Gradient-guided Unsupervised Lexically Constrained Text Generation

Lei Sha
Apple Inc.
lei_sha@apple.com

Abstract

Lexically-constrained generation requires the target sentence to satisfy some lexical constraints, such as containing some specific words or being the paraphrase to a given sentence, which is very important in many real-world natural language generation applications. Previous works usually apply beam-search-based methods or stochastic searching methods to lexically-constrained generation. However, when the search space is too large, beam-search-based methods always fail to find the constrained optimal solution. At the same time, stochastic search methods always cost too many steps to find the correct optimization direction. In this paper, we propose a novel method G2LC to solve the lexically-constrained generation as an unsupervised gradient-guided optimization problem. We propose a differentiable objective function and use the gradient to help determine which position in the sequence should be changed (deleted or inserted/replaced by another word). The word updating process of the inserted/replaced word also benefits from the guidance of gradient. Besides, our method is free of parallel data training, which is flexible to be used in the inference stage of any pre-trained generation model. We apply G2LC to two generation tasks: keyword-to-sentence generation and unsupervised paraphrase generation. The experiment results show that our method achieves state-of-the-art compared to previous lexically-constrained methods.

1 Introduction

In many natural language generation applications, there are usually some constraints required to be satisfied by the generated sequences. The constraints can be classified into two types:

1. Hard constraints: some specific words or phrases must occur in the target sentence. For example, the facts in abstractive summarization (See et al., 2017a). In detail, when doing summarization, it is always required to keep some key information like the facts. So, the facts are hard constraints for the summarization generation. Another example is the keywords (name or topic) in dialogue generation (Li et al., 2016). These keywords are usually determined by the context of a dialogue, and are required to occur in the next utterance.

2. Soft constraints: such as that the target sentence must have a similar meaning to a given sentence. For example, the synonymous constraint in paraphrase generation (Prakash et al., 2016; Li et al., 2019).

Previous works for lexically-constrained generation can be divided into two branches: enhanced beam search (Hokamp and Liu, 2017; Post and Vilar, 2018) and stochastic search (Miao et al., 2019; Liu et al., 2019). Among the various enhanced beam search methods, grid beam search (Hokamp and Liu, 2017) is the most representative approach, which proposed to add candidate sequences that meet the lexical constraints to the beam in each step to constrain the search space. Dynamic beam allocation methods (Post and Vilar, 2018; Hu et al., 2019) are the extension of grid beam search, which groups the candidates that meet the same amount of constraints into *banks* to accelerate the inference process. However, the reason why beam search based methods work well with machine translation tasks is that the number of potential candidates in each step is relatively small. For general natural language generation tasks with a larger sentence space, beam search based methods will cost too much time to find a solution or even failed.

Stochastic search methods are very promising to solve the above problems. CGMH (Miao et al.,

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 8692–8703,
November 16–20, 2020.

2019) uses Metropolis-Hastings sampling to determine a series of editing actions to a given prototype sentence, including insertion, deletion, and replacement. UPSA (Liu et al., 2019) is a discrete optimization method, which focuses on generating paraphrased sentences in an unsupervised way. UPSA creates a discrete objective function to evaluate semantic similarity and language fluency for a given candidate sentence, and use simulated annealing to search for the optimum solution with three editing actions (insertion, deletion, and replacement). However, in each of the stochastic search steps, many decisions are randomly chosen, such as which position is going to be edited, and which word is going to be replaced with. Then, an accepted rate is applied to decide whether this action should be taken. This stochastic trial-and-error strategy will potentially lead to a waste of search steps.

In this paper, we propose a novel method named G2LC for lexically-constrained generation. G2LC proposes a differentiable objective function that is able to evaluate the semantic similarity, the language fluency, and whether the constraints are satisfied. We use back-propagation to obtain the gradient on the representation of each word, then we propose to take the position with the largest gradient norm as the editing position for insertion, replacement, or deletion. When we choose to insert or replace, we first use a fuzzy word as the inserted or replacement word, which is then to be updated by an optimizer, such as Adagrad (Duchi et al., 2011). The word token will be replaced with a new word if the updated word representation is closed enough to the embedding of that new word.

Our G2LC can be applied to a large variety of tasks. In the experiment, we apply G2LC to two tasks: keyword-to-sentence generation and unsupervised paraphrase generation. The experiment results show that our method achieved state-of-the-art performance compared to previous lexically-constrained generation methods[1].

Our contributions can be summarized as follows:

- We propose a differentiable objective function for multiple keyword/keyphrase constraints inspired by the convolution operation.
- We propose to use gradients to determine the editing position and the new word for replacement or insertion in each search step. This will make the searching process easier to find the optimum result.
- We make our approach possible to assist existing generation models to conduct lexically-constrained generation without any parallel corpus for training. This can be applied to a large variety of generation tasks.

[1]The code is available at https://sites.google.com/view/lcgcode/%E9%A6%96%E9%A1%B5

2 Related Works

Earlier works about lexically-constrained generation are mainly relying on *auxiliary inputs* to constrain the decoder outputs. For example, with the desired prefix, machine translation models are designed to search for the best suffix for the target sentence output (Foster and Lapalme, 2002; Barrachina et al., 2009; Green, 2014; Wuebker et al., 2016; Knowles and Koehn, 2016). To make the keyword able to occur in the middle of the target sentence, Mou et al. (2016) proposed a backward-forward generation method which guarantees to contain only one keyword. For multiple keyword constraints, Cheng et al. (2016) and Domingo et al. (2016) proposed an interactive post-editing method, but these methods tend to bind the lexical constraints to the original model, which will lead to the retraining of the whole model if we would like to use existing generation model to conduct lexically-constrained generation.

Enhanced beam search is proposed as a plug-and-play method to lexically-constrained generation. Grid beam search (Hokamp and Liu, 2017) conduct beam search in two dimensions searching for the candidate sentences that satisfied the given lexical constraints. Anderson et al. (2017) did a similar searching process using finite state automata. Dynamic beam allocation methods (Post and Vilar, 2018; Hasler et al., 2018; Hu et al., 2019) are the extension of grid beam search, which groups the candidates that meet the same amount of constraints into *banks* to accelerate the inference process. Although enhanced beam search works well in the tasks with limited search space such as machine translation, it will cost a lot of time on searching candidate sentences or even failed in general generation tasks when there is a much larger search space.

Stochastic search is also a plug-and-play method that can be applied to large search space. CGMH (Miao et al., 2019) extends Gibbs sampling with word insertion and deletion, then use Metropolis-Hastings sampling to choose new

words for the editing position. For each searching step, an *accept rate* is calculated to decide whether this step should be conducted. The sampling process should be continued until the Markov chain converges to a stationary distribution. UPSA (Liu et al., 2019) also uses a sampling method to decide editing actions and choose new words. Instead of the Markov chain Monte Carlo (MCMC) approach in Miao et al. (2019), UPSA models lexically-constrained generation as a discrete optimization problem and use simulated annealing to solve it. Without the help of gradient, such approaches usually require more search steps to find the correct optimization paths. By contrast, our method is a gradient-guided method, which uses gradient's norm and direction to decide the editing position and the new word for replacement or insertion.

Nearly all generation model can be extended by plug-and-play method, including machine translation (Zhu et al., 2020), table-to-text generation (Sha et al., 2018; Liu et al., 2018), dialogue response generation (Xing et al., 2017; Shi et al., 2019), and Neural Turing Machine (NTM) driven methods (Graves et al., 2014; Sha et al., 2020). There exist two methods: (1) directly replace the original searching method. For example, enhanced beam search (Hokamp and Liu, 2017; Post and Vilar, 2018) replaced the generation model's original beam search method. (2) modify the generated word token sequence. For example, stochastic search methods (Miao et al., 2019; Liu et al., 2019) modify the sequence to maximize a designed score function. Our proposed method can also be taken as a plug-in for existing generation methods, which belongs to the second plug-in method described above. In our method, the original generation model is used as a part of the score function. The largest difference between our method and previous works is that all of our score functions are differentiable, and the gradients can be used to indicate the edit positions and actions.

3 Approach

Although we also use the gradient for optimizing, our approach is inverse to the normal neural network training procedure. In our method, all model weights are separately trained and then fixed, instead, the words' representations are taken as the parameters to be trained. So, our objective must ensure to be differentiable w.r.t the words' representations (which is the input).

3.1 Differentiable Cost Function

In this section, we would like to introduce the design of differentiable cost functions for different objectives.

3.1.1 Lexical Constraint Objective

We assume that there are multiple key phrases $C = \{c_1, \ldots, c_{N_c}\}$ in one generation process, including unigrams and multi-grams. Assume that the sentence tokens are $w_1, \ldots, w_n$ (n is the sentence length), and the word representations are $y_1, \ldots, y_n$, respectively. Different from conventional deep models, in our method, $y_i, i = 1, \ldots, n$ are trainable parameters instead of simply inputs or outputs. So, the exact value of y_i will be updated during the optimizing process. We will use $e(w_i)$ to represent the "fixed" word embedding of word token w_i in this paper.

Given a constraint phrase c with length m, we first transform it into word embedding matrix $M_c \in \mathbb{R}^{md}$ (d is the embedding dimension). Since we require that each key phrase must occur in the target sentence, the most direct method is to compare the key phrase with each subsequence of length m in the sentence. Inspired by convolution operation, we propose to use a sliding window of size m to split the sentence into $n - m$ slices. Then, we calculate the difference between the slices and the constraint phrase's embedding matrix. To make the key phrase occur in the sentence at least once, the smallest difference must reach 0. Therefore, the smallest 2-norm of these differences D_c is defined as the *keyword constraint objective* for constraint c as is shown in Equation 1.

$$D_c = \min_{i=0}^{n-m} \left\| y[i : i + m] - M_c \right\|_2, \qquad (1)$$

where $x[i : i + m]$ represents the i-th slice with length m in the sentence, $\| \cdot \|_2$ represents the 2-norm.

Then, the *keyword constraint objective* for all keyword constraints is the sum of each objective.

$$L_c(y_1, \ldots, y_n) = \sum_{c' \in C} D_{c'}. \qquad (2)$$

3.1.2 Semantic Similarity Objective

For unsupervised paraphrase generation, we propose to separately train a paraphrasing recognition model, which discriminates whether two input sentences have the same meaning. Given two sentences $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$,

we design the paraphrasing recognition model as a deep neural network using the architecture described in Tan et al. (2018). Different from them, the output layer was changed to a binary classification representing paraphrasing or not. Then, the *semantic similarity objective* is defined as:

$$L_{ss} = 1 - P(1|X, Y; \theta_{ss}), \tag{3}$$

where $P(1|X, Y; \theta_{ss})$ represents the probability that X and Y have the same meaning, θ_{ss} is the separately trained parameters which is fixed during the searching process.

3.1.3 Language Fluency Objective

The basic requirement of a generated sentence is to be fluent. We apply a separately trained language model to calculate the likelihood of the given sentence, which is similar to previous works (Miao et al., 2019; Liu et al., 2019).

However, in our method, to make the objective differentiable to the input (words' representation), we apply the language model to sentences in vector level instead of token level. So, the *language fluency objective* is defined as follows:

$$L_{\text{LM}} = -\sum_i \log p(y_i|y_{<i}), \tag{4}$$

where y_i is vector, so cross entropy cannot be directly used in the calculation of $p(y_i|y_{<i})$. We propose to take the word vocabulary as latent variables and obtain the probability as follows:

$$p(y_i|y_{<i}) = \sum_{w \in \mathcal{V}} p(y_i|w)p(w|y_{<i}), \tag{5}$$

where $p(w|y_{<i})$ can be directly obtained by the language model. The first item $p(y_i|w)$ is calculated using Equation 6[2].

$$p(y_i|w) = \frac{\exp\big(-||y_i - e(w)||/T\big)}{\sum_{w' \in \mathcal{V}} \exp\big(-||y_i - e(w')||/T\big)}, \tag{6}$$

where T is a hyperparameter representing the temperature which is set to 0.3 to make sure that $P(w|w)$ extremely close to 1[3].

[2]Since $p(y_i|w) = \frac{p(w|y_i)p(y_i)}{p(w)} \propto p(w|y_i)$, we ignored the constants and directly use the calculation process of $p(w|y_i)$, which is defined by the distance between words in the embedding space in this paper.

[3]Given an embedding of word w, we need to be very sure that it belongs to word w itself. So that $P(w|w)$ should be as close to 1 as possible.

3.1.4 Plug in Existing Generation Model

We can plug our method into any separately trained generation model. Given a language generation model $P(Y|X; \theta_g)$ with the parameters θ_g pretrained and fixed, we can design the plug-in objective similar to the *language fluency objective* under the condition of the encoder part X.

$$L_{\text{plug}} = -\sum_i \log P(y_i|y_{<i}, X). \tag{7}$$

In previous methods, beam search based methods (Hokamp and Liu, 2017; Post and Vilar, 2018) can certainly work together with the existing generation model because they are the decoder part themselves. Our plug-in method can be taken as an extension of beam search based methods, and is more flexible to be applied on other kind of methods. The simulation annealing method (Liu et al., 2019) and the MCMC methods (Miao et al., 2019) are both potentially able to apply our method to be a plug-in via a discrete version of L_{plug}.

3.2 Gradient-guided Editing

The most direct idea of searching for the best sentence is to optimize the representation of all input words according to an optimizer (such as Adagrad (Duchi et al., 2011), Adam (Kingma and Ba, 2014), etc.) until it converges. However, simply update the word representations according to the continuous optimization method will lead to local minima problems. So, we directly edit the word tokens in the sequence to help it escape the local minima. Inspired by previous works (Miao et al., 2019; Liu et al., 2019), we also use three edit actions: *insert, delete*, and *replace*. In each optimizing step, we first choose a position in the sequence for editing. Then, we choose which action to take, *insert, delete, or word replacement*.

The whole search process is composed of many sequence editing actions. We propose to use the gradient of the differentiable score function w.r.t input sentence as the guidance of sequence editing. We focus on three main problems on sequence editing in this section: (1) How to decide the position in the sequence for editing? (2) How to decide the editing action? (3) How to refine the inserted or replaced words?

3.2.1 Edit Position Selecting

With each input of word sequence $y_1, \ldots, y_n$, we calculate the differentiable score function J, and use back propagation to obtain the gradient w.r.t

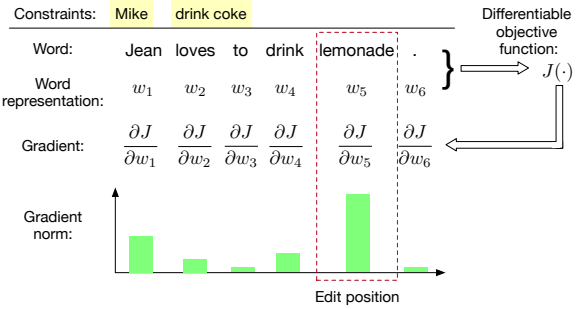

Figure 1: The process of choosing editing position.

each word $\frac{\partial J}{\partial y_i}$. Intuitively, for a variable, the larger the gradient is, the more urgent it is to be updated. Similarly, the 2-norm of each gradient $||\frac{\partial J}{\partial y_i}||_2$ can be taken as a measure of each gradient vector's "length" in the searching space. So, we take the position with the largest gradient norm as the additional editing position. This process is shown in Figure 1.

3.2.2 Edit Action Sampling

After the editing position k is determined, we randomly sample an action with probability $[p_{\text{ins}}, p_{\text{del}}, p_{\text{rep}}]$. Intuitively, a better action with a lower cost should have larger probability. The probabilities are obtained using the following procedure.

First we need to calculate the cost function for each action. For *insertion*, the cost function is $L_{\text{ins}} = L(y_1, \ldots, y_{k-1}, y_*, y_k, \ldots, y_n)$, where y_* represents the inserted word's embedding. Here, we do not know what word should be inserted, so y_* is just a fuzzy word vector which is the average embedding of a subset of the whole vocabulary $\mathcal{W}$ (detailed later). Similarly, for *replacement*, we replace the word y_k to y_*, and then calculate the cost function $L_{\text{rep}} = L(y_1, \ldots, y_{k-1}, y_*, y_{k+1}, \ldots, y_n)$. For *deletion*, we just delete the word at position k and calculate the cost function $L_{\text{del}} = L(y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_n)$. Then, we normalize the three costs[4] as Equation 8.

$$\begin{aligned} L_{\text{ins}}^{\text{norm}} &= -(L_{\text{ins}} - E(L))/\text{Std}(L) \\ L_{\text{del}}^{\text{norm}} &= -(L_{\text{del}} - E(L))/\text{Std}(L) \\ L_{\text{rep}}^{\text{norm}} &= -(L_{\text{rep}} - E(L))/\text{Std}(L), \end{aligned} \tag{8}$$

where $E(L)$ and $\text{Std}(L)$ stands for the mean and standard deviation of the normalized costs. Finally, the probabilities are obtained by `softmax`:

$$[p_{\text{ins}}, p_{\text{del}}, p_{\text{rep}}] = \text{Softmax}([L_{\text{ins}}^{\text{norm}}, L_{\text{del}}^{\text{norm}}, L_{\text{rep}}^{\text{norm}}]). \tag{9}$$

[4]We take a negative operation to the normalized costs because better action tend to have lower cost.

If the *insertion* action is chosen, we have to decide whether to insert y_* in the front or back of the position. We simply make them equal probability and sample for the decision.

Note that when deciding the subset $\mathcal{W}$, we use the pre-trained language model to obtain the top 50 possible words to be filled in the $*$ place. This is similar to the method in CGMH (Miao et al., 2019), and UPSA (Liu et al., 2019). However, our idea in this paper is to let the gradient do what it's supposed to do. When two words occur far away in the word space, gradients will not help much in changing one of them to another, but they can help to refine the word representation to a nearby word. So, we use the language model to coarsely update the word, and then use the gradient to fine-tune the word representation (detailed in Section 3.2.3).

3.2.3 Word Selection and Update

After a new fuzzy word is inserted or replaced, we propose to update the fuzzy word under the guidance of gradients. Under the gradient decent optimization method, the update of each word vector is $y_*^{(t+1)} = y_*^{(t)} - \eta \frac{\partial J}{\partial y_*}$. Note that we can use any optimization method here, including Adam (Kingma and Ba, 2014), Adagrad (Duchi et al., 2011), LBFGS (Liu and Nocedal, 1989), etc.

After the word vectors' update, we need to update the word tokens. Here, we calculate the probability of a word vector to be a word token $P(w|y_*)$, and select the word w_* with the largest probability as the candidate word as is shown in Equation 10.

$$\begin{aligned} w_* &= \arg\max_{w \in \mathcal{V}} P(w|y_*) \\ &= \arg\max_{w \in \mathcal{V}} \frac{\exp(y_*^\top w)}{\sum_{w' \in \mathcal{V}} \exp(y_*^\top w')}, \end{aligned} \tag{10}$$

where $\mathcal{V}$ stands for the whole vocabulary. If $P(w_*|y_*)$ is above a threshold[5] and w_* is not the same as the original word, we will replace the fuzzy word to w_* and also assign w_*'s word embedding to the vector y_*. The gradient updating process can be conducted for several steps until the fuzzy word is replaced to any real word.

In conclusion, the searching process is illustrated in Algorithm 1.

[5]Here, we set the threshold to 0.9 in practice.

Input: Initial word sequence: $W = [w_1, \ldots, w_n]$,
Initial word embeddings: $Y = [y_1, \ldots, y_n]$
Data: Epochs: N
Output: The search result W_*
for $i \leftarrow 1 \ldots N$ **do**
  Calculate $L(Y)$ and $\frac{\partial L(Y)}{\partial Y}$;
  Select the edit position k according to Figure 1;
  Calculate $[p_{\text{ins}}, p_{\text{del}}, p_{\text{rep}}]$ according to Equation 9;
  Sample an action according to multinomial distribution $[p_{\text{ins}}, p_{\text{del}}, p_{\text{rep}}]$;
  Calculate $\mathcal{W}$ according to language model;
  if *action="insert"* **then**
    Sampling: in the front or in the back;
    $W \leftarrow [w_1, \ldots, *, \ldots, w_n]$;
    $y_* \leftarrow \frac{1}{|\mathcal{W}|} \sum_{w \in \mathcal{W}} e(w)$;
    $Y \leftarrow [y_1, \ldots, y_*, \ldots, y_n]$;
  end
  else if *action="delete"* **then**
    $W \leftarrow [w_1, \ldots, w_{k-1}, w_{k+1}, \ldots, w_n]$;
    $Y \leftarrow [y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_n]$;
  end
  else if *action="replace"* **then**
    $W \leftarrow [w_1, \ldots, w_{k-1}, *, w_{k+1}, \ldots, w_n]$;
    $y_* \leftarrow \frac{1}{|\mathcal{W}|} \sum_{w \in \mathcal{W}} e(w)$;
    $Y \leftarrow [y_1, \ldots, y_{k-1}, y_*, y_{k+1}, \ldots, y_n]$;
  end
  while Y *contains fuzzy word* **do**
    Update Y: $Y \leftarrow Y - \eta \frac{\partial L(Y)}{\partial Y}$;
  end
end
return $W_* \leftarrow W$;

Algorithm 1: G2LC searching process

4 Experiments

4.1 Constraint decoding

4.1.1 Dataset & Preprocessing

We use keyword-to-sentence task to evaluate the performance of our method on constraint decoding. Keyword-to-sentence task is extremely important in topic-driven dialogue response generation (Xing et al., 2017), where we need to generate a response with a few hints (usually a keyword as a hard constraint or a topic word as a soft constraint).

The language model is trained using One-Billion-Word Corpus (Chelba et al., 2013)[6]. As is consistent with previous work (Miao et al., 2019), we also sample a 3k-sentence set to provide keyword constraints. For each sentence, we randomly sample 1∼4 keywords as test constraints. The architecture of the language model is a forward 2-layer LSTM RNN. In this task, the target sentence needs to satisfy the lexical constraints while ensuring language fluency. Therefore, the differentiable loss function is as follows:

$$L = \lambda_c L_c + \lambda_{\text{LM}} L_{\text{LM}}, \tag{11}$$

where λ_c and λ_{LM} are hyperparameters, which are set to 1 and 10 in our experiment, respectively.

4.1.2 Competing Methods

We compared our method with the following state-of-the-art approaches:

(1) *Sequence to backward-forward method (seq-B/F)* is proposed by Mou et al. (2016), which takes one keyword as input, and generate the sequence before the keyword as well as the sequence after the keyword using a backward and a forward generator, respectively.

(2) *Asynchronously sequence to backward-forward method (asyn-B/F)* is another method proposed by Mou et al. (2016), which generates the forward "half" sentence under the condition of the backward "half" sentence.

(3) *Grid-beam search (GBS) method* is proposed by Hokamp and Liu (2017), which applies an enhanced beam search to find a valid solution in the constrained search space of the generator.

(4) *Dynamic Beam Allocation (DBA) method* is proposed by Post and Vilar (2018), which is a much faster beam search based method.

(5) *Metropolis-Hastings Sampling (CGMH) method* is proposed by Miao et al. (2019), which uses a series of editing operations to achieve a stationary distribution of Markov chain thus generating the lexically-constrained sentence.

For GBS and DBA method, we just use the trained language model as the decoder and use their method to search for the target sentence. For our method, we simply use a sequence composed of the keywords as the initial sentence, which is the same as CGMH. We run our algorithm for 100 epochs, and finally, select the sentence with the smallest loss as the final result.

4.1.3 Evaluation Metrics

The language fluency of generated target sentences is measured by negative log-likelihood (NLL) loss, which is evaluated by a third party language model (a trigram Kneser-Ney Language Model (Heafield, 2011)). This language model is trained using the English monolingual corpus in WMT18 [7], which is again consistent with previous work (Miao et al., 2019).

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

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

	NLL				Human			
#keyword	1	2	3	4	1	2	3	4
seq-B/F	7.80	-	-	-	0.11	-	-	-
asyn-B/F	8.30	-	-	-	0.09	-	-	-
GBS	7.42	8.72	8.59	9.63	0.32	0.55	0.49	0.55
DBA	7.41	8.58	8.54	9.25	0.43	0.53	0.54	0.59
CGMH	7.04	7.57	8.26	7.92	0.45	0.61	0.56	0.65
G2LC	**7.02**	**7.46**	**8.01**	**7.76**	**0.47**	**0.73**	**0.65**	**0.67**

Table 1: The NLL loss and human evaluation score of the generated sentence with 1∼4 keywords.

For human evaluation, we asked 6 data graders to help us evaluate the language fluency for each target sentence. The labeled score is between 0 (not fluent) and 1 (extremely fluent). The detailed annotation method is shown in Appendix A. The experiment result is shown in Table 1.

According to Table 1, under the condition of 1∼4 keywords, our method G2LC outperformed the previous methods in both the NLL loss and the human evaluation score. Our inter-rater agreement is acceptable due to Krippendorff (2004) with Krippendorff's alpha values 0.73, 0.72, 0.78, 0.80. All results are significant due to the Wilcoxon Signed Rank Test ($p < 0.05$). Compared to CGMH, our method can not only use the gradient norm to locate the editing position more accurately but also our method can use the gradient to fine-tune the inserted or replaced words. Therefore, our gradient-guided method can achieve a better result than CGMH. Note that in the human evaluation, the fluency of single keyword is lower than the fluency of 2, 3, or 4 keywords. The reason is that fewer keywords will lead to larger search space, so the task of searching a sentence based on only one keyword is harder than based on 2, 3, or 4 keywords.

Some of the generated results of our method are listed in Table 2. We can see that for these given keywords, our method can generate a sensitive fluent sentence including these keywords.

Keywords	Generated sentence
computer	the **computer** you want is here
couple	the **couple** made a conversation
claim, street	**claim** that we are on **street** by accident
friends, home	her **friends** are at **home**
death, medical, care	the farmers that diagnosed with alcohol **death** can suspend **medical care**
police, central, bank	the **police** at **central bank** is sleeping
sunday,agents, worked, service	the **sunday** telegraph **agents worked** on the **service**
attempt, copy, painting, denounced	the **attempt** to **copy** the **painting** was **denounced**

Table 2: The example results of keywords-to-sentence generation.

4.2 Unsupervised Paraphrase Generation

4.2.1 Datasets & Implementation details

We use the Quora question pair dataset[8] as our testbed of unsupervised paraphrase generation. This dataset contains 140K paraphrased sentence pairs and 260K non-paraphrased sentence pairs. We follow the experiment settings of Miao et al. (2019) and Liu et al. (2019) to hold out 3K for the validation set and 30K for the test set.

For the training of the paraphrasing recognition model (L_{ss}), we mix the rest paraphrased sentence pairs and the non-paraphrased sentence pairs together, and then take out 1K and 3K for validation and testing, respectively. The paraphrasing recognition model can be designed in any architecture (Sha et al., 2015, 2016; Tan et al., 2018). The accuracy of the trained paraphrasing recognition model achieved 85%. We also trained another version of the paraphrasing recognition model using the SNLI dataset[9] for a cross-domain experiment.

In the paraphrase generation task, the target sentence should be the paraphrase of the input sentence, also we require the target sentence to ensure language fluency. We also conduct experiments that require the target sentence to satisfy the lexical constraints simultaneously. Therefore, the differentiable loss function is as follows:

$$L_{\text{rec}} = \lambda_{\text{LM}} L_{\text{LM}} + \lambda_s L_{\text{ss}}, \tag{12}$$

where λ_{LM}, and λ_{ss} are predefined hyperparameters, which are set to 1, and 2 in our experiment, respectively. L_{ss} is the loss of the paraphrasing recognition model.

For the training of the paraphrasing generation model (L_{plug}), we take the rest paraphrased sentence pairs as the train/valid/test dataset, the valid set has 1K sentence pairs and the test set has 3K sentence pairs. We evaluate the performance of the paraphrasing generation model by BLEU value, which achieves 17.02 on the test set.

We can also generate the target sentence using the paraphrasing generation model, the differentiable loss function is as shown in Equation 13.

$$L_{\text{gen}} = \lambda_{\text{LM}} L_{\text{LM}} + \lambda_p L_{\text{plug}}, \tag{13}$$

[8] https://www.kaggle.com/c/quora-question-pairs

[9] https://nlp.stanford.edu/projects/snli/

	Model	iBLEU	BLEU	ROUGE-1	ROUGE-2	NLL
Supervised	ResidualLSTM	12.67	17.57	59.22	32.40	-
	VAE-SVG-eq	15.17	20.04	59.98	33.30	-
	Pointer generator	16.79	22.65	61.96	36.07	-
	Transformer	16.25	21.73	60.25	33.45	-
	DNPG	**18.01**	**25.03**	**63.73**	**37.75**	-
Supervised (Domain-adapted)	Pointer generator	5.04	6.96	41.89	12.77	-
	Shallow fusion	6.04	7.95	44.87	14.79	-
	MTL	4.90	6.37	37.64	11.83	-
	DNPG	10.39	16.98	56.01	28.61	-
Unsupervised	VAE	8.16	13.96	44.55	22.64	7.74
	CGMH	9.94	15.73	48.73	26.12	7.46
	UPSA	12.02	18.18	56.51	30.69	6.97
	G2LC (Recognizer)	14.34	20.13	58.90	32.79	6.56
	G2LC (Recognizer, Cross)	13.21	19.95	58.02	30.76	6.54
	G2LC (Generator)	14.46	23.27	59.65	33.08	6.12
	G2LC (Generator, Cross)	13.44	21.68	58.89	32.85	6.23

Table 3: The comparison of overall performance between our proposed method and previous methods. We use sentence-level BLEU as is consistent with Liu et al. (2019).

where λ_{LM}, and λ_{p} are again predefined hyperparameters, which are set to 1, and 2 in our experiment, respectively. Here, the paraphrasing generation model is used as a plug-in loss function L_{plug}, which requires the target sentence to have a high probability under the condition of the input sentence.

4.2.2 Competing Methods

We compare the performance of our method with three branches of approaches as follows:

(1) *Supervised methods* are usually Seq2Seq methods, including ResidualLSTM (Prakash et al., 2016), VAE-SVG-eq (Gupta et al., 2018), pointer generator (See et al., 2017b), the Transformer network (Vaswani et al., 2017), and the current state-of-the-art approach DNPG (Li et al., 2019).

(2) *Domain-adapted supervised methods* trained their model using one corpus and then conduct the inference stage on another corpus. These methods include shallow fusion (Gulcehre et al., 2015) and a multi-task learning (MTL) method (Domhan and Hieber, 2017). This kind of method is necessary to be compared with because our model also uses another corpus to train the paraphrase recognition model and paraphrase generation model for the loss function L_{ss} and L_{plug}, respectively.

(3) *Unsupervised methods* do not require any parallel data for training. We have three unsupervised competing methods: VAE (Kingma and Welling, 2013), CGMH (Miao et al., 2019), and UPSA (Liu et al., 2019). VAE is trained using non-paraphrased sentences by minimizing reconstruction loss and KL loss. In the inference stage, sentence vectors are sampled from the latent space and then generated to sentences. CGMH seeks to achieve the stationary state on the Markov chain using Metropolis hasting sampling algorithm. UPSA applies simulated annealing method to maximize a discrete score function, which is the state-of-the-art method among all the unsupervised methods.

We have the following methods for comparison:

- *G2LC (Recognizer)* is optimized by L_{rec}, which use paraphrase recognizer L_{ss} as part of a score function. The paraphrase recognizer L_{ss} is trained by Quora dataset.
- *G2LC (Recognizer, Cross)* is also optimized by L_{rec}, but the paraphrase recognizer L_{ss} is trained by SNLI dataset.
- *G2LC (Generator)* is optimized by L_{gen}, which is a plug-in of a pretrained paraphrase generator, and L_{plug} is part of the score function. The paraphrase generator L_{plug} is trained by Quora dataset.
- *G2LC (Generator, Cross)* is also optimized by L_{gen}, but the paraphrase generator L_{plug} is trained by SNLI dataset.

Method	Relevance	Fluent
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

Table 4: The human evaluation result of our method.

4.2.3 Evaluation Metrics

We use the standard metrics of paraphrase generation (BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004)) for evaluation. However, in the paraphrase generation task, only compare how similar is the generated and reference sentence is not enough, the diversity between the generated and original sentence should also be considered. So, we use iBLEU (Sun and Zhou, 2012) as the major metric, which penalizes the similarity between the generated and original sentence. Li et al. (2019) and Liu et al. (2019) also applied this metric.

We also asked 6 data graders to help us evaluate the language fluency for each target sentence, and evaluate the relevance between each target sentence and their corresponding input sentence. The detailed annotation method is shown in Appendix A.

4.2.4 Results

In Table 3, we listed many variants of our method. *G2LC (Recognizer)* constrains the paraphrasement by the paraphrase recognizing model, which is trained by the Quora dataset. In comparison, the recognizing model L_{ss} in *G2LC (Recognizer, Cross)* is trained by the SNLI dataset. We can see that the generation performance of the cross-domain loss outperforms the previous methods in all evaluation metrics. When L_{ss} is trained on the Quora dataset, the performances are even higher, which is easy to understand because the test data is selected from the same domain. Also, we trained two versions of paraphrasing generation model L_{plug} using Quora and SNLI datasets and reported the results in the lines of *G2LC (Generator)* and *G2LC (Generator, Cross)*. We can see that G2LC with L_{plug} is slightly better than G2LC with L_{ss}, which tells us that the loss calculated by the paraphrasing generation model is a better evaluator for the quality of the target sentence.

Table 4 shows the human evaluation result of our method. We sampled 300 sentences from the generated sentences and asked 6 data graders to judge the relevance score and fluent score of these sentences. Both of the scores range from 0 (the worst) to 1 (the best). All of our inter-rater agreement are acceptable (with Krippendorff's alpha values > 0.70) due to Krippendorff (2004). Due to the resource limit, we do not conduct the human evaluation for the cross-domain methods. According to Table 4, our methods achieved better performance on both of the human evaluation metrics.

Input	Generated sentence
how do i control my anxiety while under situations of extreme pressure	how do i control my anxiety when i am under extreme stress
what are the easiest ways to make good money using the internet	what are the easiest ways that can make money on the internet
why did britain vote to leave the european union	why did britain leave the european union
what can you tell about a person through their handwriting	what can you tell about a person by their handwriting
how do i earn more by investing in share market	how do i make money by investing in share market

Table 5: The example results of paraphrase generation.

Table 5 shows some examples generated by our method *G2LC (Generator)*. With the guide of gradients, the generated sentences are different from the input sentence in some words but are still paraphrase to the input sentence.

5 Conclusion & Future Works

In this paper, we propose a gradient guided method to conduct unsupervised lexical constraint generation. The lexical constraints include hard constraints (keywords) and soft constraints (paraphrasing). We first defined a series of differentiable loss functions which represents the fluency of the generated sentence as well as whether the constraints are satisfied. Then, we use the value of the gradient norm to decide which word in the sentence has the most urgent need to be edited, including being inserted in the front or the back, being deleted, and being changed. We applied our method in two tasks, keyword-to-sentence generation, and unsupervised paraphrasing. Our method achieved state-of-the-art performance on both of these tasks. Using post-editing methods for lexical constraint generation can be taken as an initial step of controlling the generation result. Future research works can be conducted to make the generation process more robust and interpretable.

Acknowledgement

We would like to thank the three anonymous reviewers and the anonymous meta-reviewer for so many good suggestions.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945, Copenhagen, Denmark. Association for Computational Linguistics.

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, et al. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Shanbo Cheng, Shujian Huang, Huadong Chen, Xin-Yu Dai, and Jiajun Chen. 2016. PRIMT: A pick-revise framework for interactive machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1240–1249, San Diego, California. 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*, pages 1500–1505, Copenhagen, Denmark. Association for Computational Linguistics.

Miguel Domingo, Alvaro Peris, and Francisco Casacuberta. 2016. Interactive-predictive translation based on multiple word-segments. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 282–291.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

George Foster and Guy Lapalme. 2002. *Text prediction for translators*. Ph.D. thesis, Université de Montréal.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Spence Green. 2014. *Mixed-initiative natural language translation*. Ph.D. thesis, Stanford University.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 506–512, New Orleans, Louisiana. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved lexically constrained decoding for translation and monolingual rewriting. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 839–850, Minneapolis, Minnesota. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Rebecca Knowles and Philipp Koehn. 2016. Neural interactive translation prediction. In *Proceedings of the Association for Machine Translation in the Americas*, pages 107–120.

Klaus Krippendorff. 2004. Content analysis: An introduction to its methodology thousand oaks. *Calif.: Sage*.

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

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable neural paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3403–3414, Florence, Italy. Association for Computational Linguistics.

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

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. 2019. Unsupervised paraphrasing by simulated annealing. *arXiv preprint arXiv:1909.03588*.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 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 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358, Osaka, Japan. The COLING 2016 Organizing Committee.

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*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

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

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

Abigail See, Peter J Liu, and Christopher D Manning. 2017b. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read LSTM unit for textual entailment recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2870–2879, Osaka, Japan. The COLING 2016 Organizing Committee.

Lei Sha, Sujian Li, Baobao Chang, Zhifang Sui, and Tingsong Jiang. 2015. Recognizing textual entailment using probabilistic inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1620–1625, Lisbon, Portugal. Association for Computational Linguistics.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Lei Sha, Chen Shi, Qi Chen, Lintao Zhang, and Houfeng Wang. 2020. Estimating minimum operation steps via memory-based recurrent calculation network. In *2020 international joint conference on neural networks (IJCNN)*. IEEE.

Chen Shi, Qi Chen, Lei Sha, Hui Xue, Sujian Li, Lintao Zhang, and Houfeng Wang. 2019. We know what you will ask: A dialogue system for multi-intent switch and prediction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 93–104. Springer.

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

Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway attention networks for modeling sentence pairs. In *International Joint Conferences on Artificial Intelligence*, pages 4411–4417.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Joern Wuebker, Spence Green, John DeNero, Saša 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 (Volume 1: Long Papers)*, pages 66–75, Berlin, Germany. Association for Computational Linguistics.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3351–3357.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating bert into neural machine translation. *arXiv preprint arXiv:2002.06823*.

Appendices

A Human Evaluation Question Marks

In both tasks, we ask the data graders to grade each sentence with a fluent score. In unsupervised paragraph generation task, we ask the data graders to annotate each pair of sentence with a relevance score. The data graders are not necessary to give one of these 5 scores, they can also give some mid scores if need be.

A.1 Fluency

Q: How fluent do you think the sentence is?

Please choose a score according to the following description. Note that the score is not necessary the same as listed, you can give scores like 0.32 or 0.49 , if you deem appropriate.

- 1.00: Extremely fluent.
- 0.75: Can be understood with several grammatical errors.
- 0.50: Can be understood by some extent, but with many grammatical errors .
- 0.25: Can not be understood, but some segments are fluent.
- 0.00: Not readable.

A.2 Relevance

Q: How relevant do you think the given two sentences is?

Please choose a score according to the following description. Note that the score is not necessary the same as listed, you can give scores like 0.32 or 0.49 , if you deem appropriate.

- 1.00: They are exactly the same meaning.
- 0.75: They have similar meaning, but some details are not identical.
- 0.50: Although the two sentence have some semantic meaning in common, they have too much different details.
- 0.25: Most of the meaning are not the same, but some details are identical.
- 0.00: They are totally different.