-trained classifiers of CBART are shown in the Appendix A and B.

Automatic Evaluation Metrics. We evaluate the generated sentences from two aspects: generation quality and diversity. Following previous work (Zhang et al., 2020b), we use BLEU (Papineni et al., 2002), NIST (Doddington, 2002) and METEOR (Banerjee and Lavie, 2005) as metrics for the generation quality, which measure the similarity between the generated text and the human reference. A higher BLEU, NIST or METEOR score indicates that a model can generate sentences similar to human references. In this paper, we do not use NLL as a metric for sentence fluency, since a lower NLL value does not always denote better sentence quality. Recent work (Holtzman et al., 2020) has found that language models assign low NLL scores not only to high-quality sentences, but also to repetitive and generic sentences.

As for generation diversity, we first compute the cumulative 4-gram Self-BLEU score (SB-4) (Zhu et al., 2018) to measure how similar one sentence is to the other generations by treating one sentence as the hypothesis and the others as references. Then, we calculate distinct bigrams (D-2) and 4-grams (D-4) (Li et al., 2016), which are the number of unique bigrams and 4-grams divided by the total number of generated tokens. A lower Self-BLEU or higher distinct n-gram value indicates higher diversity. Finally, we measure the n-gram repetitions on a sentence level. Since the length of generated sentences varies greatly, we focus on each sentence's first 20 tokens. Concretely, if a unigram appears more than two times or a trigram appears more than one time within the first 20 tokens of a sentence, we will regard the sentence as containing a repetition.

4.2 Main Comparison Experiment Results

We show the experiment results on One-Billion-Word and Yelp test sets in Table 2, from which we can draw four conclusions:

(1) **Sep-B/F, asyn-B/F and GBS have low generation quality and diversity.** Sep-B/F, asyn-B/F and GBS have low BLEU, NIST and METEOR values, indicating poor generation quality. That is possible because these models force keywords to be incorporated into outputs during decoding, thus degrading the generation quality. Moreover, compared with human-written text, sentences generated by sep-B/F, asyn-B/F and GBS are much less diverse as they have higher Self-BLEU and lower

		Metrics	BLEU ↑		NIST ↑		↑	↓	Distinct ↑		↓	↓	↑		
			B-2	B-4	N-2	N-4	M	SB-4	D-2	D-4	Ref	La	S	Rep	Len
One-Billion-Word		Human	-	-	-	-	-	10.3%	78.1%	99.5%	-	-	-	1.4%	23.6
	Baselines	sep-B/F	4.4%	0.7%	0.616	0.618	7.0%	52.1%	46.3%	78.8%	-	1.900	5.20	0.2%	13.5
		asyn-B/F	4.3%	0.7%	0.554	0.556	6.8%	50.3%	47.8%	80.9%	-	1.865	5.29	0.1%	13.2
		GBS	10.1%	2.8%	1.487	1.497	13.5%	37.0%	59.3%	87.2%	-	9.234	1.07	0	14.0
		CGMH	9.9%	3.5%	1.153	1.165	13.1%	10.2%	**78.9%**	99.3%	200	9.871	1.00	2.5%	11.6
		X-MCMC-C	12.5%	4.1%	2.511	2.532	13.8%	16.9%	69.7%	98.8%	200	31.41	0.31	1.4%	16.6
		POINTER	2.5%	0.1%	0.961	0.961	10.2%	-	-	-	6	-	-	-	65.0
		POINTER-2	8.7%	1.6%	2.109	2.117	14.3%	37.3%	46.5%	90.9%	6	0.727	13.6	18.6%	35.5
	CBART	**Greedy**	15.6%	**6.6%**	2.157	2.191	**15.2%**	22.1%	66.6%	97.2%	**4.8**	**0.351**	**28.1**	1.2%	14.5
		k**=5,** c**=1**	15.0%	5.8%	2.460	2.491	14.8%	16.1%	70.6%	98.8%	5.1	0.360	27.4	1.0%	15.7
		k**=5,** c**=5**	15.6%	5.9%	2.677	2.712	14.9%	19.9%	65.8%	98.0%	5.3	0.669	14.8	2.1%	16.4
		k**=50,** c**=1**	14.4%	5.1%	2.740	2.768	14.2%	11.0%	76.3%	99.5%	5.5	0.396	24.9	0.8%	17.2
		k**=50,** c**=5**	15.1%	5.4%	2.941	2.974	14.4%	13.7%	71.6%	99.2%	5.7	0.720	13.7	1.6%	18.1
		p**=0.5,** c**=1**	15.1%	6.1%	2.308	2.341	14.8%	14.9%	72.7%	99.0%	5.0	0.373	26.5	0.9%	15.2
		p**=0.5,** c**=5**	**15.8%**	6.4%	2.524	2.561	14.9%	18.2%	68.0%	98.4%	5.2	0.674	14.6	1.4%	15.7
		p**=0.9,** c**=1**	14.7%	5.4%	2.780	2.811	14.2%	**9.6%**	78.8%	**99.6%**	5.5	0.408	24.2	0.7%	17.3
		p**=0.9,** c**=5**	15.2%	5.5%	**2.983**	**3.017**	14.4%	12.2%	74.4%	99.4%	5.7	0.759	13.0	1.1%	18.3
Yelp		Human	-	-	-	-	-	26.1%	57.7%	97.0%	-	-	-	2.0%	23.9
	Baselines	sep-B/F	6.9%	2.1%	0.521	0.531	8.7%	67.1%	31.9%	64.6%	-	1.807	6.14	0.1%	12.6
		asyn-B/F	7.5%	2.3%	0.698	0.711	9.0%	68.0%	31.0%	64.3%	-	1.771	6.26	0.1%	13.4
		GBS	13.6%	4.5%	1.680	1.712	15.3%	59.3%	37.5%	70.2%	-	8.634	1.28	0.3%	14.6
		CGMH	12.3%	4.6%	1.413	1.446	14.6%	23.6%	**60.7%**	97.7%	200	11.09	1.00	5.2%	12.6
		X-MCMC-C	15.3%	5.4%	2.753	2.803	15.5%	38.5%	47.4%	92.4%	200	31.68	0.35	2.0%	17.8
		POINTER	4.0%	0.3%	1.139	1.140	13.0%	-	-	-	6	-	-	-	65.3
		POINTER-2	10.6%	2.4%	2.142	2.164	16.8%	49.1%	35.2%	86.3%	6	0.741	15.0	13.6%	39.8
	CBART	**Greedy**	19.4%	**9.0%**	2.541	2.635	**17.4%**	45.1%	44.4%	88.1%	**4.9**	0.357	31.1	2.3%	15.4
		k**=5,** c**=1**	18.4%	7.4%	2.946	3.024	16.8%	35.8%	48.7%	94.1%	5.4	0.391	28.4	2.4%	17.4
		k**=5,** c**=5**	19.1%	7.7%	3.088	3.172	16.9%	38.8%	45.9%	92.7%	5.5	0.670	16.6	3.5%	18.0
		k**=50,** c**=1**	17.6%	6.7%	3.103	3.172	16.2%	26.4%	55.9%	97.3%	5.8	0.412	26.9	1.2%	19.5
		k**=50,** c**=5**	18.3%	7.0%	3.220	3.292	16.4%	29.5%	52.6%	96.4%	6.0	0.742	14.9	2.0%	20.4
		p**=0.5,** c**=1**	19.0%	8.3%	2.779	2.867	17.0%	37.9%	48.5%	92.9%	5.1	**0.349**	**31.8**	2.0%	16.5
		p**=0.5,** c**=5**	**19.7%**	8.6%	2.985	3.082	17.1%	41.2%	45.4%	91.2%	5.3	0.658	16.9	3.3%	17.2
		p**=0.95,** c**=1**	17.6%	6.8%	3.163	3.234	16.0%	**23.5%**	59.3%	**97.9%**	5.9	0.384	28.9	1.3%	20.2
		p**=0.95,** c**=5**	18.2%	6.9%	**3.225**	**3.295**	16.3%	27.3%	55.2%	97.0%	6.1	0.703	15.8	2.1%	21.1

Table 2: Results on One-Billion-Word and Yelp test sets. ("Human" means human references. k and p are hyper-parameters for top-k and top-p decoding, respectively. c is the number of parallel sequences for the multiple-sequence decoding. "M" refers to METEOR. "Ref" denotes the average number of refinements taken during decoding. "La" (latency) is the average decoding time (second) per sentence computed on test sets without mini-batching. "S" denotes speedup. "Rep" means the percentage of sentences containing n-gram repetitions. "Len" represents the average length of the generated sentences.) Results for sep-B/F and asyn-B/F are on the test set with $N = 1$ constraint. Results for the remaining models are averaged over the six test sets with $N = 1$ to $N = 6$ lexical constraints.

distinct n-gram scores. Since these models are not aware of keywords before generation, they tend to generate generic phrases (*'he said'*, *'he would'*, etc.), thereby degrading the generation diversity.

(2) **Sampling-based methods have high generation diversity and low quality.** CGMH is on par with humans in generation diversity (Self-BLEU and distinct n-gram scores), yet it comes at the expense of degrading sentence quality (lower BLEU scores). This conclusion is in line with the results in previous work (Zhang et al., 2020b; He and Li, 2021). Compared with CGMH, X-MCMC-C slightly boosts the generation quality due to the decreasing of random modifications.

(3) **POINTER reduces the inference latency yet with low generation quality.** Compared with other baselines, POINTER significantly reduces the inference latency but has low generation quality. There are two possible reasons: (1) POINTER is based on BERT, which is not designed for text generation; (2) POINTER imposes all the burden of generation on the decoder. In addition, pre-training POINTER on Wikipedia (POINTER-2) improves the performance, consistent with what is observed in previous work (Zhang et al., 2020b). However, it is not fair for other models, as we may also make improvements by training them on larger datasets.

(4) **The proposed model outperforms baselines in most metrics.** Similar to POINTER, CBART can refine multiple tokens in each refinement step. Therefore, CBART only needs several steps to complete a sentence with the given keywords, thus dramatically reducing the inference time. As shown in Table 2, CBART with greedy decoding needs around five refinement steps and is about 28 and 31 times faster than CGMH on One-Billion-Word and Yelp. However, CBART outperforms POINTER in generation quality and diversity by a large margin.

Fluency: which sentence is more fluent?				
Model A won		Tied	Model B won	
CBART	32.7%	24.0%	**43.3%**	Human
CBART	**70.0%**	14.0%	16.0%	GBS
CBART	**56.0%**	25.3%	18.7%	CGMH
CBART	**44.7%**	33.3%	22.0%	X-MCMC-C
CBART	**77.3%**	18.0%	4.7%	POINTER-2
Informativeness: which sentence is more informative?				
Model A won		Tied	Model B won	
CBART	8.7%	9.3%	**82.0%**	Human
CBART	**65.3%**	10.7%	24.0%	GBS
CBART	**52.7%**	18.0%	29.3%	CGMH
CBART	**33.3%**	35.4%	31.3%	X-MCMC-C
CBART	**48.0%**	12.0%	40.0%	POINTER-2

Table 3: Human evaluation on One-Billion-Word.

#	**Metrics** / **CBART**	↑ B-2	↑ N-2	↑ M	↓ SB-4	Rep
1	Left	**19.4%**	**2.541**	**17.4%**	45.1%	2.3%
2	Middle	19.3%	2.431	**17.4%**	45.5%	3.1%
3	Right	18.5%	2.151	17.2%	43.9%	2.5%
4	Random	18.2%	2.030	17.1%	42.2%	2.1%
5	TF-IDF	16.7%	1.778	16.7%	**36.7%**	0.9%
6	w/o RP	**22.3%**	**3.537**	17.2%	50.7%	54.7%
7	$N = 1$	5.7%	0.312	8.3%	55.2%	0.3%
8	$N = 2$	9.7%	0.742	11.8%	52.3%	0.7%
9	$N = 3$	16.0%	1.726	15.7%	45.5%	1.8%
10	$N = 4$	22.4%	2.996	19.4%	42.2%	2.8%
11	$N = 5$	28.0%	4.122	22.7%	38.7%	3.6%
12	$N = 6$	**34.5%**	**5.347**	**26.3%**	**36.5%**	4.7%

Table 4: Results of CBART variants on Yelp test sets. At the top, we compare the performance of CBART trained with different synthetic datasets. At the middle, we show the impact of the repetition penalty (RP). At the bottom, we show the effect of the number of lexical constraints N.

This is possible because CBART shifts part of the burden from the decoder to the encoder and benefits from the intrinsic generation ability of BART.

To summarize, it is non-trivial to satisfy all metrics when generating constrained sentences. CBART with greedy decoding can generate sentences with relatively high sentence quality and diversity while largely reducing the inference latency. We can also control sentence quality and diversity with top-k or top-p sampling. For example, increasing k or p allows CBART to generate tokens with low probabilities, thus improving sentence diversity.

4.3 Human Evaluation

To further assess the proposed model, we conduct a human evaluation. We compare CBART (greedy decoding) with GBS, CGMH, X-MCMC-C, POINTER-2 and human references. For each model, we randomly select 50 sentences and invite three volunteers[4] to compare the sentences generated by different models. Following previous work (Huang et al., 2020), each annotator should compare sentence A with sentence B and decide which one is more fluent or informative. A tie is allowed if they have no preference. Sentences in each pair are shuffled before annotation to avoid bias. We show the results of the human evaluation in Table 3. The inter-rater agreement measured by Fleiss' kappa (Fleiss, 1971) is 0.69 and 0.65 for fluency and informativeness, indicating a substantial inter-rater agreement, according to Landis and Koch (1977). CBART outperforms baselines in both fluency and informativeness. CBART is even on par with humans in terms of sentence fluency. However, the proposed model still lags far behind humans in terms of informativeness. We speculate that this is because the model is only taught to generate fluent sentences during training, resulting in the generated sentences being shorter than human references.

[4]All annotators have Bachelor's or higher degrees. They are independent of our research group.

4.4 Ablation Study and Analysis

Effect of Methods for Creating Synthetic Datasets. We train CBART on different synthetic datasets constructed with different insertion ways and show the results at the top of Table 4. CBART trained with the *left* method gets the best sentence quality in terms of BLEU, NIST and METEOR. We speculate that CBART fine-tuned with the *left* method generates the leftmost tokens first, which is consistent with the generation order of BART. Therefore, CBART trained with the *left* method has a relatively smaller gap between training and fine-tuning than other variants.

Effect of the Repetition Penalty. In Table 4, CBART with the repetition penalty removed tends to get stuck in repetition loops, and the percentage of sentences containing repetitions surges from 2.3% to 54.7% (row 1 vs. row 6).

Effect of the Number of Constraints. As shown at the bottom of Table 4, with constraints increased, BLEU, NIST, and METEOR also increase. Since the increased constraints shrink the solution space, it becomes much easier for the model to generate sentences close to human references, thus boosting

#	Metrics CBART	↑ B-2	↑ N-2	↑ M	↓ SB-4	Rep
1	LM w/ CAM	**19.4%**	**2.541**	**17.4%**	45.1%	2.3%
2	MLM w/ CAM	18.7%	2.296	17.2%	**43.9%**	2.1%
3	LM w/o CAM	19.2%	2.453	17.4%	45.8%	2.5%
4	MLM w/o CAM	10.2%	1.835	13.1%	53.3%	99.3%
5	w/ Random	16.4%	1.769	16.5%	47.9%	3.6%
6	w/ BART-base	18.4%	2.310	17.0%	45.6%	1.9%
7	w/ BART-large	**19.4%**	**2.541**	**17.4%**	**45.1%**	2.3%

Table 5: Results of CBART variants on Yelp test sets. At the top, we show the effect of training objectives. At the bottom, we compare the effect of pre-trained models. ("CAM" is the causal attention mask.)

#	Metrics	↑ B-2	↑ N-2	↑ M	↓ SB-4	Rep
1	$\alpha = 0.5$	17.7%	2.024	16.9%	**44.4%**	2.1%
2	$\alpha = 1.0$	**18.4%**	**2.310**	**17.0%**	45.6%	1.9%
3	$\alpha = 1.5$	18.2%	2.203	17.0%	45.3%	2.4%
4	$\alpha = 2.0$	18.1%	2.144	17.0%	44.8%	2.0%
5	$\alpha = 2.5$	18.1%	2.139	17.0%	45.2%	2.1%
6	$\alpha = 3.0$	18.1%	2.155	17.0%	44.8%	2.0%

Table 6: The effect of α. Results of CBART-base trained with different α on the Yelp test set.

the generation quality. Furthermore, the model diversity improves because it is less likely to generate similar sentences as constraints increase.

Effect of Training Objectives. We conduct experiments to analyze the effect of training CBART with different training objectives, language modeling (LM) and masked language modeling (MLM). The difference between them is that during training, LM will reconstruct the original text, while MLM only predicts the masked tokens. As shown in Table 5, CBART trained with LM (row 1) performs better than CBART trained with MLM (row 2), possibly because LM is more suitable for text generation, in line with previous results (Lewis et al., 2020).

Previous non-autoregressive translation (NAT) models (Ghazvininejad et al., 2019; Stern et al., 2019) removed the causal attention mask (CAM) from the decoder so that each target token can attend to other tokens of the decoder input. Therefore, we also conduct experiments to analyze the effect of CAM. However, we do not observe any significant improvement (row 1 vs. row 3) for our task. This arises from the difference in input between our task and machine translation. For machine translation, the encoder and decoder take different languages as input. By comparison, for our task, the decoder input is constructed by inserting mask tokens before some positions and replacing some tokens with mask tokens. Therefore, each token of the decoder input Y^M also appears in the encoder input X. When training CBART with CAM, each token of the decoder input can attend to the encoder input via cross attention, which is equivalent to attending to other tokens of the decoder input by removing CAM. That is why we cannot make further improvements by removing CAM.

Effect of Pre-trained Models. We train two base CBART models with 6 layers initialized with random values (row 5) or the BART-base model (row 6) and a large CBART model initialized with the BART-large model (row 7). From Table 5, we conclude that pre-trained models (row 5 vs. row 6) and model size are important (row 6 vs. row 7). These are in line with our intuitions: (1) CBART inherits some syntactic and semantic knowledge from BART; (2) increasing the model size will improve the performance. Note that in this paper, the experiment results of our proposed model are based on the large CBART model, if not specified.

Effect of the Hyper-parameter α. Since α is an important hyper-parameter for training CBART, we train CBART-base (the base CBART model, which is initialized with BART-base.) with different α. From the results in Table 6, we find that $\alpha = 1.0$ is an appropriate value for CBART. We speculate that when α is too large, CBART will pay more attention to the generation/decoder loss. On the contrary, when α is too small, CBART will put more focus on the classification/encoder loss. Both losses are essential for the performance of CBART. As we can see, CBART trained with $\alpha = 1.0$ performs best in most metrics, which is a good trade-off between the encoder loss and the decoder loss.

4.5 Samples and Analysis

We show some sentences generated by baselines and our proposed model with lexical constraints extracted from Yelp test sets in Table 7. From this table, we can see that the sentence generated by CBART with greedy decoding is more fluent and meaningful than baselines. We can also generate more diverse and informative sentences with top-p or top-k sampling by increasing k or p, but there is a risk of generating less fluent sentences (see $p = 0.95, c = 1$). Increasing the number of sampling sequence c can slightly alleviate this (see $p = 0.95, c = 5$). More generated sentences are shown in Table 11 and Table 12 in the Appendix C.

Cons	**family**, **good**, **location**, **star**
Human	my **family** and i did not have a **good** experience here at this **location** . the one **star** is for the food .
GBS	this is a **good location** for **family** and **star** wars fans .
CGMH	very nice **family** friendly spot , **good location** on **star** .
CBART with Different Decoding Methods	
greedy	my **family** and i always get **good** food at this **location** . the one **star** is for customer service .
k**=5,** c**=1**	my favorite **family** owned restaurant ! friendly staff , **good** food and the **location** is great and always a 5 **star** experience .
k**=50,** c**=1**	it was a **family** friendly restaurant in las vegas with **good** food for children and nice **location** , this two **star** is because of service .
k**=50,** c**=5**	great **family** owned restaurant and **good** food . great **location** and a solid **star** coffee experience in general !
p**=0.5,** c**=1**	my **family** and i really enjoyed this place . **good** food and great **location** , but my one **star** rating is for the service !
p**=0.95,** c**=1**	olive case has been **family** run and had a **good** reputation , but this **location** was the **star** . if you purchase online ask to check in on the map !
p**=0.95,** c**=5**	our **family** went here for dinner . we always have such an **good** time at this **location** on a **star** wars movie night !

Table 7: Sentences generated by baselines and CBART with lexical **cons**traints extracted from Yelp test sets. "Human" refers to the human reference.

5 Related Work

Pre-trained Language Model. Large-scale pre-trained models have achieved remarkable success in many natural language understanding (NLU) tasks (Devlin et al., 2019; Liu et al., 2019b) and natural language generation (NLG) tasks (Radford, 2018; Radford et al., 2019). Some unified pre-trained language models, such as XLNet (Yang et al., 2019), UNILM (Dong et al., 2019) and BART (Lewis et al., 2020), have attempted to solve both NLU and NLG tasks. Unlike previous work, we have extended our work from BART, in order to generate text under specified lexical constraints.

Non-autoregressive Generation. NAT (Gu et al., 2018) generates all tokens in parallel, thus speeding up the inference. The parallel decoding comes at the cost of degrading generation quality since NAT breaks the dependency among target tokens. To alleviate this problem, recent NAT models (Lee et al., 2018; Stern et al., 2019; Gu et al., 2019; Ghazvininejad et al., 2019) generate the output with several steps, making a trade-off between the decoding speed and generation quality. Susanto et al. (2020) extended Levenshtein Transformer (Gu et al., 2019) by injecting keywords into machine translation. It works on machine translation because the source and target are mostly aligned and the solution space is small. However, it performs worse than BART on general text generation (Lin et al., 2020), which has a much larger solution space. Different from previous work, CBART inherits the generation ability from BART while maintaining the decoding speed of NAT models.

Lexically Constrained Text Generation. B/F-LMs (Mou et al., 2015; Liu et al., 2019a) are limited to generating text with one lexical constraint. GBS (Hokamp and Liu, 2017; Post and Vilar, 2018) incorporates multiple constraints into the output by controlling the decoding process yet degrades the generation quality and diversity. Recently, MCMC sampling has been applied to constrained text generation (Miao et al., 2019; Zhang et al., 2020a). Nevertheless, refinements conducted by these models are decided randomly. To alleviate this, Sha (2020) used the gradient information, and He and Li (2021) used a token-level classifier to decide the refinements, but these models updated only one token in each step. POINTER (Zhang et al., 2020b) can refine multiple tokens in each step but is based on BERT and imposes all the generation burden on the decoder, thus degrading the sentence quality. To solve these problems, we propose CBART, which is based on BART and transfers part of the decoder's burden to the encoder.

Another approach also generates text based on keywords (Fan et al., 2018; Tan et al., 2020; Lin et al., 2020) but does not force them to appear in the output. By comparison, our work focuses on requiring all keywords to appear in the output.

6 Conclusion

In this paper, we presented CBART for lexically constrained text generation. Compared with previous work, CBART leverages BART and transfers part of the burden from the decoder to the encoder. Furthermore, CBART refines multiple tokens in parallel in each refinement step, thus accelerating the inference. Experiment results on One-Billion-Word and Yelp datasets show that CBART can generate fluent and diverse text with lexical constraints and dramatically reduce the inference time.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive and informative feedback.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation*, pages 65–72.

Mathias Berglund, Tapani Raiko, Mikko Honkala, Leo Kärkkäinen, Akos Vetek, and Juha T Karhunen. 2015. Bidirectional recurrent neural networks as generative models. In *NIPS*, pages 856–864.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *NIPS*, pages 13063–13075.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of ACL*, pages 889–898.

J. Fleiss. 1971. Measuring nominal scale agreement among many raters.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of AAAI*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of EMNLP*, pages 6114–6123.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *ICLR*.

Jiatao Gu, Changhan Wang, and Jake Zhao. 2019. Levenshtein transformer. In *NIPS*.

Xingwei He and Victor O.K. Li. 2021. Show me how to revise: Improving lexically constrained sentence generation with xlnet. In *Proceedings of AAAI*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of ACL*, pages 1535–1546.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *ICLR*.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of ACL*, pages 1638–1649.

Yichen Huang, Yizhe Zhang, Oussama Elachqar, and Yu Cheng. 2020. INSET: Sentence infilling with INter-SEntential transformer. In *Proceedings of ACL*, pages 2502–2515.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2020. Ctrl: A conditional transformer language model for controllable generation. In *ICLR*.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of EMNLP*, pages 1173–1182.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of ACL*, pages 7871–7880.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL*, pages 110–119.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *Proceedings of NAACL*, pages 1865–1874.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840.

Dayiheng Liu, Jie Fu, Qian Qu, and Jiancheng Lv. 2019a. Bfgan: Backward and forward generative adversarial networks for lexically constrained sentence generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2350–2361.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of AAAI*, pages 6834–6842.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING*, pages 3349–3358.

Lili Mou, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2015. Backward and forward language modeling for constrained sentence generation. *arXiv preprint arXiv:1512.06612*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of NAACL*, pages 1314–1324.

Alec Radford. 2018. Improving language understanding by generative pre-training. *Technical Report, OpenAI*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *Technical Report, OpenAI*.

Lei Sha. 2020. Gradient-guided unsupervised lexically constrained text generation. In *Proceedings of EMNLP*, pages 8692–8703.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*, pages 6830–6841.

Mitchell Stern, William Chan, J. Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *ICML*, pages 5976–5985.

Jinyue Su, Jiacheng Xu, Xipeng Qiu, and Xuanjing Huang. 2018. Incorporating discriminator in sentence generation: a gibbs sampling method. In *Proceedings of AAAI*.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. 2020. Lexically constrained neural machine translation with Levenshtein transformer. In *Proceedings of ACL*, pages 3536–3543.

Bowen Tan, Zichao Yang, Maruan AI-Shedivat, Eric P Xing, and Zhiting Hu. 2020. Progressive generation of long text. *arXiv preprint arXiv:2006.15720*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Jingjing Xu, Xu Sun, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of ACL*, pages 979–988.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NIPS*, pages 5753–5763.

Maosen Zhang, Nan Jiang, Lei Li, and Yexiang Xue. 2020a. Language generation via combinatorial constraint satisfaction: A tree search enhanced Monte-Carlo approach. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1286–1298.

Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and Bill Dolan. 2020b. POINTER: Constrained progressive text generation via insertion-based generative pre-training. In *Proceedings of EMNLP*, pages 8649–8670.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *SIGIR*, pages 1097–1100.

Appendix

A Performance of Language Models

The forward GPT-2, backward GPT-2, separate forward GPT-2 and separate backward GPT-2 are initialized with the pre-trained GPT-2 small model. These models are fine-tuned on the training sets of One-Billion-Word or Yelp. We choose the checkpoint with the lowest NLL loss on the validation set. They are used for baselines, including sep-B/F, asyn-B/F, GBS and CGMH. NLL results of the fine-tuned language models on the validation sets are shown in Table 8.

B Performance of Classifiers

We create synthetic datasets for CBART with One-Billion-Word and Yelp. We select $1M$ and $0.1M$ sentences from each dataset as the training and validation sets. For each sentence, we create 10 synthetic data instances. Therefore, for each dataset, the size of the synthetic training and validation sets are $10M$ and $1M$. We fine-tune CBART on the synthetic training set for two epochs with a learning rate of $1e-5$ and select the best checkpoint with the lowest loss on the synthetic validation set. We show the performance of classifiers of CBART-base (the base CBART model, which is initialized with BART-base.) and CBART-large (the large CBART model, which is initialized with BART-large.) in Table 9 and Table 10, respectively.

C Generating Text with Lexical Constraints

We show some text generated by baselines and our proposed model with lexical constraints extracted from One-Billion-Word and Yelp test sets in Table 11 and Table 12, respectively.

Models	One-Billion-Word	Yelp
Forward GPT-2	3.463	2.885
Backward GPT-2	4.130	3.171
Separate forward GPT-2	3.540	3.055
Separate backward GPT-2	4.377	3.260

Table 8: NLL of different language models on the validation sets.

Datasets	Labels	P	R	F1
One-Billion-Word	Copy	0.970	0.985	0.978
	Replacement	0.981	0.975	0.978
	Insertion	0.928	0.862	0.894
	Macro-average	0.960	0.941	0.950
Yelp	Copy	0.974	0.985	0.979
	Replacement	0.993	0.991	0.992
	Insertion	0.928	0.874	0.900
	Macro-average	0.965	0.950	0.957

Table 9: Results of the classifier of CBART-base on the synthetic validation sets of One-Billion-Word and Yelp. "P" and "R" denote precision and recall.

Datasets	Labels	P	R	F1
One-Billion-Word	Copy	0.974	0.986	0.980
	Replacement	0.985	0.984	0.984
	Insertion	0.933	0.880	0.906
	Macro-average	0.964	0.950	0.957
Yelp	Copy	0.978	0.985	0.981
	Replacement	0.995	0.994	0.994
	Insertion	0.925	0.894	0.910
	Macro-average	0.966	0.958	0.962

Table 10: Results of the classifier of CBART-large on the synthetic validation sets of One-Billion-Word and Yelp. "P" and "R" denote precision and recall.

	Constraints	**hearing**, **system**, **need**
	Human	We are already **hearing** arguments for focusing everything on the economy damaged by failure in the banking **system** , dropping the **need** to fix the climate system .
Baselines	GBS	" We **need** to have a **system** of **hearing** protection , " he said .
	CGMH	A public **hearing** is just one more **system** that we **need** .
	X-MCMC-C	The new public **hearing system** will **need** funding for about six months after the court ruling .
	POINTER-2	and he said at the senate **hearing** that it ? s changing in the entire current **system** , it ? s a **need** for a reform now . . .
CBART	**greedy**	The new **hearing system** is less expensive , and there was no **need** for a specialist .
	k**=5,** c**=1**	" The **hearing system** is something we **need** to improve .
	k**=50,** c**=1**	She has a **hearing system** and is in **need** of glasses .
	p**=0.5,** c**=1**	The **hearing system** is closed , but you will not **need** to pay for it .
	p**=0.9,** c**=1**	Is there any existing **hearing system** that would **need** to be adapted ?
	Constraints	**admitted**, **health**, **heavy**, **cold**
	Human	Philip , 86 , was **admitted** to the King Edward VII Hospital in central London on Thursday after his **health** deteriorated having caught a **heavy cold** .
Baselines	GBS	He **admitted** that he had been " **cold** and **heavy** " in the **health** department
	CGMH	He **admitted** that he had faced mental **health** problems and a **heavy cold** .
	X-MCMC-C	The Labour MP was **admitted** to a mental **health** hospital after suffering a **heavy cold** for most of the week .
	POINTER-2	but when she was **admitted** to a local mental **health** unit because what she did was so **heavy** a burden on mea , and one or more afraid of the other colds . . . ?
CBART	**greedy**	The singer **admitted** to having mental **health** problems and suffering from a **heavy cold** .
	k**=5,** c**=1**	The court **admitted** the pair have two previous **health** problems , including **heavy** doses of **cold** and flu medication .
	k**=50,** c**=1**	Health ministers **admitted** that the **health** service was weak because it had suffered a **heavy** dose of **cold** and flu .
	p**=0.5,** c**=1**	She **admitted** to having **health** problems , including **heavy cold** and flu .
	p**=0.9,** c**=1**	He had been **admitted** to the hospital mental **health** unit , suffering from a **heavy** head **cold** and fever .
	Constraints	**way**, **back**, **missing**, **weeks**, **due**
	Human	John Lackey is all the **way back** after **missing** the first six **weeks due** to injury , and pitching like an ace again .
Baselines	GBS	" The **way back** is **due** to the **missing weeks** , " he said .
	CGMH	The only **way back** is after **missing** four **weeks due** to injury .
	X-MCMC-C	He is on his **way back** home after **missing** two **weeks due** to an Achilles tendon injury .
	POINTER-2	he has found his **way** into england despite his **back** injury , **missing** three of the last two **weeks** with a calf injury and another two **due** to a calf injury .
CBART	**greedy**	He is on his **way back** to the squad after **missing** two **weeks due** to a stomach bug .
	k**=5,** c**=1**	She is making her **way back** into action after **missing** three **weeks due** to illness .
	k**=50,** c**=1**	Took his **way back** from a groin injury after **missing** six **weeks due** to knee injuries .
	p**=0.5,** c**=1**	But he was on his **way back** after **missing** two **weeks due** to a visa issue .
	p**=0.9,** c**=1**	He is on the **way back** after **missing** two **weeks due** to a fractured collarbone and bruised ribs .
	Constraints	**likely**, **certain**, **respond**, **others**, **new**, **environment**
	Human	That said , it is **likely** that **certain** forms of religion would **respond** better than **others** to the **new environment** .
Baselines	GBS	" The **new environment** is **likely** to **respond** to **certain others** , " he said
	CGMH	Not **likely** , but **certain** areas will **respond** better than **others** to the **new environment** .
	X-MCMC-C	And they are also **likely** to be **certain** to **respond** to **others** who want to create a **new environment** to replace the old .
	POINTER-2	they are more **likely** to be unable to do a **certain** things , about how they **respond** better than **others** in this area , are new **new** or why the current **environment** is different .
CBART	**greedy**	The first group is most **likely** to be **certain** to **respond** better than **others** in a **new environment** .
	k**=5,** c**=1**	The company is **likely** to depend on **certain** sectors and **respond** to **others** in the **new environment** .
	k**=50,** c**=1**	We have no more **likely** chance of survival , while **certain** groups **respond** differently and the **others** adapt to a **new environment** .
	p**=0.5,** c**=1**	How **likely** to be that you learn about **certain** issues and **respond** to **others** in a **new environment** .
	p**=0.9,** c**=1**	I suspect it is **likely** to find **certain** chemicals would **respond** better than **others** , which could cope in the **new environment** .

Table 11: Generated text with constraints from One-Billion-Word test sets. "Human" refers to the human reference.

	Constraints	**past**, **decided**, **try**
	Human	i have driven **past** this place a few times , and finally one morning i **decided** to give it a **try** .
Baselines	GBS	i **decided** to give this place a **try** based on the **past** reviews .
	CGMH	driving **past** this place and **decided** to **try** it out .
	X-MCMC-C	we stayed here this **past** weekend and **decided** to **try** it out .
	POINTER-2	i have been walking **past** by here a few times , and finally **decided** to give it a **try** out .
CBART	**greedy**	i drive **past** this place everyday and finally **decided** to **try** it out .
	k**=5,** c**=1**	i walked **past** this restaurant and **decided** to give it a **try** one evening .
	k**=50,** c**=1**	we have been driving **past** this buffet but it was a while so we **decided** to **try** this restaurant since they were in a good location .
	p**=0.5,** c**=1**	i have driven **past** this place many times and finally **decided** to **try** it out .
	p**=0.95,** c**=1**	my husband has always drove **past** this place and we **decided** to **try** it .
	Constraints	**dinner**, **called**, **arrived**, **early**
	Human	christmas eve **dinner** at a so **called** steakhouse . **arrived early** for our 630 reservation and we had to wait 20 minutes for a table .
Baselines	GBS	we **arrived early** for **dinner** and **called** to make an appointment .
	CGMH	when our first wedding anniversary **dinner** was **called** , the driver **arrived** 10 minutes **early** .
	X-MCMC-C	we just had **dinner** here tonight and we **called** ahead to place our order and **arrived early** .
	POINTER-2	went here to have **dinner** out here tonight . we had **called** in a few other places were ahead of me , and **arrived** just a few minutes **early** , and were promptly seated .
CBART	**greedy**	we had a wonderful **dinner** here last night . we **called** ahead and **arrived** 15 minutes **early** .
	k**=5,** c**=1**	i had **dinner** there last night . i **called** ahead for a reservation and **arrived** 10 minutes **early** .
	k**=50,** c**=1**	we came here with a **dinner** for 10 , **called** in and **arrived** 20 minutes **early** .
	p**=0.5,** c**=1**	had **dinner** reservations and **called** to confirm , **arrived** 15 minutes **early** .
	p**=0.95,** c**=1**	we had **dinner** here and enjoyed everything . i **called** in ahead so we **arrived** 15 minutes **early** .
	Constraints	**eat**, **crab**, **fresh**, **large**, **salty**
	Human	got all u can **eat crab** legs and they were **fresh** , **large** , not **salty** , and not over cooked like some local restaurants .
Baselines	GBS	this is a great place to **eat** . **fresh crab** , **large salty** fish ,
	CGMH	do not **eat** the **crab** legs here ! **fresh** , **large** , and **salty** !
	X-MCMC-C	the best all you can **eat** buffet the **crab** legs are **fresh** and **large** and everything is not too **salty** or heavy .
	POINTER-2	great place to **eat** at off the strip ! i had the snow **crab** wontons . they were **fresh** and delicious . they were in a very **large** portion and were not greasy or too **salty** at all .
CBART	**greedy**	i **eat** here often , the **crab** legs are **fresh** and **large** and not too **salty** .
	k**=5,** c**=1**	this is a place i **eat** at ! the **crab** cakes were **fresh** , **large** and not overly **salty** .
	k**=50,** c**=1**	this is one great place to **eat** at on vacation ! the **crab** legs were **fresh** and **large** slices of butterfish ; very **salty** , though .
	p**=0.5,** c**=1**	all you can **eat** here is great . the **crab** legs were **fresh** and **large** , but a little too **salty** .
	p**=0.95,** c**=1**	we came just for the opportunity of hicklate late night bite too **eat** here . oysters & **crab** legs are **fresh** , **large** portion and **salty** enough to finish !
	Constraints	**way**, **home**, **decided**, **friendly**, **loved**, **atmosphere**
	Human	this place was on the **way** back **home** for me and i **decided** to try it out . the workers were super **friendly** and i **loved** the **atmosphere** .
Baselines	GBS	i **loved** the **atmosphere** and **friendly** staff . i **decided** to go **home way** before
	CGMH	our **way home** was **decided** very fast ! very **friendly** service , **loved** the food and **atmosphere** !
	X-MCMC-C	it was on our **way home** we **decided** to stop in and they were very **friendly** and we **loved** the **atmosphere** and decor .
	POINTER-2	we were out on our **way** back to **home** , and we are so very glad that we **decided** to stop here . very **friendly** local coffee shop ! i **loved** the decor and the **atmosphere** of this place ! .
CBART	**greedy**	we were on our **way home** and **decided** to stop for a bite , **friendly** staff and **loved** the **atmosphere** .
	k**=5,** c**=1**	on my **way home** and **decided** to try it out ! **friendly** staff , **loved** the **atmosphere** .
	k**=50,** c**=1**	we saw it on our **way home** and **decided** to stop in . very **friendly** staff and i **loved** the **atmosphere** of this brewery !
	p**=0.5,** c**=1**	on our **way home** and **decided** to stop for dinner ! **friendly** staff , **loved** the **atmosphere** .
	p**=0.95,** c**=1**	stumbled upon this place on my **way home** one night , & **decided** to give it a try ! super **friendly** staff and **loved** the **atmosphere** .

Table 12: Generated text with constraints from Yelp test sets. "Human" refers to the human reference.