Show Me How To Revise: Improving Lexically Constrained Sentence Generation with XLNet

Xingwei He, Victor O.K. Li
Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, China
hexingwei15@gmail.com, vli@eee.hku.hk

Abstract

Lexically constrained sentence generation allows the incorporation of prior knowledge such as lexical constraints into the output. This technique has been applied to machine translation, and dialog response generation. Previous work usually used Markov Chain Monte Carlo (MCMC) sampling to generate lexically constrained sentences, but they randomly determined the position to be edited and the action to be taken, resulting in many invalid refinements. To overcome this challenge, we used a classifier to instruct the MCMC-based models where and how to refine the candidate sentences. First, we developed two methods to create synthetic data on which the pre-trained model is fine-tuned to obtain a reliable classifier. Next, we proposed a two-step approach, "Predict and Revise", for constrained sentence generation. During the *predict* step, we leveraged the classifier to compute the learned prior for the candidate sentence. During the *revise* step, we resorted to MCMC sampling to revise the candidate sentence by conducting a sampled action at a sampled position drawn from the learned prior. We compared our proposed models with many strong baselines on two tasks, generating sentences with lexical constraints and text infilling. Experimental results have demonstrated that our proposed model performs much better than the previous work in terms of sentence fluency and diversity. Our code, pre-trained models and Appendix are available at https://github.com/NLPCode/MCMCXLNet.

Introduction

Recently, there has been much interest in generating sentences in a controlled manner. Lexically constrained sentence generation aims to incorporate some pre-specified keywords or phrases into the generated sentences, and has been widely used for many natural language processing (NLP) tasks. For example, Mou et al. (2016) alleviated generating universal dialog responses by injecting a keyword into the dialogue replies. Some researchers (Hokamp and Liu 2017; Post and Vilar 2018) incorporated the domain-specific terminologies into the translated sentences.

To generate sentences with lexical constraints, Mou et al. (2015) proposed a novel backward and forward language model (B/F-LM). Liu et al. (2019b) extended B/F-LM by introducing a discriminator. However, the capability of these models is limited, as they can generate sentences with only one lexical constraint. To generate sentences with multiple lexical constraints, Hokamp and Liu (2017) proposed grid beam search (GBS) by controlling the decoding process. But this may degrade the quality of the generated sentence since the constraints are not considered when generating previous tokens. Applying GBS to unconditional generative models will suffer from this issue more seriously because the solution space is much wider than that of conditional generative models. In addition, beam search-based models tend to generate generic and repetitive sentences in unconditional text generation (Holtzman et al. 2020).

Another line of work applies MCMC sampling to lexically constrained sentence generation. Berglund et al. (2015) first used Gibbs sampling to generate sentences from the bidirectional RNN. Then, Wand and Cho (2019) extended Gibbs sampling to generate sentences from BERT (Devlin et al. 2019). Su et al. (2018) incorporated soft constraints such as sentiments via discriminators into the generated sentences. Furthermore, Miao et al. (2019) proposed CGMH, which can generate flexible-length sentences under the given lexical constraints with replacement, insertion, and deletion actions. Different from GBS, MCMC-based models generate the constrained sentence via many refinements instead of one pass, which allows them to utilize both the past and future contexts to refine tokens one by one iteratively.

However, previous MCMC-based models typically conduct many invalid and redundant refinements because of the randomly chosen actions and positions. This problem is more serious when generating sentences with lexical constraints since it needs many more operations to generate a complete sentence. To demonstrate this problem, we showed an example in Figure 1. It is easy for us to realize that a token should be inserted before "very" to complete this sentence. Therefore, operations such as replacing a token with another token or inserting a token at some other position are very likely to be rejected since these refinements can hardly improve the quality of the candidate sentence. Without sufficient refinements, these models will risk generating ungrammatical sentences.

To reduce the redundant refinements conducted by MCMC-based models, we proposed a two-step approach, "Predict and Revise", for constrained sentence generation, shown in Figure 1. Intuitively, suppose a classifier can tell

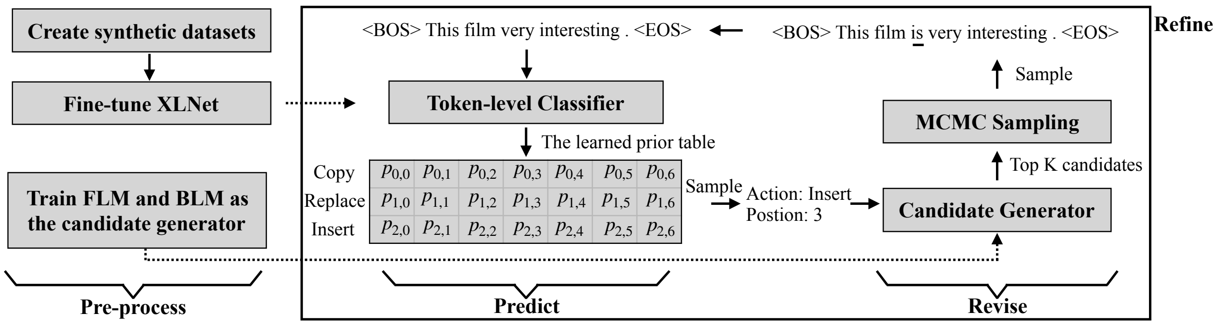


Figure 1: Illustration of our approach. In the *predict* step, we leveraged the token-level classifier to compute the learned prior for the current sentence. Then, we sampled a proposed action and position from the learned prior. In the *revise* step, we generated the top K candidate sentences, from which we sampled a candidate sentence with MCMC.

the model where and how to refine the candidate sentence, then we will undoubtedly avoid many invalid operations, thus improving the quality of the generated sentences. To train a reliable classifier, we proposed two methods (random and masked language model) to create synthetic data. When creating synthetic data for the replacement action, the random method replaces the chosen token with a randomly sampled token, while the masked language model method masks the chosen token and leverages the permutation-based language model, i.e., XLNet (Yang et al. 2019), to predict the masked token. A token from the top N tokens is drawn as the replaced token. These two methods are complementary to each other, contributing to training a classifier. Then, we fine-tuned XLNet on the synthetic dataset to obtain a token-level classifier, which provides accurate revision information including the position to be revised and the action to be conducted. In the *predict* step, an action and a position are drawn from the learned prior given by the classifier for the candidate sentence. In the *revise* step, we used the candidate generator to generate the top K candidate sentences, and drew a candidate sentence with MCMC.

The main contributions of our work are threefold: (1) we proposed two approaches to create the synthetic dataset and fine-tuned XLNet to obtain a reliable classifier; (2) we proposed a two-step approach for constrained sentence generation, guided by the classifier to refine the candidate sentence; (3) we conducted extensive experiments on two tasks, generating sentences with lexical constraints and text infilling. Experiment results show that our proposed model outperforms previous baselines in sentence fluency and diversity.

Problem Definition

Generating Sentences with Lexical Constraints

Generating sentences with lexical constraints aims to incorporate the given lexical constraints into the output. Given the constraints $c_1, c_2, \cdots, c_k$, this task is defined as follows:

$$X^* = \arg\max_X P(X|c_1, c_2, \cdots, c_k), \tag{1}$$

where X is the sentence containing the given lexical constraints. This task is meant to find a fluent sentence by maximizing the conditional probability.

Text Infilling

Text infilling aims to infill the missing portions of a sentence based on the past and future contexts to make the sentence complete and fluent. In this paper, we follow a more general definition of text infilling (Zhu, Hu, and Xing 2019), where the number of missing portions is arbitrary, and each portion can be infilled with an arbitrary number of tokens. Given an incomplete sentence $X^B = \{x_1, \cdots, x_{i-1}, \underline{m}, x_{i+1}, \cdots, x_{j-1}, \underline{m}, x_{j+1}, \cdots, x_n\}$, an arbitrary number of tokens can be filled into each blank $\underline{m}$ to make it meaningful. Therefore, text infilling can be formulated as follows:

$$X^* = \arg\max_X P(X|X^B). \tag{2}$$

Text infilling is a special case of constrained sentence generation, where the known portions can be regarded as lexical constraints. Compared with generating sentences with lexical constraints, tokens are permitted to be inserted only at specific positions, where blanks are located.

Methodology

The overview of our proposed approach is demonstrated in Figure 1. In this section, we will first introduce how to create the synthetic dataset for training a classifier. Then, we will describe how to train a classifier and use it to compute the learned prior for the candidate sentence. Finally, we will demonstrate the process of generating candidate sentences with the candidate generator and refining the current candidate sentence with MCMC sampling.

Creating the Synthetic Dataset

We aim to obtain a three-class token-level classifier, which is expected to tell us how to refine the candidate sentence. We use labels 0, 1, 2 to indicate copy, replacement, and insertion actions. The copy action indicates that the current token does not need to be modified. The replacement action indicates that the current token should be replaced with another token to make the sentence more fluent. Similarly, the insertion action means some tokens should be inserted before the current token to complete the sentence. To train a reliable classifier, we will create the synthetic dataset $D = \{(X, L)\}$, where

(X, L) denotes a data instance. We used One-Billion-Word corpus[1] to construct the synthetic dataset. Each time we randomly selected a sentence from One-Billion-Word corpus. Assume the selected sentence is "Opponents of the tariff say U.S. manufacturing would suffer under the climate bill regardless of trade policy changes." Then, we randomly chose a continuous segment from this sentence, "manufacturing would suffer under the climate bill."

To construct synthetic data for the insertion action, we randomly deleted some tokens from the selected part. Then, we obtained a synthetic input pair, X={<BOS>, manufacturing, suffer, under, the, climate, bill, <EOS>} and $L = \{0, 2, 2, 0, 0, 0, 0, 2\}$. In the above example, the labels for "manufacturing" and "<EOS>" are 2, which means we need to insert something before these tokens. Since we deleted "would" from the selected sub-sentence, the label for "suffer" is also 2. Labels for the remaining tokens are 0. "<BOS>" and "<EOS>" are special tokens, which represent the start and end of a sentence, respectively.

We proposed two methods (random and masked language model) to create synthetic data for the replacement action. The random method replaces the chosen token with a randomly sampled token. For example, replacing the token "would" with "(". In this case, the classifier can easily infer the label for "(" is 1. Nevertheless, this case is far from the real mistakes made by humans. Therefore, the classifier will struggle to infer labels in scenarios where tense errors appear, or some inappropriate words are used. To solve this problem, the masked language model method uses XLNet to mimic mistakes made by humans. Specifically, we first replaced the chosen token, e.g., "would" with the masked token "<MASK>". Then, we used XLNet to predict the masked token "<MASK>" and sampled a token from the top N predicted tokens with the highest probability. (In our experiments, we set N to 20). Before using XLNet to create synthetic data, we fine-tuned it on the masked dataset constructed with One-Billion-Word to give reliably predicted tokens. These two methods are complementary to each other, and we combined them to create the synthetic dataset.

The Token-level Classifier

After getting the synthetic dataset, we fine-tuned XLNet on them to get a token-level classifier. We chose XLNet as the classifier because it can not only be used as a classifier but also serve as a transformer-based language model (Vaswani et al. 2017). We also conducted experiments using BERT as the classifier and GPT-2 as the language model, but the results are worse than XLNet because BERT and GPT-2 do not share the same vocabulary.

In this paper, we also conducted experiments with LSTM-based language models. However, the learned prior given by XLNet cannot be directly used by LSTM-based language models. Since XLNet uses SentencePiece (Kudo and Richardson 2018) to tokenize the text, each token of LSTM-based language models may be divided into multiple sub-tokens. To solve this problem, we use the label of the first sub-token as the label for the given token. For example, the token "fine-tuning" is divided into four sub-tokens ("_fine", "-", "tun" and "ing") by XLNet. The label of "_fine" will be regarded as the label of "fine-tuning" when feeding the outputs of XLNet classifier to LSTM-based language models.

After getting the learned prior table P shown in Figure 1, we set the replacement probability of the "<EOS>" token and lexical constraints to zero to make sure the given keywords appear in the output. In addition, we set the probabilities of the "<BOS>" token to zero to prohibit any modification at the starting position. We computed the sum of these probabilities across the horizontal dimension of the table to compute the probabilities for the replacement and insertion actions, from which we sampled an action a. Then, we drew a position from the row of the sampled action.

The Candidate Generator and MCMC Sampling

Suppose the candidate sentence at time step t is $X_t = [x_1, \cdots, x_i, \cdots, x_n]$. Assume the proposed action and position is replacement and i, respectively. We need to compute the probability of replacing the i-th token with another token. Using Gibbs sampling (Geman and Geman 1984), we computed the conditional distribution as follows:

$$q(x_i = w_j | x_{-i}) = \frac{p(x_1, \cdots, x_i = w_j, \cdots, x_n)}{\sum_{w \in V} p(x_1, \cdots, x_i = w, \cdots, x_n)}, \quad (3)$$

where V denotes the vocabulary, and x_{-i} denotes tokens of X_t except x_i. $p(x_1, \cdots, x_i = w_j, \cdots, x_n)$ denotes the sentence probability, computed by a forward language model.

It is non-trivial to compute the conditional distribution with Equation (3) since we need to compute probabilities for $|V|$ candidate sentences. Previous work resorted to a candidate generator to solve this problem, which takes x_{-i} as inputs and outputs the top K tokens with the highest probabilities. Then, we get K candidate sentences by replacing the i-th token with the top K tokens. In this paper, we test four different candidate generators. First, we trained a forward language model (FLM), a backward language model (BLM), and a masked language model (MLM). FLM takes the tokens before x_i as inputs ($FLM(x_{<i})$), while BLM takes the tokens after x_i as inputs ($BLM(x_{>i})$). To leverage both the past and future contexts, MLM takes the tokens before and after x_i as inputs ($MLM(x_{<i}, x_{>i})$). Our last method combines FLM and BLM ($FLM(x_{<i}) \times BLM(x_{>i})$), and also uses the contextual information to predict x_i. Not surprisingly, the last method performs better than using only FLM or BLM as the candidate generator. However, we also found that the last method performs slightly better than MLM. We speculated that the conditional distribution given by the last method is more consistent with Equation (3). Therefore, in the following experiments, we use the last method as our candidate generator, and we set K to 50.

Assume the proposed action is insertion. We need to insert a special token, "<MASK>", before x_i. We used the candidate generator to generate the top K candidate sentences. Then, we used FLM to compute sentence probabilities for them. Finally, we sampled a sentence X' from them. Similar to the replacement action, a sentence with a higher probability is more likely to be selected. The difference is that the proposal distribution for insertion is asymmetric. To

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

Algorithm 1 Constrained Sentence Generation with XLNet

```
1:  Create the synthetic dataset, and train a token-level classifier $C$ on the synthetic training set.
2:  Train FLM and BLM on the training set.
3:  Set lexical constraints as the initial state $X_0$ of MCMC.
4:  for $t \leftarrow 1$ to $T$ do
5:      Compute the learned prior $C(X_{t-1})$, and draw an action $A$ and a position $P$ from $C(X_{t-1})$.
6:      Compute the top $K$ candidate tokens with the candidate generator for $(A, P, X_{t-1})$.
7:      Create $K$ candidate sentences with the top $K$ tokens, and compute probabilities for them with FLM.
8:      Draw a sentence $X'$ from the candidate sentences based on the normalized sentence probabilities.
9:      Compute the acceptance rate $A(X'|X_{t-1})$, and draw a number $\alpha$ from Uniform [0,1].
10:     If $\alpha < A(X'|X_{t-1})$ then $X_t \leftarrow X'$ else $X_t \leftarrow X_{t-1}$.
11: end for
12: Output the sentence $X^*$ with the lowest NLL.
```

make it meet the *detailed balance condition*, the Metropolis-Hastings (MH) algorithm (Metropolis et al. 1953; Hastings 1970; He et al. 2017a,b) introduces an acceptance term:

$$A_{insert}(X'|X_t) = min(1, A^{\star}_{insert}(X'|X_t)), \quad (4)$$

$$A^{\star}_{insert}(X'|X_t) \approx \frac{q(X_t|X') \times p(X')}{q(X'|X_t) \times p(X_t)} \quad (5)$$

$$\approx \frac{\sum_{X \in S} p(X)}{p(X_t)}, \quad (6)$$

where S is the set of candidate sentences for the insertion action. When computing the acceptance rate, we ignored the probabilities for the chosen action and position, and we found ignoring these terms improves the experiment results.

We summarize the process of our proposed approach in Algorithm 1. We first created synthetic datasets and then fine-tuned XLNet on them to get the classifier. Next, we trained FLM and BLM and used them as the candidate generator. Finally, we refined the candidate sentence with the classifier and MCMC sampling.

Experiments

Generating Sentences with Lexical Constraints

Experiment Setups and Baselines. We selected 6M, 0.3M and 1K sentences from One-Billion-Word corpus as the training, validation, and test sets, respectively. We trained forward and backward LSTM-based language models on the training set. Similarly, we fine-tuned the pre-trained XLNet (base-cased version) model on the training set to get forward and backward XLNet-based language models. For each language model, we selected the checkpoint with the lowest validation loss. For both LSTM-based and XLNet-based models, the forward and backward language models serve as candidate generators. The forward language models are also used to compute sentence probabilities. We fine-tuned XLNet (base-cased version) on the synthetic dataset to get our classifier. The experiment setups for language models and the classifier are shown in the Appendix. We constructed four kinds of test sets for constrained sentence generation by varying the length of lexical constraints k from 1 to 4. For each case, we randomly extracted $1,000$ sets of lexical constraints from the test sentences.

To compare with previous work, we implemented two variants of the backward and forward language model (sep-B/F and asyn-B/F), and GBS. After these models are well-trained, we ran beam search decoding (beam width = 10) to generate sentences with the extracted lexical constraints. As for CGMH, we used the code and the pre-trained model provided by the author to generate sentences.

In this paper, we proposed four models. The LSTM-based MCMC model (L-MCMC) uses LSTM-based models as the generator and language models. L-MCMC w/ deletion is our implementation for CGMH. As shown in Table 1, sentences generated by our implementation have lower NLL values indicating higher quality. L-MCMC only uses replacement and insertion actions, which outperforms L-MCMC w/ deletion. Therefore, all our proposed models don't use the deletion action. When refining the generated sentence, L-MCMC randomly chooses a position and an action. In comparison, the LSTM-based MCMC w/ classifier (L-MCMC-C) samples a position and an action from the learned prior table provided by the classifier. Similarly, we also extended MCMC sampling to XLNet-based models and proposed the XLNet-based MCMC (X-MCMC) and the XLNet-based MCMC w/ classifier (X-MCMC-C). Our models are implemented with HuggingFace (Wolf et al. 2019).

For a fair comparison, all MCMC-based models were run for 200 steps to generate sentences with the given constraints and then output the sentence with the lowest NLL. In addition, sep-B/F, asyn-B/F, GBS, L-MCMC, and L-MCMC-C use the same LSTM model structure.

Automatic Evaluation for Quality. To automatically evaluate the quality of generated sentences, we followed (Wang and Cho 2019) by computing negative log-likelihood (NLL) of sentences. A lower NLL value means that the generated sentence is more fluent and coherent. We used the pre-trained GPT-2 small (117M) to measure NLL values of sentences. We also computed BLEU scores (Papineni et al. 2002) between generated sentences and human references. A high BLEU score indicates a model can generate sentences similar to human references. NLL and BLEU results are shown in Table 1. Our proposed models (L-MCMC-C and X-MCMC-C) outperform baselines in NNL.

It is worth mentioning that sep-B/F, asyn-B/F, and GBS achieve relatively low NLL and high BLEU scores. Does it mean these models can generate well-formed sentences? Holtzman et al. (2020) found that beam search decoding tends to lead to degeneration. Language models are expected to assign low NLL scores to high-quality sentences, yet they also give low NLL scores to repetitive and generic sentences. We found that beam search-based models (sep-B/F, asyn-B/F, and GBS) may easily fall into repetitive loops, which is consistent with what is observed in previous work (Holtzman et al. 2020). We showed the percentage of sentences

Metrics	NLL (GPT-2) (↓)				BLEU (bigram) (↑)				Human evaluation (↑)			
Models	k=1	k=2	k=3	k=4	k=1	k=2	k=3	k=4	k=1	k=2	k=3	k=4
sep-B/F	4.432	-	-	-	**7.1%**	-	-	-	0.333	-	-	-
asyn-B/F	4.304	-	-	-	**7.1%**	-	-	-	0.369	-	-	-
GBS	3.985	4.163	4.178	4.209	6.6%	**9.8%**	**12.8%**	16.0%	0.495	0.384	0.445	0.421
CGMH (200)	5.079	5.103	5.227	5.246	1.3%	3.9%	7.3%	11.5%	0.345	0.344	0.400	0.408
L-MCMC w/ deletion (200)	4.923	4.775	4.856	4.874	1.0%	3.9%	7.7%	12.5%	-	-	-	-
L-MCMC (200)	4.549	4.564	4.617	4.672	2.5%	5.9%	10.3%	14.4%	0.433	0.459	0.469	0.429
L-MCMC-C (50)	4.425	4.389	4.491	4.487	1.6%	4.8%	8.9%	14.2%	-	-	-	-
L-MCMC-C (200)	3.762	3.773	3.819	3.904	3.7%	7.5%	12.4%	**16.7%**	0.575	0.520	0.573	0.557
X-MCMC (200)	4.225	4.319	4.427	4.463	1.7%	4.5%	8.3%	12.4%	0.471	0.516	0.487	0.516
X-MCMC-C (60)	4.152	4.259	4.398	4.490	1.9%	4.7%	8.3%	12.1%	-	-	-	-
X-MCMC-C (200)	**3.532**	**3.683**	**3.779**	**3.879**	2.9%	6.1%	10.0%	14.4%	**0.681**	**0.589**	**0.621**	**0.655**

Metrics	Self-BLEU (4-gram) (↓)				Distinct (bigram) (↑)				Entropy (4-gram) (↑)			
Human Reference	8.2%	9.0%	9.0%	9.0%	81.5%	80.7%	80.7%	80.7%	9.882	9.875	9.875	9.875
sep-B/F	81.7%	-	-	-	19.5%	-	-	-	7.664	-	-	-
asyn-B/F	80.2%	-	-	-	20.3%	-	-	-	7.924	-	-	-
GBS	84.2%	80.6%	73.4%	67.9%	16.2%	20.9%	27.0%	31.8%	6.741	7.555	8.117	8.567
CGMH (200)	19.8%	14.9%	11.7%	11.4%	65.6%	68.0%	69.9%	71.0%	8.605	8.871	9.124	9.301
L-MCMC w/ deletion (200)	14.4%	13.8%	10.5%	9.2%	74.1%	74.8%	76.2%	77.0%	8.309	8.676	8.961	9.172
L-MCMC (200)	20.1%	16.5%	13.8%	13.3%	64.4%	67.3%	68.6%	68.6%	9.014	9.260	9.497	9.669
L-MCMC-C (50)	20.2%	16.3%	14.2%	12.6%	68.6%	70.8%	71.1%	73.1%	8.591	8.901	9.144	9.339
L-MCMC-C (200)	33.0%	28.3%	24.1%	21.9%	54.1%	57.5%	60.1%	62.3%	**9.219**	**9.415**	**9.577**	**9.685**
X-MCMC (200)	**11.3%**	**9.1%**	8.3%	**7.1%**	**77.5%**	**78.9%**	**79.4%**	79.3%	8.832	9.141	9.391	9.559
X-MCMC-C (60)	12.6%	11.1%	**8.1%**	8.5%	75.5%	77.3%	79.3%	**80.0%**	8.781	9.018	9.202	9.372
X-MCMC-C (200)	19.0%	15.1%	13.6%	12.1%	69.6%	72.6%	74.0%	74.2%	9.097	9.279	9.453	9.577

Table 1: Results on One-Billion-Word test sets with different k. (Numbers in brackets refer to the number of time steps.)

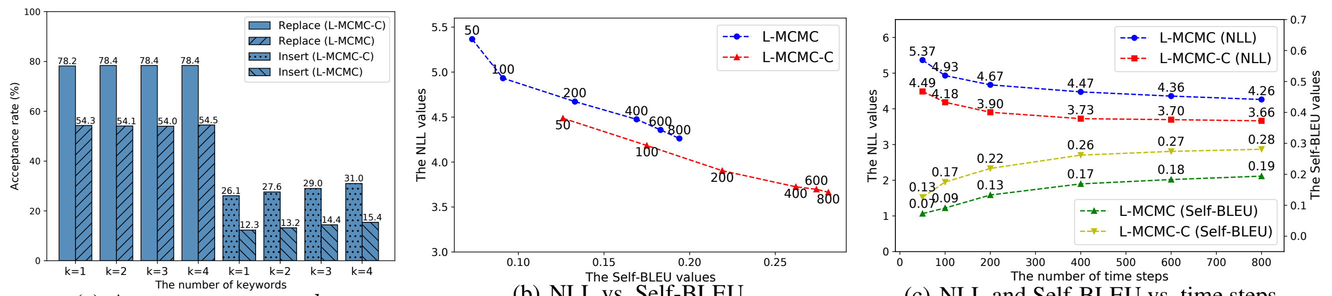
containing n-gram repetitions in Table 6 in the Appendix. In addition, we found that some generic sub-sequences (‘I am going to’, ‘he said’, ‘referring to’, ‘adding that’, etc.) frequently appear in the generated sentences. Both repetitive n-grams and generic sub-sequences will help these models to achieve low NLL and high BLEU scores. Therefore, it is insufficient to assess the generated sentences with NLL and BLEU. To complement them, we also automatically measured the diversity of each model’s generations.

Automatic Evaluation for Diversity. We used Self-BLEU (Zhu et al. 2018) to measure the diversity of each model’s generations. Self-BLEU evaluates how one sentence resembles the other generated ones. For each sentence, we treated it as a hypothesis and the others as references. We also used distinct n-gram (Li et al. 2016) and entropy (Zhang et al. 2018) to measure diversity. Distinct n-gram reflects the percentage of unique n-grams. Entropy further considers the distribution of n-grams. Lower Self-BLEU or higher distinct n-gram and entropy values indicate higher diversity. From Table 1, we can see that sep-B/F, asyn-B/F and GBS have higher Self-BLEU, lower distinct bigram and lower entropy scores, indicating they have low sample diversity, which is consistent with our analysis above. By comparison, MCMC-based models have high sample diversity.

When we ran L-MCMC-C for 50 steps, it achieved similar performance in sentence quality, and diversity to L-MCMC ran for 200 steps. Similarly, X-MCMC-C with 60 refinement steps is on par with X-MCMC with 200 refinement steps. We showed NLL and Self-BLEU achieved by L-MCMC and L-MCMC-C with different refinement steps when $k = 4$ in Figure 2(b). We can see that the proposed model (L-MCMC-C) requires much fewer refinement steps to achieve similar sentence quality and diversity compared to L-MCMC. The improvements achieved by our proposed models are mainly due to the classifier’s guidance, which reduces the number of invalid refinements. To verify this point, we showed the acceptance rates of LSTM-based models in Figure 2(a) and showed the acceptance rates of XLNet-based models in the Appendix. We can see that applying the classifier significantly improves the acceptance rates for replacement and insertion actions. In addition, adding the classifier brings a small amount of overhead. For example, the running time of the classifier only accounts for about 1/7 of X-MCMC-C.

When we ran L-MCMC-C and X-MCMC-C for 200 steps, NLL values declined dramatically. Meanwhile, Self-BLEU increased, and distinct bigram decreased. Increasing the number of time steps will improve sentence quality but degrade sentence diversity. It seems to indicate that sentence quality and diversity contradict each other. To further analyze this phenomenon, we showed the relationship between NLL and Self-BLEU when $k = 4$ in Figure 2(c). We can see that NLL values decrease with the increase of time steps, while Self-BLEU values increase for both models. Therefore, it is nontrivial to improve both sentence quality and diversity. Generic sentences with high probabilities favor sentence quality in terms of NLL but disfavor sentence diversity. Another advantage of our method is that we can make a trade-off between quality and diversity by controlling the number of time steps.

When comparing L-MCMC with X-MCMC (or L-MCMC-C with X-MCMC-C), we found the latter outper-



(a) Acceptance rates vs. k. (b) NLL vs. Self-BLEU. (c) NLL and Self-BLEU vs. time steps.

Figure 2: In subfigure (a), k is the number of keywords. In subfigure (b), numbers near lines denote the number of time steps.

forms the former in both sentence quality (NLL) and diversity (Self-BLEU and distinct bigram), mainly because the latter uses the pre-trained model as language models.

Human Evaluation. We also conducted a human evaluation to further compare these models. We selected 100 sentences for each model and invited three volunteers to assess the sentences in terms of fluency. Concretely, we first chose a group of sentences generated by different models and shuffled them to avoid bias. Then, the annotators ranked these sentences and assigned scores (0 to 1) to them, where 1 indicates the best quality. We showed the results of human evaluation in Table 1. Our proposed models (X-MCMC-C and L-MCMC-C) also outperform baselines in human evaluation. We also performed paired t-test comparisons between the proposed model (X-MCMC-C) and baselines, and p-values are less than 0.01, indicating the differences between the proposed model and baselines are statistically significant.

Ablation Study. We performed an ablation study to demonstrate the importance of each design. We mainly focused on two aspects: the effectiveness of the learned prior and the importance of the synthetic dataset. In Table 2, we compared L-MCMC-C (row 1) with five variants (rows 2-6). For a fair comparison, all models use LSTM-based language models and are run for 200 steps.

Removing the learned prior for actions (row 2) (actions are randomly sampled, but positions are still sampled from the learned prior) results in a slight increase in NLL, while removing the learned prior for positions (row 3) causes a sharp decline in performance, which is nearly as poor as L-MCMC (row 7). Therefore, the learned prior for positions is much more important than that of actions. Further, we used one-third of the synthetic dataset to train the classifier (row 4). The results of row 4 are slightly worse than those of row 1. Based on row 4, if we remove the random method and only use the masked LM (XLNet) to create the synthetic dataset (row 5), NLL values will increase marginally. By comparison, removing the masked LM (row 6) has a detrimental effect on performance. The random and masked LM methods are complementary, and both are indispensable for training the classifier.

To summarize, both the learned prior and the synthetic dataset play an essential role in our proposed model.

	Metrics	NLL (GPT-2) (↓)			
#	Models variants	k=1	k=2	k=3	k=4
1	**L-MCMC-C**	**3.762**	**3.773**	**3.819**	**3.904**
2	– prior for actions	3.850	3.861	3.893	3.932
3	– prior for positions	4.498	4.513	4.605	4.646
4	– size of dataset	3.921	3.950	4.015	4.030
5	– random	3.954	3.982	4.024	4.066
6	– masked LM	4.063	4.108	4.167	4.189
7	L-MCMC	4.549	4.564	4.617	4.672

Table 2: Ablation study on our model.

Constraints	**person**, **home**, **problems**, **depression**
GBS	A **person** familiar with the matter said : " The **problems** of **depression** are **home** to the people of the United States .
CGMH	The **person** 's head of **home** health **problems** have sparked **depression** .
L-MCMC-C	The average **person** who 's lost money since he was a child at **home** has no **problems** with **depression** .
X-MCMC-C	One **person** was sent **home** with mental health **problems** and severe **depression** .

Table 3: Sentences generated with lexical constraints.

Samples and Analysis. We showed some sentences generated by our proposed models and baselines in Table 3. Our proposed models can generate high-quality, lexically constrained sentences. As for GBS, the lexical constraint "home" conflicts with the previous tokens since GBS is not aware of the future lexical constraints when generating previous tokens. Therefore, forcing to incorporate the lexical constraint "home" degrades the quality of the generated sentence. The sentence generated with CGMH lacks coherence and fluency. Since CGMH refines the generated sentence randomly, it still needs more refinements. More generated sentences are shown in Table 8 in the Appendix.

Text Infilling

Experiment Setups and Baselines. We used $1,000$ sentences from the test set of One-Billion-Word corpus to create test sets for text infilling. Following (Liu et al. 2019a), we resorted to two mask strategies (random and middle) and three mask ratios (r = 25%, 50%, or 75%) to construct test sets for text infilling. The random mask strategy randomly removes r of tokens from the original sentence. The middle

Metrics	NLL (GPT-2) (↓)						BLEU (4-gram) (↑)					
Models	Middle			Random			Middle			Random		
	25%	50%	75%	25%	50%	75%	25%	50%	75%	25%	50%	75%
Human Reference	4.022	4.022	4.022	4.022	4.022	4.022	100%	100%	100%	100%	100%	100%
Template	-	-	-	-	-	-	65.1%	36.5%	9.0%	49.5%	16.1%	2.6%
FLM	4.417	4.479	4.223	4.434	4.727	4.543	68.4%	39.5%	12.2%	66.4%	31.9%	10.5%
BLM	4.453	4.460	4.130	4.545	4.860	4.620	68.3%	39.6%	12.8%	65.9%	31.2%	9.9%
F+B LM	4.288	4.355	4.130	4.309	4.605	4.457	**69.1%**	40.0%	12.5%	68.3%	33.4%	**10.9%**
Bayesian MCMC	4.188	4.268	4.242	4.164	4.310	4.349	68.7%	39.6%	12.3%	69.7%	35.4%	10.5%
TIGS (strategy 1)	4.985	5.515	5.865	5.183	5.911	6.268	65.8%	37.2%	9.9%	56.1%	21.8%	4.3%
TIGS (strategy 2)	4.416	4.341	4.042	4.638	5.018	4.852	67.9%	39.7%	12.3%	62.9%	28.5%	7.9%
L-MCMC	4.216	4.313	4.303	4.178	4.367	4.436	68.5%	39.7%	12.5%	69.0%	34.4%	10.1%
L-MCMC-C	**4.146**	**4.053**	**3.796**	**4.135**	**4.199**	**4.106**	69.0%	**40.3%**	**12.9%**	**70.0%**	**36.1%**	10.5%
L-MCMC-C (w/o BP)	-	-	-	-	-	-	70.9%	43.1%	14.6%	70.9%	37.6%	11.8%

Table 4: NLL and BLEU results for different mask strategies and rates. ("BP" refers to the brevity penalty.)

mask strategy removes r of tokens from the middle of the original sentence. Therefore, we have six types of test sets.

To compare our proposed model with previous work, we implemented several strong baselines. The forward language model (FLM) and the backward language model (BLM) fills the blanks from left to right, and from right to left with beam search (beam width = 10), respectively. F+B LM (Wang et al. 2016) fills the blanks by FLM and BLM and then selects the output with the lowest NLL. Bayesian MCMC (Berglund et al. 2015) initializes the blanks with random values and uses the replacement action to refine the filled values, but it can only generate sentences with a fixed length. We also implemented TIGS (Liu et al. 2019a), which iteratively refines the filled tokens with gradient descent. Before refining with TIGS, we resorted to two strategies to initialize the blanks with random values or values predicted by FLM with greedy search. All models use LSTM-based language models with the same structure (the setups for LSTM-based language models have been introduced in the first task). For a fair comparison, we ran 20 iterations for Bayesian MCMC, TIGS, L-MCMC, and L-MCMC-C.

Automatic Evaluation. Following previous work (Liu et al. 2019a), we also resorted to NLL and BLEU to automatically evaluate the infilled sentences. Similar to our first task, NLL is measured by GPT-2, and BLEU is computed between the infilled sentences and human references. BLEU measures how similar the infilled sentence is to the ground truth, while NLL assesses the fluency and coherence of the infilled sentences. We showed the results of NLL and the corpus-level BLEU (Papineni et al. 2002) in Table 4. Our proposed model achieves the lowest NLL scores in all cases, which means the infilled sentences of our proposed model are more fluent than those of previous methods. One advantage of our proposed model is that it does not need to know the number of blanks when infilling. In contrast, other baselines need to know the number of blanks before infilling. Therefore, our proposed models (L-MCMC and L-MCMC-C) may generate sentences with different lengths from the ground truths. Even though the BLEU algorithm with the brevity penalty will penalize our model when it generates shorter sentences, our model still outperforms baselines in most cases in terms of BLEU. Compared with TIGS, the proposed model does not need any initialization before infilling. In addition, TIGS is sensitive to initialization strategies. TIGS performs much better when initialized with the results of FLM, which may be unfair to other models. Compared with L-MCMC, L-MCMC-C significantly improves the performance in NLL and BLEU, mainly benefiting from the learned prior given by the classifier. We showed some infilled sentences in Table 9 in the Appendix.

Related Work

Pre-trained Language Models

Recently, many downstream NLP tasks are driven by large-scale pre-trained language models such as GPT (Radford 2018), GPT-2 (Radford et al. 2019), BERT (Devlin et al. 2019), ROBERTA (Liu et al. 2019c), ELECTRA (Clark et al. 2020), SpanBERT (Joshi et al. 2020), XLNet (Yang et al. 2019), MASS (Song et al. 2019) and BART (Lewis et al. 2020). However, these models cannot be directly applied to lexically constrained sentence generation.

Generating Sentences with Lexical Constraints

B/F LMs (Mou et al. 2015; Liu et al. 2019b) are limited to generating text with one lexical constraint. GBS (Hokamp and Liu, 2017) can generate sentences with multiple lexical constraints but degrades the generation quality and diversity. CGMH (Miao et al. 2019) revises candidate sentences randomly, causing many invalid operations. To solve this problem, we used an XLNet-based token-level classifier to guide MCMC-based models to refine the candidate sentence.

Conclusion

In this paper, we aim to reduce the redundant refinements conducted by previous MCMC-based models. To achieve this, we used a token-level classifier to instruct MCMC-based models where and how to refine the candidate sentence. Compared with previous MCMC-based approaches, our proposed model can iteratively refine the candidate sentence with the learned prior given by the pre-trained classifier. Experiment results show that our proposed model can generate fluent and diverse sentences for constrained sentence generation, outperforming all baselines.

Acknowledgements

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

References

Berglund, M.; Raiko, T.; Honkala, M.; Kärkkäinen, L.; Vetek, A.; and Karhunen, J. T. 2015. Bidirectional Recurrent Neural Networks as Generative Models. In *NIPS*, 856–864.

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*, 4171–4186.

Geman, S.; and Geman, D. 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on pattern analysis and machine intelligence* 6(6): 721–741.

Hastings, W. K. 1970. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika* 57(1): 97–109.

He, X.; Xu, H.; Li, J.; He, L.; and Yu, L. 2017a. FastBTM: Reducing the Sampling Time for Biterm Topic Model. *Knowledge-Based Systems* 132: 11–20.

He, X.; Xu, H.; Sun, X.; Deng, J.; Bai, X.; and Li, J. 2017b. Optimize Collapsed Gibbs Sampling for Biterm Topic Model by Alias Method. In *IJCNN*, 1155–1162.

Hokamp, C.; and Liu, Q. 2017. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In *Proceedings of ACL*, 1535–1546.

Holtzman, A.; Buys, J.; Forbes, M.; and Choi, Y. 2020. The Curious Case of Neural Text Degeneration. In *ICLR*.

Joshi, M.; Chen, D.; Liu, Y.; Weld, D. S.; Zettlemoyer, L.; and Levy, O. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics* 8(5): 64–77.

Kudo, T.; and Richardson, J. 2018. SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing. In *Proceedings of EMNLP*, 66–71.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of ACL*, 7871–7880.

Li, J.; Galley, M.; Brockett, C.; Gao, J.; and Dolan, W. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. In *Proceedings of NAACL*, 110–119.

Liu, D.; Fu, J.; Liu, P.; and Lv, J. 2019a. TIGS: An Inference Algorithm for Text Infilling with Gradient Search. In *Proceedings of ACL*, 4146–4156.

Liu, D.; Fu, J.; Qu, Q.; and Lv, J. 2019b. 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.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019c. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* .

Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; and Teller, E. 1953. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* 21(6): 1087–1092.

Miao, N.; Zhou, H.; Mou, L.; Yan, R.; and Li, L. 2019. CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling. In *Proceedings of AAAI*, 6834–6842.

Mou, L.; Song, Y.; Yan, R.; Li, G.; Zhang, L.; and Jin, Z. 2016. Sequence to Backward and Forward Sequences: A Content-Introducing Approach to Generative Short-Text Conversation. In *Proceedings of COLING*, 3349–3358.

Mou, L.; Yan, R.; Li, G.; Zhang, L.; and Jin, Z. 2015. Backward and Forward Language Modeling for Constrained Sentence Generation. *arXiv preprint arXiv:1512.06612* .

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, 311–318.

Post, M.; and Vilar, D. 2018. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In *Proceedings of NAACL*, 1314–1324.

Radford, A. 2018. Improving Language Understanding by Generative Pre-Training. *Technical Report, OpenAI* .

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. *Technical Report, OpenAI* .

Song, K.; Tan, X.; Qin, T.; Lu, J.; and Liu, T. 2019. MASS: Masked Sequence to Sequence Pre-training for Language Generation. In *ICML*, 5926–5936.

Su, J.; Xu, J.; Qiu, X.; and Huang, X. 2018. Incorporating Discriminator in Sentence Generation: a Gibbs Sampling Method. In *Proceedings of AAAI*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NIPS*, 5998–6008.

Wang, A.; and Cho, K. 2019. BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, 30–36.

Wang, C.; Yang, H.; Bartz, C.; and Meinel, C. 2016. Image Captioning with Deep Bidirectional LSTMs. In *Proceedings of the 24th ACM international conference on Multimedia*, 988–997.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.;

and Brew, J. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv preprint arXiv:1910.03771* .

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J. G.; Salakhutdinov, R.; and Le, Q. V. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NIPS*, 5753–5763.

Zhang, Y.; Galley, M.; Gao, J.; Gan, Z.; Li, X.; Brockett, C.; and Dolan, W. 2018. Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization. In *NIPS*, 1810–1820.

Zhu, W.; Hu, Z.; and Xing, E. 2019. Text Infilling. *arXiv preprint arXiv:1901.00158* .

Zhu, Y.; Lu, S.; Zheng, L.; Guo, J.; Zhang, W.; Wang, J.; and Yu, Y. 2018. Texygen: A Benchmarking Platform for Text Generation Models. In *SIGIR*, 1097–1100.