Effective Unsupervised Constrained Text Generation based on Perturbed Masking

Anonymous ACL submission

Abstract

Unsupervised constrained text generation aims to generate text under a given set of constraints without any supervised data. Current state-of-the-art methods stochastically sample edit positions which may cause unnecessary search steps. In this paper, we propose PMCTG to improve effectiveness by searching for the best position and action in each step. Specifically, PMCTG extends the perturbed masking technique to effectively search for the best edit position. Then it uses proposed multi-aspect scoring functions to select edit action to further reduce search difficulty. Since PMCTG does not require supervised data, it can extend to different generation tasks. We show PMCTG achieves state-of-the-art results in keywords-to-sentence generation and paraphrasing.

1 Introduction

Constrained text generation is the task of generating text that satisfies a given set of constraints, and it serves many real-world text generation applications, such as dialogue generation (Li et al., 2016) and summarization (See et al., 2017). There are broadly two types of constraints: (1) Hard constraints such as including a set of given words or phrases in the generated text. Example 1 in Table 1 shows that the keywords "*You*" and "*beautiful*" must occur in the generated sentence. (2) Soft constraints such as acquiring the generated text to be semantically similar to the original text. Example 2 in Table 1 shows a pair of paraphrases where "*What are the effective ways to learn cs?*" and "*How to learn cs effectively?*" share a similar meaning.

Conventional approaches model the task as an encoding-decoding problem with a supervised setting (Prakash et al., 2016; Gupta et al., 2018). However, these methods have certain shortcomings for two constrained generation tasks. For hard constrained text generation, without external constrained means, it is difficult for these methods to guarantee that the generated text can satisfy all constraints. For soft constrained tasks, conventional methods treat it as a machine translation (MT) task (Sutskever et al., 2014) and require massive parallel supervised data for training. However, constructing such datasets is resource-intensive. Besides, the domain-specific supervised models may be difficult to transfer to new domains. (Li et al., 2019).

No.	Original Text	Generated Text
1	You, beautiful	You are so beautiful .
2	How to learn cs effectively?	What are the effective ways to learn cs?

Table 1: Examples on constrained text generation.

Unsupervised text generation is an effective solution to address the above challenges. There are recently two research directions: Beam search-based method aims to generate candidates in order from left to right that satisfy the constraints in each step, inspired by MT (Hokamp and Liu, 2017; Post and Vilar, 2018). However the search space of MT systems is relatively small, and when applied to other generation tasks, such as paraphrase, the beam search-based approach does not work as optimally as expected because the search space is too large (Sha, 2020). Local edit-based method represented by CGMH (Miao et al., 2019) and USPA (Liu et al., 2020) is another effective solution. These methods propose stochastic local edit strategies to search for reasonable sentences in a huge search space based on the given constraints. One main concern is that these methods take a long time to search for the optimal solution because they are based on stochastic strategies. Intuitively, they need more search steps to converge. G2LC (Sha, 2020) tries to use gradients to determine edit positions and actions to improve search effectiveness. But it still relies on supervised data.

Dedicated to improving the local edit-based

methods, this paper proposes a framework PMCTG (**P**erturbed **M**asking for **C**onstrained **T**ext **G**eneration) for constrained text generation. PMCTG focuses on controlling the search direction and reducing the number of search steps by searching for the best edit position and the best edit action. Specifically, PMCTG extends perturbed masking (Wu et al., 2020) from the pre-trained BERT model (Devlin et al., 2019) to find the edit position in the sequence. Perturbed masking aims to estimate the correlation between words in a sequence, which can be naturally used to find the edit location. We also propose a series of scoring functions for different tasks to select the edit action. PMCTG does not rely on supervised data and only needs a pre-trained BERT model to perform perturbed masking.

We evaluate PMCTG in two constrained text generation tasks, keywords-to-sentence generation and paraphrasing. Experimental results show that PMCTG achieves competitive performance compared to multiple baselines. In summary, the contributions are as follows:

1. We extend perturbed masking to constrained text generation to find edit positions more effectively.
2. We design different scoring functions to select the best action effectively. With different scoring functions, PMCTG can be extended to various generation tasks (Kikuchi et al., 2016; Ficler and Goldberg, 2017; Hu et al., 2017).
3. We demonstrate our method's state-of-the-art performance in keywords-to-sentence generation and paraphrasing tasks.

2 Related Work

2.1 Constrained Text Generation

Constrained text generation is formulated as a supervised sequence-to-sequence problem under the encoding-decoding paradigm (Sutskever et al., 2014). For example, (Prakash et al., 2016) and (Li et al., 2019) respectively propose a stacked residual LSTM network and a transformer-based model (Vaswani et al., 2017), and (Gupta et al., 2018) propose to leverage a combination of variational autoencoders (VAEs) with LSTM models to generate paraphrases. A new sentence generation language model is proposed by (Guu et al., 2018), where a prototype sentence is first extracted from the training corpus and then edited into a new sentence. However, these methods do not allow the integration of constraints (Miao et al., 2019). Some works have attempted to add constraints on the generated models. (Wuebker et al., 2016) and (Knowles and Koehn, 2016) utilize prefixes to guide the generation of the target text. (Mou et al., 2016) use pointwise mutual information (PMI) to predict a keyword and treat it as a constraint to generate target text. However, these methods always bind the constraints to the original model and are therefore difficult to apply to new domains and new generation models (Li et al., 2019). Moreover, the above approaches rely on an adequate parallel supervised corpus, which is hard to obtain in real-world application scenarios.

Unsupervised constrained text generation has become a popular research direction due to the low training cost and the mitigation of insufficient training data. VAEs and their variants (Bowman et al., 2016; Roy and Grangier, 2019) are leveraged to generate sentences from a continuous latent space. These methods can effectively get rid of the reliance on supervised datasets but remain difficult to control and incorporate generative constraints.

Beam search is a representative direction for unsupervised constrained text generation. Grid Beam Search (GBS) (Hokamp and Liu, 2017) is an algorithm that extends beam search by allowing the inclusion of pre-specified lexical constraints. (Post and Vilar, 2018) propose Dynamic Beam Allocation (DBA), a much faster beam search-based method with hard lexical constraints. (Zhang et al., 2020) propose an insertion-based approach consisting of insertion-based generative pre-training and inner-layer beam search. For the tasks where the search space is limited (represented by machine translation), such methods work well. However, when faced with a large search space, they do not work as optimally (Sha, 2020).

Local edit-based methods have attracted attention recently, as they can be applied to reduce search spaces. CGMH (Miao et al., 2019) applies the Metropolis-Hastings algorithm (Metropolis et al., 1953) to unsupervised constrained generation. UPSA (Liu et al., 2020) is another local edit-based method. It directly models paraphrasing as an optimization problem and uses simulated annealing to solve it. However, these models require more steps and running time to generate reasonable sentences since they are based on stochastic strategies. (Sha, 2020) proposes a gradient-guided method G2LC that uses the gradient of tokens to determine the edit actions and positions, making

the generation process more controllable. However, a problem with G2LC is that it still relies on the supervised corpus to train a binary classification model to serve their semantic similarity objective.

2.2 Perturbed Masking

Perturbed masking (Wu et al., 2020) is a parameter-free probing technique to analyze and interpret pre-trained models. It introduces the perturbed masking technique based on a pre-trained BERT-based model with masked language modeling (MLM) objective to measure the impact a word has on predicting another word. It is originally used in syntax-based tasks such as syntactic parsing and discourse dependency parsing. We extend perturbed masking to constrained text generation.

3 Methodology

In this section, we would like to introduce the proposed model PMCTG by first introducing the specific process of using perturbed masking to select edit positions, and then explaining the proposed scoring functions and the use of them to select the edit actions.

3.1 Edit Position Selection

Most previous works select edit locations stochastically, which lead to many unnecessary search steps. To reduce the number of search steps, we propose to use perturbed masking (Wu et al., 2020) to sample the edit position.

Background. The perturbed masking technique is proposed to assess the inter-word information (i.e., the impact one word has on another word in a sequence) based on masked language modeling (MLM). It is originally used for dependency parsing.

Formally, given a sequence with n tokens $\boldsymbol{x} = \{x_i\}_{i=1}^{n}$ and a pre-trained BERT-based model(Devlin et al., 2019) trained with MLM objective, we obtain contextual representations for each token $H(\boldsymbol{x})_i$. To quantify the impact a token x_j has on another token x_i, we conduct the following three-step calculation:

1. Replace x_i with $[MASK]$ token and feed the new sequence $\boldsymbol{x} \backslash x_i$ into BERT, a contextual representation denoted as $H(\boldsymbol{x} \backslash \{x_i\})_i$ for x_i is obtained.
2. Replace x_i and x_j with $[MASK]$ token and feed the new sequence $\boldsymbol{x} \backslash \{x_i, x_j\}$ into BERT, another contextual representation denoted as $H(\boldsymbol{x} \backslash \{x_i, x_j\})_i$ for x_i is obtained.
3. Given the distance metric $d(,)$, compute the difference between two vectors $I(\boldsymbol{x}|x_j, x_i) = d(H(\boldsymbol{x} \backslash \{x_i\})_i, H(\boldsymbol{x} \backslash \{x_i, x_j\})_i)$. In this paper, we leverage cosine similarity as our distance metric.

$I(\boldsymbol{x}|x_j, x_i)$ indicates the impact x_j has on x_i, where a higher value indicates a lower impact, and vice versa. Intuitively, if $H(\boldsymbol{x} \backslash \{x_i\})_i$ and $H(\boldsymbol{x} \backslash \{x_i, x_j\})_i$ are similar, it means that the presence or absence of x_j has little effect on the prediction of x_i, thus reflecting the low importance of x_j to x_i.

Position Selection. It is natural to apply perturbed masking to select the edit position for constrained text generation. Based on perturbed masking technique, we compute the edit score for each token in the sequence and then sample the token with the highest score to edit. The token with minimal impact on its adjacent tokens indicates that it has the weakest correlation with adjacent words and therefore requiring edit. We add the special tokens $[CLS]$ and $[SEP]$ to the original sentence and then use the pre-trained BERT to calculate the edit score for each token:

$$ES_i = 1 - \frac{1}{2}(I(\boldsymbol{x}|x_i, x_{i+1}) + I(\boldsymbol{x}|x_i, x_{i-1})) \tag{1}$$

Then we can get an edit score vector $\boldsymbol{ES} = \{ES_i\}_{i=0}^{n}$. Later, we feed it into a softmax layer and obtain the edit probabilities:

$$p_i^{edit} = \frac{exp(ES_i)}{\sum_j exp(ES_j)} \tag{2}$$

After that, the $\boldsymbol{p^{edit}}$ is utilized as the weights to sample the edit position x_e in $\boldsymbol{x}$ where e indicates the edit position index.

3.2 Edit Action Selection

After sampling the edit position, next we need to determine the edit action. The edit three actions we focus on are: insert, replace and delete. Specifically, our strategy in this step is to pre-implement the three actions first and then sample the actions based on their action scores. When scoring insertion action, we simply make the equal probability of the front or back of the position for token insertion. We first introduce the scoring functions

for different tasks and then explain the edit action selection based on the action scores.

3.2.1 Scoring Function Design

We propose multiple scoring functions to improve generated text. Given the initial sentence $\boldsymbol{x_0} = \{x_{0,1}, x_{0,2}, \ldots, x_{0,n}\}$ with n tokens and the generated sentence $\boldsymbol{x_*} = \{x_{*,1}, x_{*,2}, \ldots, x_{*,m}\}$ with m tokens, the scoring functions include fluency, editorial rationality, semantic similarity and diversity.

Fluency. The primary condition for a reasonable sentence is fluency, thus we use the average negative log-likelihood to estimate a sentence's fluency based on a forward language model. The score is calculated as:

$$S_{flu}(\boldsymbol{x_*}) = -\frac{1}{m}\sum_{i=1}^{m} logp_{LM}(x_{*,i}|x_{*,<i}) \quad (3)$$

Editorial Rationality. Since the sentence generation process is based on local edits, we further use perturbed masking to design a local edit score for different actions to evaluate their rationality. After a replacement action is executed at index i in $\boldsymbol{x_0}$, we obtain the sentence $\boldsymbol{x_*} = \{x_{*,1}, x_{*,2}, \ldots x_{*,i-1}, x', x_{*,i+1}, \ldots, x_{*,n}\}$, where x' is the replaced token and $m = n$. Then we define the edit score as:

$$S_{edit}(\boldsymbol{x_*}) = \frac{1}{2}(I(\boldsymbol{x_*}|x', x_{0,i+1}) + I(\boldsymbol{x_*}|x', x_{0,i-1})) \quad (4)$$

Similarly, after an insertion action, we obtain $\boldsymbol{x_*} = \{x_{*,1}, x_{*,2}, \ldots x_{*,i}, x', x_{*,i+1}, \ldots, x_{*,n}\}$, where x' is the inserted token and $m = n + 1$. The edit score is calculated as:

$$S_{edit}(\boldsymbol{x_*}) = \frac{1}{2}(I(\boldsymbol{x_*}|x', x_{0,i+1}) + I(\boldsymbol{x_*}|x', x_{0,i})) \quad (5)$$

After a deletion action, we obtain $\boldsymbol{x_*} = \{x_{*,1}, x_{*,2}, \ldots x_{*,i-1}, x_{*,i+1}, \ldots, x_{*,n}\}$, where $m = n - 1$. The edit score calculated for deletion is a little different from replacement and insertion action:

$$S_{edit}(\boldsymbol{x_*}) = \frac{1}{2}(I(\boldsymbol{x_*}|x_{0,i-1}, x_{0,i+1}) + I(\boldsymbol{x_*}|x_{0,i+1}, x_{0,i-1})) \quad (6)$$

Semantic Similarity. The semantic similarity consists of keyword similarity and sentence similarity. We use KeyBERT (Grootendorst, 2020) to extract the keyword set K from $\boldsymbol{x_0}$. And the pre-trained BERT is leveraged to encode $\boldsymbol{x_0}$ and $\boldsymbol{x_*}$, where $ik = idx(k)$ indicates the index of keyword k in $\boldsymbol{x_0}$. The keyword similarity is defined as finding the closest word in $\boldsymbol{x_*}$ by computing their cosine similarity:

$$S_{sem,key}(\boldsymbol{x_*}, \boldsymbol{x_0}) = \frac{1}{|K|}\sum_{k\in K} \underset{i}{max}(cos(H(\boldsymbol{x_0})_{ik}, H(\boldsymbol{x_*})_i)) \quad (7)$$

As for the sentence similarity into account, assuming that $H(x)$ indicates the $[CLS]$ representation in x from BERT and is leveraged to presents the whole sentence (Devlin et al., 2019), we define the sentence similarity $S_{sem,sen}(\boldsymbol{x_*}, \boldsymbol{x_0})$ as:

$$S_{sem,sen}(\boldsymbol{x_*}, \boldsymbol{x_0}) = cos(H(\boldsymbol{x_0}), H(\boldsymbol{x_*})) \quad (8)$$

Altogether, the semantic similarity score is:

$$S_{sem}(\boldsymbol{x_*}, \boldsymbol{x_0}) = S_{sem,key}(\boldsymbol{x_*}, \boldsymbol{x_0}) + S_{sem,sen}(\boldsymbol{x_*}, \boldsymbol{x_0}) \quad (9)$$

Diversity. Followed (Liu et al., 2020), a BLEU-based (Papineni et al., 2002) function is adopted to evaluate the expression diversity of the original and generated sentence.

$$S_{exp}(\boldsymbol{x_*}, \boldsymbol{x_0}) = (1 - BLEU(\boldsymbol{x_*}, \boldsymbol{x_0})) \quad (10)$$

3.2.2 Action Scoring

As mentioned above, after sampling the edit position i, we need to determine the edit action by re-implementing three actions and sampling the actions based on their action scores. We generate the inserted and replaced candidate x' from a language model such as LSTM (Hochreiter and Schmidhuber, 1997) and GPT2 (Radford et al., 2019).

$$p_{candidate} = p_{LM}(x_{0,i}|x_{0,<i}) \quad (11)$$

We use $p_{candidate}$ as weights to sample x'.After obtaining the edit position i and candidate x', we need to calculate the edit score for each action. We adopt S_{flu} and S_{edit} as our scoring function for keywords-to-sentence generation:

$$S_{hard}(\boldsymbol{x_*}) = \lambda_{flu}S_{flu} + \lambda_{edit}S_{edit} \quad (12)$$

and S_{flu}, S_{sem}, S_{exp} and S_{edit} for paraphrasing:

$$S_{soft}(\boldsymbol{x_*}) = \lambda_{flu}S_{flu} + \lambda_{edit}S_{edit} + \lambda_{sem}S_{sem} + \lambda_{exp}S_{exp} \quad (13)$$

Notably, since different scores are in different magnitudes, they need to be normalized to avoid the dominance of one type of the score. After scoring different actions, we use the scores as weights to sample the edit action.

3.3 Overall Searching Process

With $\boldsymbol{x_0}$ (given keywords in the keywords-to-sentence generation task or original sentence in the paraphrasing task) as input, we repeat the above steps including edit position selection with perturbed masking and edit action selection with scoring functions for local edit. Until the maximum searching steps, we choose the sentence that achieves the highest score as the final output, according to (12) for keywords-to-sentence generation task or (13) for paraphrasing task respectively.

4 Experiments

We evaluate our method on two constrained text generation tasks, namely keywords-to-sentence generation, and paraphrasing.

4.1 Keywords-to-Sentence Generation

Experimental Setting. Keywords-to-Sentence generation aims to generation a sentence containing the given keywords which is a representative hard constrained text generation task. We conduct keywords-to-sentence generation experiments on the One-Billion-Word dataset[1] (Chelba et al., 2014) . Two language models for generation, namely two-layer LSTM (followed as (Miao et al., 2019; Sha, 2020)) and GPT2 (Radford et al., 2019), are evaluated. We randomly sample 5 million sentences for pre-training BERT-based-cased and GPT2 for domain adaption and hold out 3 thousand sentences as the test set.

As for hyperparameters, for each test sentence, we randomly sample 1 to 4 keywords as hard constraints. The maximum searching step set in this task is 100. And λ_{flu} and λ_{edit} are set as 1 in equation (12). Besides, when the keyword indexes are sampled as edit positions, we directly conduct insert action since the keywords cannot be replaced and deleted.

As for evaluation metrics, the generated target sentence is measured by negative log-likelihood (NLL) loss. NLL is given by a third-party language mode which is an n-gram Kneser-Ney language model (Heafield, 2011) trained in a monolingual English corpus from WMT18[2]. In addition to automatic evaluation metrics, we also introduce human evaluation. Specifically, we invite 3 experts who are fluent English speakers to score the generated sentences according to their quality. The score ranges from 0 to 1 with an accuracy of two decimal places, where 1 indicates the best score. The automatic and human evaluation criteria are consistent with previous works (Sha, 2020). The scoring guideline is shown in Appendix a.

Baseline. We compare our method with several advanced methods:

- **sep-B/F** (Mou et al., 2016) is a variant of the backward forward model. In sep-B/F, the backward and forward sequences respectively behind and after the keyword are generated separately. It only supports only one keyword.
- **asyn-B/F** (Mou et al., 2016) is similar to sep-B/F. The difference is that the two sequences are generated asynchronously, i.e., the backward sequence is first generated, and then the forward sequence is generated based on the backward one.
- **GBS** (Hokamp and Liu, 2017) is a searching approach that aims to search for a valid solution in the constrained search space of the generator with grid beam search.
- **DBA** (Post and Vilar, 2018) is another beam search-based approach with a higher search speed.
- **CGMH** (Miao et al., 2019) is a stochastic search method based on Metropolis-Hastings sampling.
- **G2LC** (Sha, 2020) is a gradient-guided approach. It improves CGMH by leveraging gradient to decide the edit positions and actions.

Automatic and Human Evaluation Results. Table 2 shows the performance of multiple methods on keywords-to-sentence generation task. Among different kinds of methods, we can see that the local edit-based methods work better than beam search-based methods, indicating their superior searching ability. CGMH can narrow the search range and make it easy to find higher-quality sentences. G2LC and PMCTG outperform CGMH, which illustrates the importance of determining the correct edit position and action for each step. Exploration and strategies for these two issues can better guide the model to find a more optimal solution, while also greatly reducing the waste of potentially non-essential search steps. Overall, the proposed PMCTG model outperforms other methods on average in both automatic and human evaluation metrics. PMCTG utilizes perturbed masking technology to

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

[2] http://www.statmt.org/wmt18/translation-task.html

Models	NLL					Score (Human Evaluation)				
	1	2	3	4	avg	1	2	3	4	avg
seq-B/F	7.80	/	/	/	/	0.11	/	/	/	/
asyn-B/F	8.30	/	/	/	/	0.09	/	/	/	/
GBS	7.42	8.72	8.59	9.63	8.59	0.32	0.55	0.49	0.55	0.48
DBA	7.41	8.58	8.54	9.25	8.45	0.43	0.53	0.54	0.59	0.52
CGMH	7.04	7.57	8.26	7.92	7.70	0.45	0.61	0.56	0.65	0.57
G2LC	7.02	7.46	8.01	7.76	7.56	0.47	**0.73**	0.65	0.67	0.63
PMCTG-GPT2	6.98	7.45	**7.69**	7.89	7.50	0.51	0.68	**0.70**	0.72	0.65
PMCTG-LSTM	**6.92**	**7.33**	7.93	**7.68**	**7.47**	**0.53**	0.69	0.68	**0.74**	**0.66**

Table 2: Performance on keywords-to-sentence generation task. Lower NLL and higher score indicate better result. 1,2,3 and 4 present the keyword numbers and avg indicates the average score.

Keywords	Sentences
worried	We are very worried about there .
agreement	To achieve such an agreement , it is important .
competition, action	The shots of competition and action are on display here .
change, hours	This will change it in the next 24 hours .
The,greatest, court	The world's greatest size court will be presented to you .
I,things, him	I can do lots of things for him .
body, advanced, July,funeral	The body was found advanced in July and funeral were held in September .
Miley,more, final,spots	But Miley Cyrus has played more than three times in the finaltwo spots .

Table 3: Generated examples of PMCTG-LSTM in keywords-to-sentence generation task.

identify edit locations and reflect the reasonableness of edit actions more intuitively and practically. Moreover, PMCTG shows its effectiveness by using fewer or equal search steps to achieve better generation results (The maximum search steps set in CGMH and G2LC are 200 and 100 respectively). Interestingly, PMCTG-LSTM seems to be superior to PMCTG-GPT2 in this task, we believe that since keywords are locally ill-formed and semantically distant, the information of keywords may be difficult to support GPT2 to generate reasonable candidates without taking backward probability into account. In contrast, the two-layer LSTM considers both forward and backward probabilities and may be more suitable for generating candidates between two less correlated tokens.

We find that more keywords may lead to better results, one possible reason is that more keywords can further narrow the search space and facilitate the search of the model.

Case Study. Some generated examples of PMCTG-LSTM are shown in Table 3. We observe that the proposed model can generate fluent and meaningful sentences while containing the given keywords.

4.2 Paraphrasing

Experimental Setting. Paraphrasing aims to convert a sentence to a different surface form but with the same meaning. We evaluate PMCTG on two paraphrase datasets, namely Quora[3] and Wikianswers (Fader et al., 2013). The Quora question pair dataset consists of 140 thousand parallel sentences pairs and 640 thousand non-parallel sentences. Following previous works (Liu et al., 2020), we randomly sample 20 thousand sentences as the test set. The Wikianswers dataset contains 2.3 million question pairs scrawled from the Wikipedia website. We also conduct an experiment on two-layer LSTM (followed as (Miao et al., 2019; Liu et al., 2020; Sha, 2020)) and GPT2 for better comparison. Following previous works (Liu et al., 2020) again, we randomly sample 20 thousand sentences respectively in two datasets as test sets and used the other sentences to continually pre-train BERT-based-cased and GPT2 for domain adaption.

As for hyperparameters, the maximum searching step set in this task is 50 and λ are all set as 1 in equation (13).

In terms of evaluation metrics, we leverage the representative metrics sentence-level BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) as the basic metrics. In addition, as stated in (Sun

[3]http://www.statmt.org/wmt18/translation-task.html

Models	Quora				Wikianswer			
	iBLEU	BLEU	R1	R2	iBLEU	BLEU	R1	R2
ResidualLSTM	12.67	17.57	59.22	32.40	22.94	27.36	48.52	18.71
VAE-SVG-eq	15.17	20.04	59.98	33.30	26.35	32.98	50.93	19.11
Pointer-generator	16.79	22.65	61.96	36.07	31.98	39.36	57.19	25.38
Transformer	16.25	21.73	60.25	33.45	27.70	33.01	51.85	20.70
Transformer+Copy	17.98	24.77	63.34	37.31	31.43	37.88	55.88	23.37
DNPG	18.01	25.03	67.73	37.75	34.15	41.64	57.32	25.88
Pointer-generator	5.04	6.96	41.89	12.77	21.87	27.94	53.99	20.85
Transformer+Copy	6.17	8.15	44.89	14.79	23.25	29.22	53.33	21.02
Shallow fusion	6.04	7.95	44.87	14.79	22.57	29.76	53.54	20.68
MTL	4.90	6.37	37.64	11.83	18.34	23.65	48.19	17.53
MTL + Copy	7.22	9.83	47.08	19.03	21.87	30.78	54.1	21.08
DNPG	10.39	16.98	56.01	28.61	25.60	35.12	56.17	23.65
VAE	8.16	13.96	44.55	22.64	17.92	24.13	31.87	12.08
CGMH	9.94	15.73	48.73	26.12	20.05	26.45	43.31	16.53
UPSA	12.02	18.18	56.51	30.69	24.84	32.39	54.12	21.45
G2LC-Recognizer	14.34	20.13	58.90	32.79	/	/	/	/
G2LC-Generator	14.46	23.27	**59.65**	33.08	/	/	/	/
PMCTG-LSTM	14.79	23.73	59.21	31.92	25.66	33.87	56.21	21.92
PMCTG-GPT2	**15.22**	**24.37**	59.03	**32.89**	**26.13**	**35.02**	**56.89**	**23.21**

Table 4: Performance on paraphrasing task. R1 and R2 respectively indicate ROUGE1 and ROUGE2.

and Zhou, 2012), standard BLEU and ROUGE could not reflect the diversity between the generated and original sentences. Therefore, we adopt iBLEU (Sun and Zhou, 2012) which penalize the generated sentences with high similarity with the original ones as an additional evaluation metric. Besides, we also invite experts to evaluate the generated paraphrases. Specifically, we sample 300 sentences from the Quora test set and ask 3 experts to score each sentence according to two aspects: relevance and fluency. The evaluation criterion is again consistent with the previous works (Miao et al., 2019; Liu et al., 2020). The scoring guideline is shown in Appendix b.

Baseline. We compare our methods with three baselines:

- **Supervised methods** are original sequence-to-sequence models trained in in-domain supervised data, including ResidualLSTM (Prakash et al., 2016), VAE-SVG-eq (Gupta et al., 2018), Pointer-generator (See et al., 2017), the Transformer (Vaswani et al., 2017), and DNPG (the decomposable neural paraphrase generation) (Li et al., 2019).
- **Domain-adapted supervised methods** train models in one domain and then adapt the models to another domain, including shallow fusion (Gülçehre et al., 2015) and a multi-task learning (MTL) method (Domhan and Hieber, 2017).
- **Unsupervised methods** that are free of any supervised data and easily adapted to multiple new domains, including VAE (Kingma and Welling, 2014), CGMH (Miao et al., 2019), UPSA (Liu et al., 2020), and the recurrent state-of-the-art method G2LC (Sha, 2020). Notably, G2LC has two variants of G2LC-Generator and G2LC-Recognizer.

Automatic Evaluation Results. Table 4 presents the results of multiple methods on paraphrasing tasks. From the first part of Table 4, we can see that supervised methods significantly outperform the other two kinds of methods. The supervised models were trained on 100 thousand question pairs for Quora and 500 thousand question pairs for Wikianswers. Their superiority indicates the effectiveness of learning knowledge from massive parallel data. However, such in-domain supervised data is hard to obtain in real-world applications.

Besides, the second section of Table 4 shows the domain-adapted supervised models' performance. These models are trained in one domain (Quora or Wikianswers) and then evaluated in another domain (Wikianswers or Quora). Their performances

Method	Relevance	Fluency
VAE	0.53	0.64
CGMH	0.62	0.70
UPSA	0.75	0.73
G2LC(Recognizer)	0.79	0.77
G2LC(Generator)	**0.81**	0.78
PMCTG-GPT2	0.76	**0.81**

Table 5: Human evaluation results on paraphrasing.

Type	Sentence
Ori	what can make physics easy to learn?
Gen	how to learn physics easily?
Ref	how can you make physics easy to learn?
Ori	is it possible to pursue many different things in life?
Gen	is it good to buy many different things in life?
Ref	how do i refuse to choose between different things to do in my life?
Ori	how do i choose a journal to publish my paper?
Gen	how do you choose a journal to publish your first book?
Ref	where do i publish my paper?
Ori	where can i get free books to read or download?
Gen	where did i download free books to read?
Ref	where can i get free books?

Table 6: Generated examples of PMCTG-GPT2 in paraphrasing task.

are much lower than in-domain supervised models' performances. This demonstrates the poor generalizability of supervised models and calls for the need for unsupervised methods.

The last section of Table 4 shows the results of multiple unsupervised methods. VAE seems to work worst on both datasets, which suggests that paraphrasing by latent space sampling performs not as well as local edit methods. PMCTG achieves the best performance in most cases, which indicates the effectiveness of PMCTG again. Unsupervised PMCTG does not require parallel data and can easily generalize to new domains, thus some unsupervised methods tend to achieve higher performance than the domain-adapted supervised models. In addition, it is worthwhile to note that the performance of some unsupervised methods (UPSA, G2LC, and PMCTG) is even better than some supervised methods (Residual LSTM and VAE-SVG-eq), which indicates that the gap between supervised and unsupervised methods has narrowed due to the effective searching strategies of the local edit-based methods. In addition, different from the keywords-to-sentence generation task, GPT2 works better than two-layer LSTM in the paraphrasing task. We believe that given a partially fluent text, GPT2 can generate more reasonable candidates due to its powerful language model.

Human Evaluation Results. From Table 5, we show PMCTG-GPT2 achieves state-of-the-art performance in terms of fluency, but still suffers from relevance. We plan to improve its relevance in future research.

Case Study. Table 6 lists some representative generated examples from PMCTG-GPT2. They show the four most common types of paraphrasing for the proposed method. The first type is the change of syntax such as the interchange of "*what can...*" and "*how to...*" as in the first example. The second type is the change of adjective such as the second example where the "*possible*" is changed into "*good*". The third type is the change of personal pronouns such as the interchange of "*you*" and "*I*" in the third example. The last type is the change of tense, the most common is the interchange of general past tense and general present tense as the last example. In general, one limitation of the proposed model is the relatively low expressive diversity of generated sentences. One possible reason is that since each search step modifies only one token, and the unit of conversion from one expression to another is usually phrases or sentence blocks, thus the model may be biased not to search in that direction.

5 Conclusion

We propose a method PMCTG to improve the previous stochastic searching methods in the topic of unsupervised constrained generation. PMCTG leverages perturbed masking technique to find the best edit position and leverages newly designed multiple scoring functions to decide the best edit action. We evaluate the proposed method on two representative tasks: keywords-to-sentence generation (hard constraints) and paraphrasing (soft constraints). Experimental results demonstrate the effectiveness of the proposed method which achieves competitive results on three datasets over multiple advanced baseline methods. We plan to improve the diversity and relevance of the results.

References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio.

2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21. ACL.

Ciprian Chelba, Tomás Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639. ISCA.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Tobias Domhan and Felix Hieber. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1500–1505. Association for Computational Linguistics.

Anthony Fader, Luke S. Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1608–1618. The Association for Computer Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104.

Maarten Grootendorst. 2020. Keybert: Minimal keyword extraction with bert.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5149–5156. AAAI Press.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Trans. Assoc. Comput. Linguistics*, 6:437–450.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation, WMT@EMNLP 2011, Edinburgh, Scotland, UK, July 30-31, 2011*, pages 187–197. Association for Computational Linguistics.

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

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1535–1546. Association for Computational Linguistics.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1328–1338. The Association for Computational Linguistics.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Rebecca Knowles and Philipp Koehn. 2016. Neural interactive translation prediction. In *12th Conferences of the Association for Machine Translation in the Americas: MT Researchers' Track, AMTA 2016, Austin, TX, USA, October 28 - November 1, 2016*, pages 107–120. The Association for Machine Translation in the Americas.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119. The Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable neural paraphrase generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3403–3414. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. 2020. Unsupervised paraphrasing by simulated annealing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 302–312. Association for Computational Linguistics.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. CGMH: constrained sentence generation by metropolis-hastings sampling. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6834–6842. AAAI Press.

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 *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 3349–3358. ACL.

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

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1314–1324. Association for Computational Linguistics.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek V. Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2923–2934. ACL.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Aurko Roy and David Grangier. 2019. Unsupervised paraphrasing without translation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6033–6039. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083. Association for Computational Linguistics.

Lei Sha. 2020. Gradient-guided unsupervised lexically constrained text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8692–8703. Association for Computational Linguistics.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4166–4176. Association for Computational Linguistics.

Joern Wuebker, Spence Green, John DeNero, Sasa Hasan, and Minh-Thang Luong. 2016. Models and inference for prefix-constrained machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and Bill Dolan. 2020. POINTER: constrained progressive text generation via insertion-based generative pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8649–8670. Association for Computational Linguistics.