CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling

Ning Miao,[1] Hao Zhou,[2] Lili Mou,[3] Rui Yan,[1] Lei Li[2]
[1]Institute of Computer Science and Technology, Peking University, China
[2]ByteDance AI Lab, Beijing, China
[3]AdeptMind Research, Toronto, Canada
miaoning@pku.edu.cn, zhouhao.nlp@bytedance.com, doublepower.mou@gmail.com,
rui.yan.peking@gmail.com, lileilab@bytedance.com

Abstract

In real-world applications of natural language generation, there are often constraints on the target sentences in addition to fluency and naturalness requirements. Existing language generation techniques are usually based on recurrent neural networks (RNNs). However, it is non-trivial to impose constraints on RNN based methods while maintaining generation quality, since RNNs generate sentences sequentially and approximately with beam search. In this paper, we propose CGMH, a novel approach using Monte-Carlo sampling to generate sentences under constraints. We evaluate our method on a variety of tasks, including keyword-to-sentence generation, sentence paraphrasing, and sentence error correction. The underlying model is trained in an unsupervised fashion, therefore CGMH does not require parallel corpus. CGMH also enables complex constraints such as the occurrence of multiple keywords in the target sentences, which are not handled in traditional RNN based approaches. It achieves high performance compared with previous supervised generation methods. The code is released at https://github.com/NingMiao/CGMH.

Introduction

Various natural language generation tasks involve constraints on the target sentences. The constraints can be categorized into the following: (1) Hard constraints, such as the mandatory inclusion of certain keywords in the output sentences; and (2) Soft constraints, such as requiring the generated sentences to be semantically related to a certain topic. Figure 1 illustrates an example of advertisement generation, where "BMW" and "sports" should appear in the advertising slogan. Hence, "BMW, the sports car of the future" is a valid generated sentence.

Existing sentence generation methods are mostly based on recurrent neural networks (RNNs), which generate a sentence sequentially from left to right (Sutskever, Vinyals, and Le 2014). However, it is non-trivial to impose constraints during the left-to-right generation in RNNs. Previous work proposes a backward-forward generation approach (Mou et al. 2015), which could only generate sentences with one keyword. Additionally, researchers propose grid beam search to incorporate constraints in machine translation (Post and Vilar 2018; Hasler et al. 2018; Hokamp and Liu 2017). It works with the translation task because the source and target are mostly aligned and the candidate set of translations is small. But grid-search would not work well with general sentence generation, which has many more candidate sentences.

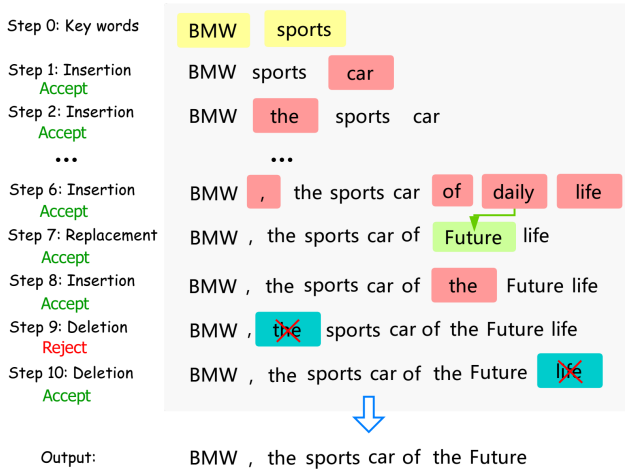

Figure 1: CGMH generates one sample of sentences with the keyword inclusion constraint. In each step, CGMH proposes a candidate modification which is accepted or rejected according to certain acceptance rate.

In this paper, we propose *Constrained Generation by Metropolis-Hastings sampling* (CGMH), a novel approach that addresses sentence generation under constraints. CGMH naturally generates more diverse samples compared with RNN-based generation algorithms. Different from previous sentence samplers working in the variational hidden space (Bowman et al. 2016), CGMH directly samples from the sentence space using the Metropolis-Hastings (MH) algorithm (Metropolis et al. 1953). MH algorithm is an instance of the general Markov chain Monte Carlo sampling algorithms (MCMC). In order to generate sentences, we define local operations in the sentence space (e.g., word replacement, deletion, insertion). During sampling, we randomly choose a word and an operation to form a *proposal* for transition. The proposed modification is either accepted or rejected according to the *acceptance rate* computed for a pre-specified *stationary distribution*. Compared with Gibbs sampling (another MCMC method), MH is more flexible to

generate sentences with arbitrary lengths.

Moreover, it is straightforward to impose constraints on MH sampling by introducing a matching function (which indicates the degree to which the constraints are satisfied) to manipulate the stationary distribution. For hard constraints, the matching function could be a binary indicator, ruling out the possibility of an infeasible sentence. For soft constraints, the matching function could be, for example, a measure of semantic similarity. In this way, we are able to sample sentences with certain constraints.

Our proposed CGMH can be applied to a variety of tasks. We first experiment keyword-to-sentence generation, where keywords are hard constraints. In this task, CGMH outperforms state-of-the-art constrained generation models in both negative likelihood (fluency) and human evaluation. We also conduct experiments on two generation tasks with soft constraints, namely, paraphrase generation and sentence error correction. Results show that, without a parallel corpus, CGMH significantly outperforms other unsupervised models, achieving close results to state-of-the-art supervised approaches.

In summary, our contributions include

- We propose CGMH, a general framework for sentence sampling, that can generate sentences with both hard and soft constraints.
- We design the proposal distribution and stationary distributions for MH sentence sampling in three tasks, including keyword-to-sentence generation, paraphrase generation, and sentence error correction. Experimental results show that our method achieves high performance compared with previous methods.
- We make it possible to accomplish the above tasks in an unsupervised fashion, which does not require a parallel corpora as is needed in previous approaches.

Related Work

In recent years, sentence generation is mostly based on the recurrent neural network (RNN) because of its capability of learning highly complicated structures of language (Sutskever, Vinyals, and Le 2014). In most tasks, sentence generation from RNN is modeled as max *a posteriori* (MAP) inference, and people use greedy search or beam search to approximate the most probable sentences (Sutskever, Vinyals, and Le 2014).

For sentence sampling, the most naïve approach, perhaps, is to sample words from RNN's predicted probabilities step by step, known as forward sampling in the Bayesian network regime. The prototype-then-edit model (Guu et al. 2018) first samples prototypes, and then edits the prototypes to obtain new sentences. Bowman et al. use variational autoencoders (VAEs) to sample sentences from a continuous hidden space. However, these approaches allow neither soft constraints that require flexible manipulation of sentence probabilities, nor hard constraints that specify one or more given words.

Berglund et al. propose a Gibbs sampling model that uses a bi-directional RNN to alternatively replace a token from its posterior distribution. Su et al. further apply it to control the sentiment of a sentence. However, the shortcoming of Gibbs sampling is obvious: it cannot change the length of sentences and hence is not able to solve complicated problems such as generation from keywords. Our paper extends Gibbs sampling with word insertion and deletion. This results in a Metropolis-Hastings sampler, enabling more flexible sampling. Harrison, Purdy, and Riedl utilize MH to sample quadruples for generating stories, which cannot be directly used in constrained sentence generation.

Another line of work tackles the problem of constrained sentence generation from a searching perspective. In neural machine translation, for example, grid beam search (Hokamp and Liu 2017, GBS) makes use of 2-dimensional beam search to find sentences that satisfy constraints, whereas constrained beam search (Anderson et al. 2017, CBS) utilizes a finite-state machine to assist searching. Post and Vilar and Hasler et al. further accelerate the search process. In machine translation, the search space is limited and highly conditions on the source sentence. But in many other generation tasks, there maybe much more candidate sentences and the GBS may fail for that the greedy pruning in the grid always makes constraints unable to find satisfied sentences in their large search space.

Sentence generation with soft constrains is also related to controlling latent features of a sentence (Prabhumoye et al. 2018; Li et al. 2018), such as meaning and sentiment. Hu et al. apply a discriminator to VAE to generate sentences with specified sentiments; and Shen et al. achieve style transfer by cross-alignment with only non-parallel data. Such approaches require an explicit definition of the latent feature (e.g., sentiment), supported by large labeled datasets.

The difference between our model and previous work is that our MH sampling framework is applicable to both hard and soft constraints. It immediately enables several non-trivial applications, including unsupervised paraphrase generation, unsupervised error correction, and keyword-based sentence generation.

Approach

In this section, we describe our CGMH model (referring to *Constrained Generation by Metropolis-Hastings*) in detail. We first introduce the MH sampling framework in general, and then we design MH components—including proposal design, stationary distributions, and acceptance decision—in the scenario of constrained sentence generation.

The Metropolis-Hastings Algorithm

The Metropolis-Hastings (MH) algorithm is a classical Markov chain Monte Carlo (MCMC) sampling approach. MCMC generates a sequence of correlated samples by iteratively jumping from one state to a new state, according to the transition matrix of a Markov chain. For sentence generation, each state of the Markov chain is a particular sentence. Under mild conditions, the distribution of samples will converge to the *stationary distribution* of the Markov chain. Therefore, we need to design a Markov chain whose stationary distribution is the desired sentence distribution.

However, it is sometimes difficult to directly specify the

transition probability of the Markov chain to obtain an arbitrary stationary distribution. The MH sampler solves this problem in a two-step fashion. It first proposes a tentative transition, but then accepts or rejects the *proposal* according an *acceptance rate*. The acceptance/rejection is computed by the desired stationary distribution and the proposal distribution. This ensures the detailed balance condition, which in turn guarantees that MH converges to the desired distribution.

More specifically, let $\pi(\mathrm{x})$ be the distribution from which we want to sample sentences (x denotes a particular sentence). MH starts from a (possibly) arbitrary state x_0 (an initial sentence or a sequence of keywords). At each step t, a new sentence x' is proposed based on a proposal distribution $g(\mathrm{x}'|\mathrm{x}_{t-1})$, where x_{t-1} denotes the sentence of the last step. Then the proposal could be either accepted or rejected, given by the acceptance rate

$$A(\mathrm{x}'|\mathrm{x}_{t-1}) = \min\{1, A^*(\mathrm{x}'|\mathrm{x}_{t-1})\} \quad (1)$$

$$\text{where} \qquad A^*(\mathrm{x}'|\mathrm{x}_{t-1}) = \frac{\pi(\mathrm{x}')g(\mathrm{x}_{t-1}|\mathrm{x}')}{\pi(\mathrm{x}_{t-1})g(\mathrm{x}'|\mathrm{x}_{t-1})} \quad (2)$$

In other words, the proposal is accepted with a probability of $A(\mathrm{x}'|\mathrm{x}_{t-1})$, and the next sentence $\mathrm{x}_t = \mathrm{x}'$. With a probability of $1 - A(\mathrm{x}'|\mathrm{x}_{t-1})$, the proposal is otherwise rejected and $\mathrm{x}_t = \mathrm{x}_{t-1}$. Theoretically, the distribution of sample x_n will converge to $\pi(\mathrm{x})$ as $n \to \infty$ for irreducible and aperiodic Markov chains. In practice, initial several samples are discarded as they are subject to the initial states x_0. If the samples converge to the stationary distribution, we say the Markov chain *mixes* or *burns in*. Readers can refer to Gelman et al. for details of MH sampling.

The MH framework is a flexible sampling algorithm because: (1) The proposal distribution could be arbitrary, as long as the Markov chain is irreducible and aperiodic; (2) The stationary distribution could also be arbitrarily specified, which will be reflected in Equation 2 to correct the proposal distribution; and (3) We can safely ignore a normalization factor of the stationary distribution and only specify an unnormalized measure, because $\pi(\cdot)$ appears in both numerator and denominator of Equation 2. All these allow flexible managing of the stationary distribution.

The design of proposal distributions and stationary distributions relies heavily on applications, which will be described in the rest of this section.

Proposals

We design a set of simple yet effective proposals, including *replacement*, *insertion*, and *deletion* of words. Concretely, starting from an initial sentence, we randomly select a word at each step, and for selected word, we perform one of the three operations randomly, with probability $[p_{\text{insert}}, p_{\text{delete}}, p_{\text{replace}}]$, which is set to $[1/3, 1/3, 1/3]$ in our experiments. We further describe the operations as follows.

Replacement. Assume that the sentence at the current step is $\mathrm{x} = [w_1, \cdots, w_{m-1}, w_m, w_{m+1}, \cdots, w_n]$, where n is sentence length, and that we decide to propose a replacement action for the m-th word w_m.

Given all other words in the current sentence, we need to choose a new word for the m-th position by the conditional probability:

$$g_{\text{replace}}(\mathrm{x}'|\mathrm{x}) = \pi(w_m^* = w^c|\mathrm{x}_{-m}) = \frac{\pi(w_1, \cdots, w_{m-1}, w^c, w_{m+1}, \cdots, w_n)}{\sum_{w \in \mathcal{V}} \pi(w_1, \cdots, w_{m-1}, w, w_{m+1}, \cdots, w_n)} \quad (3)$$

where w_m^* is the new word for position m, w^c is a candidate word for w_m^*, $\mathrm{x}' = [w_1, \cdots, w_{m-1}, w^c, w_{m+1}, \cdots, w_n]$ is the candidate sentence, $\mathcal{V}$ is the set of all words, and $g_{\text{replace}}(\mathrm{x}'|\mathrm{x})$ is the probability of choosing x' as the target of replacement action from x. However, it is difficult to compute $\pi(w_m^* = w^c|\mathrm{x}_{-m})$ for all $w^c \in \mathcal{V}$, because we have to compute $\pi(w_1, \cdots, w_{m-1}, w^c, w_{m+1}, \cdots, w_n)$ for each candidate sentence separately. This results from different words in the middle of a sentence and thereafter different RNN hidden states.

We propose to pre-select a subset of plausible candidate words. It is easy to compute $\pi(w_1, ..., w_{m-1}, w_m^* = w^c)$ as well as $\pi(w_m^* = w^c, w_{m+1}, ..., w_n)$ by a forward and a backward language model, and $\pi(w_1, ..., w_m^* = w^c, x_n)$ is no greater than either of them. We thus build a pre-selector Q to discard w^c with low forward or backward probability:

$$Q(w^c) = \min(\ \pi(w_1, ..., w_{m-1}, w_m^* = w^c), \pi(w_m^* = w^c, w_{m+1}, ..., w_n)) \quad (4)$$

After pre-selection, we compute the conditional probability of selected words by Equation (3), from which we sample a word for replacement.

Insertion and deletion. Inserting a word is done in a two-step fashion: we first insert a special token, placeholder <PHD>, at the position that we are currently working on, and then use (3) to sample a real word to replace the placeholder. As a result, g_{insert} takes a similar form to (3).

Deleting is, perhaps, the simplest operation, and we directly remove the current word. Suppose $\mathrm{x} = [w_1, \cdots, w_{m-1}, w_m, w_{m+1} \cdots w_n]$ and we are about to delete the word w_m. Then $g_{\text{delete}}(\mathrm{x}'|\mathrm{x}_{t-1})$ equals 1 if $\mathrm{x}' = [w_1, \cdots, w_{m-1}, w_{m+1} \cdots w_n]$, or 0 for other sentences.

Insertion and deletion ensure the **ergodicity** of the Markov chain, as in the worst case, any two sentences, x and x', are still reachable by first deleting all words in x, and then inserting all words in x' in order. In addition, word replacement is an intuitive operation that helps reach "semantically neighboring" states more easily. Therefore we include it as one of our proposals. It should be noticed that, although the replacement action is restricted to top-ranked candidate words, this does not affect the ergodicity of the Markov chain.

Stationary Distribution

In the proposed CGMH framework, we would like to obtain sentences from a desired distribution $\pi(\mathrm{x})$, known as the stationary distribution of the Markov chain. For constrained sentence generation, CGMH allows flexible design of the stationary distribution.

Generally, the stationary distribution $\pi(\mathrm{x})$ can be defined as

$$\pi(\mathrm{x}) \propto p(\mathrm{x}) \cdot \underbrace{\mathcal{X}_{\mathrm{c}}^0(\mathrm{x}) \cdots \mathcal{X}_{\mathrm{c}}^n(\mathrm{x})}_{\text{constraints}} \tag{5}$$

where $p(\mathrm{x})$ is the probability of a sentence in general, and $\mathcal{X}_{\mathrm{c}}^0(\mathrm{x}), \cdots, \mathcal{X}_{\mathrm{c}}^n(\mathrm{x})$ are scoring functions indicating how much a constraint is satisfied. By multiplying these scoring functions together, we could impose more than one constraints.

Technically, CGMH works with both hard and soft constraints. For a hard constraint, $\mathcal{X}_{\mathrm{c}}^i$ is an indicator function, which equals 1 if the i-th constraint is satisfied, or 0 otherwise. For a soft constraint, $\mathcal{X}_{\mathrm{c}}^i$ is a "smoothed" indicator function showing the degree to which the sentence satisfies the (soft) constraint.

The design of the stationary distribution is flexible but task related. In our paper, we apply the CGMH framework to three different tasks.

Sentence generation with keywords. In this task, we would like to generate a sentence given one or more keywords as constraints. It has been previously explored in various applications including question answering (Yin et al. 2016) and dialog systems (Mou et al. 2016). Most previous work makes use of attention or copying mechanisms to impose the keyword constraints in a soft manner, which means that the constraint may not be satisfied (Yin et al. 2016; Gu et al. 2016).

In our CGMH framework, it is natural to impose hard constraints by restricting the support of the stationary distribution to feasible solutions. In particular, we have

$$\pi(\mathrm{x}) \propto p_{\mathrm{LM}}(\mathrm{x}) \cdot \mathcal{X}_{\mathrm{keyword}}(\mathrm{x})$$

where p_{LM} is a general sentence probability computed by a language model and $\mathcal{X}_{\mathrm{keyword}}$ is the indicator function showing if the keywords are included in the generated sentence. In other words, the stationary distribution is proportional to the language model probability if all constraints are satisfied (keywords appearing in the sentence), or 0 otherwise. During generation, the initial sentence x_0 is simply a sequence of keywords, and then we perform sampling to generate valid sentences.

Unsupervised paraphrase generation. Paraphrase generation aims to synthesize literally different sentences that convey the same meaning as the input sentence. It is an important task in NLP, and can be a key component in downstream applications such as data augmentation for NLP. Previous state-of-the-art paraphrase generation methods require parallel data for training, which is not always available.

In our paper, we view paraphrase generation as sampling from a distribution, where the sentences are (1) fluent by themselves and (2) close in meaning to the input sentence x_*. The former property can be captured by a traditional language model, whereas the latter can be modeled as a constraint. Concretely, we have

$$\pi(\mathrm{x}) \propto p_{\mathrm{LM}}(\mathrm{x}) \cdot \mathcal{X}_{\mathrm{match}}(\mathrm{x}|\mathrm{x}_*) \tag{6}$$

Here, $p_{\mathrm{LM}}(\mathrm{x})$ is also the probability given by a language model, indicating the fluency of x. $\mathcal{X}_{\mathrm{match}}(\mathrm{x}|\mathrm{x}_*)$ is a matching score. In our experiments, we have several choices for $\mathcal{X}_{\mathrm{match}}(\cdot|\cdot)$:

- *Keyword matching* (KW) as hard constraints. We observe that paraphrases typically share some keywords with the original sentence. In this variant, we use Rake (Rose et al. 2010) to extract keywords and keep them fixed during sampling. That is $\mathcal{X}_{\mathrm{match}}(\mathrm{x}|\mathrm{x}_*) = 1$ if x and x_* share the same keywords, and 0 otherwise.
- *Word embedding similarity* as a soft constraint. Embeddings map discrete words to real-valued vectors, providing a softer way of measuring similarity. In this matching function, we enhance keyword matching with embedding similarity. For any word w in a sentence x, we first find the closest word in the input sentence x_* by computing their cosine similarity (Pennington, Socher, and Manning 2014). Then either the minimum or the average of these cosine measures is taken as the matching score, resulting in two variants (WVM, WVA).
- *Skip-thoughts similarity* (ST) as a soft constraint. Skip-thoughts trains sentence embeddings by predicting words of surrounding sentences (Kiros et al. 2015). We compute the cosine similarity between the skip-thought embeddings of x and x_*, and use it as the matching score.

Theoretically speaking, we may start sampling from any sentence, once the stationary distribution is defined. However, it would take too long for the Markov chain to mix/burn-in (i.e., samples are from the desired distribution). We thus use the original sentence as the initial state, i.e., $\mathrm{x}_0 = \mathrm{x}_*$. This is similar to the warm start in Gibbs sampling for contrastive divergence estimation (Hinton, Osindero, and Teh 2006).

Unsupervised sentence error correction. Previous work of sentence error correction also depends on parallel data (Felice et al. 2014; Junczys-Dowmunt and Grundkiewicz 2016; Sakaguchi, Post, and Van Durme 2017; Chollampatt, Hoang, and Ng 2016). Our CGMH framework allows us to generate samples from a distribution of correct sentences, starting from an erroneous one as the input x_*. In this application, we use the same stationary distribution (Equation 6) as in the unsupervised paraphrase setting, where p_{LM} is trained on a general corpus ensuring the fluency (correctness), and $\mathcal{X}_{\mathrm{match}}(\cdot|\cdot)$—assumed insensitive to grammatical errors—imposes a soft constraint of semantic relevance.

Acceptance Rate

In MH, both proposals and stationary distributions can be specified. The way to ensure that the samples are indeed from the desired distribution is to correct the proposal distribution by a probability of acceptance or rejection, given by an acceptance rate in Equations (1) and (2).

In our approach, we have three types of proposals, namely, deletion, insertion, and replacement. We thus break-

down our acceptance rate (before taking $\min\{1, \cdot\}$) as

$$A^*_{\text{replace}}(x'|x) = \frac{p_{\text{replace}} \cdot g_{\text{replace}}(\mathrm{x}|\mathrm{x}') \cdot \pi(x')}{p_{\text{replace}} \cdot g_{\text{replace}}(\mathrm{x}'|\mathrm{x}) \cdot \pi(x)}$$
$$\approx \frac{\pi(w_m|x_{-m}) \cdot \pi(\mathrm{x}')}{\pi(w'_m|x_{-m}) \cdot \pi(\mathrm{x})} = 1 \quad (7)$$

$$A^*_{\text{insert}}(\mathrm{x}'|\mathrm{x}) = \frac{p_{\text{delete}} \cdot g_{\text{delete}}(\mathrm{x}|\mathrm{x}') \cdot \pi(\mathrm{x}')}{p_{\text{insert}} \cdot g_{\text{insert}}(\mathrm{x}'|\mathrm{x}) \cdot \pi(\mathrm{x})}$$
$$= \frac{p_{\text{delete}} \cdot \pi(\mathrm{x}')}{p_{\text{insert}} \cdot g_{\text{insert}}(\mathrm{x}'|\mathrm{x}) \cdot \pi(\mathrm{x})} \quad (8)$$

$$A^*_{\text{delete}}(\mathrm{x}'|\mathrm{x}) = \frac{p_{\text{insert}} \cdot g_{\text{insert}}(\mathrm{x}|\mathrm{x}') \cdot \pi(\mathrm{x}')}{p_{\text{delete}} \cdot g_{\text{delete}}(\mathrm{x}'|\mathrm{x}) \cdot \pi(\mathrm{x})}$$
$$= \frac{p_{\text{insert}} \cdot g_{\text{insert}}(\mathrm{x}|\mathrm{x}') \cdot \pi(\mathrm{x}')}{p_{\text{delete}} \cdot \pi(\mathrm{x})} \quad (9)$$

In particular, (7) is trivially true because word replacement could be thought of as a step of Gibbs sampling, which is in turn a step of MH sampling whose acceptance rate is guaranteed to be 1. (8) and (9) are reciprocal because deletion and insertion are the inverse operation to each other.

Experiments

We evaluated our approach on a variety of tasks including sentence generation from keywords, unsupervised paraphrase generation, and unsupervised sentence error correction. We also conducted an in-depth analysis of the proposed CGMH method.

Keywords-to-Sentence Generation

For keywords-to-sentence generation, we used the One-Billion-Word Corpus (Chelba et al. 2013)[1] and randomly chose 5M sentences (about 130M words) to train our language models. We held out a 3k-sentence set to provide keywords for testing. For each sentence, we randomly sampled one or more words as the constraint(s). Our language models are simply a two-layer LSTM with hidden size 300. We chose 50k most frequent words as the dictionary.

For MH sampling, we used the sequence of keywords as the initial state, and chose the sentence with the lowest perplexity after 100 steps as the output. We set maximal sampling steps as 200.

We tested the negative likelihood (NLL) of sentences to evaluate their fluency. NLL is given by a third-party n-gram language model trained on the English monolingual corpus of WMT18.[2] We also invited 3 volunteers to score the fluency of generated sentences. Volunteers were asked to score 100 samples from each method according to their quality. Scores range from 0 to 1, where 1 indicates the best quality.

Table 1 compares our method with current state-of-the-art approaches of constrained generation, namely, the grid beam search approach (GBS) (Hokamp and Liu 2017) and two variants of the backward forward model (sep-B/F and asyn-B/F) (Mou et al. 2015).

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

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

#keyword(s)		CGMH	GBS	sep-B/F	asyn-B/F
1	NLL	**7.04**	7.42	7.80	8.30
	Human	**0.45**	0.32	0.11	0.09
2	NLL	**7.57**	8.72	-	-
	Human	**0.61**	0.55	-	-
3	NLL	**8.26**	8.59	-	-
	Human	**0.56**	0.49	-	-
4	NLL	**7.92**	9.63	-	-
	Human	**0.65**	0.55	-	-

Table 1: Results of NLL and human evaluation on sentences with 1 to 4 keywords. Sentences with lower NLL and higher human evaluation scores are better.

Keyword(s)	Generated Sentences
friends	My good **friends** were in danger .
project	The first **project** of the scheme .
have, trip	But many people **have** never made the **trip** .
lottery, scholarships	But the **lottery** has provided **scholarships** .
decision, build, home	The **decision** is to **build** a new **home** .
attempt, copy, painting, denounced	The first **attempt** to **copy** the **painting** was **denounced** .

Table 2: Sentences generated from keywords by CGMH.

Statistic	Value
Mean intra-annotator std	0.098
Mean intra-model std	0.280
p-value (1 keyword)	< 0.01
p-value (2 keywords)	< 0.05
p-value (3 keywords)	< 0.05
p-value (4 keywords)	< 0.01

Table 3: Statistics of human evaluation for keywords-to-sentence generation.

CGMH outperforms previous work in both NLL and human evaluations. The two variants of B/F cannot generate more than one keywords. GBS is designed for machine translation, so it works well on sentence generation with limited semantic space. But pruning in grid may make the keywords unable to find appropriate prefixes, especially when generating without limited semantics. Table 2 provides several examples of keywords-to-sentence generation by CGMH.

We present statistics of human evaluation in Table 3. "Mean intra-annotator std" is the mean standard deviation of scores from different volunteers, whereas "Mean intra-model std" is the mean standard deviation of each model. This implies that the volunteers achieve high consistency with each other, and that the gap between different models is large. p-values in this table are between CGMH and GBS; for CGMH and B/F models, p-values are lower than 0.001. This shows that the results given by human annotation are statistically significant.

Unsupervised Paraphrase Generation

We followed previous work of supervised paraphrase generation (Gupta et al. 2017; Prakash et al. 2016; Gupta et al. 2017; Li et al. 2017) and used a standard benchmark, the Quora dataset,[3] to evaluate each model. The dataset con-

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

Model	BLEU-ref	BLEU-ori	NLL
Origin Sentence	30.49	100.00	7.73
VAE-SVG (100k)	22.50	-	-
VAE-SVG-eq (100k)	**22.90**	-	-
VAE-SVG (50k)	17.10	-	-
VAE-SVG-eq (50k)	17.40	-	-
Seq2seq (100k)	22.79	33.83	6.37
Seq2seq (50k)	20.18	27.59	**6.71**
Seq2seq (20k)	16.77	22.44	6.67
VAE (unsupervised)	9.25	27.23	7.74
CGMH *w/o matching*	18.85	50.28	7.52
w/ KW	20.17	53.15	7.57
w/ KW + WVA	20.41	53.64	7.57
w/ KW + WVM	20.89	54.96	7.46
w/ KW + ST	20.70	54.50	7.78

Table 4: Performances of different paraphrase models. Ideal paraphrase generator should achieve higher BLEU-ref, lower BLEU-ori, and lower NLL scores.

Type	Examples
Ori	what 's the best plan to lose weight
Ref	what is a good diet to lose weight
Gen	what 's the best way to slim down quickly
Ori	how should i control my emotion
Ref	how do i control anger and impulsive emotions
Gen	how do i control my anger
Ori	why do my dogs love to eat tuna fish
Ref	why do my dogs love eating tuna fish
Gen	why do some dogs like to eat raw tuna and raw fish

Table 5: Paraphrase generation given by CGMH w/ KW+WVM. For each sample, we show the original sentence (Ori), the reference paraphrase (Ref), and the generated sentence (Gen).

Step	State (Sentence)	Proposal
Origin	what movie do you like most .	**replace** *what* with *which*
1	which movie do you like most .	**delete** *most*
2	which movie do you like .	**insert** *best*
3	which movie do you like best .	**replace** *like* with *think*
4	which movie do you think best .	**insert** *the*
5	which movie do you think the best .	**insert** *is*
Output	which movie do you think is the best .	-

Table 6: An example of the sampling process given by CGMH w/ KW+WVM.

tains 140k pairs of paraphrase sentences, and 260k pairs of non-paraphrase sentences. We followed the standard split of paraphrase sentences, which holds out 3k and 30k for validation and testing, respectively.

For supervised baselines, we varied the training samples to be 100k, 50k, and 20k pairs, so that we could evaluate the effect of different parallel data sizes in supervised training. For unsupervised paraphrase generation, we only need a non-parallel corpus to train the language model. The sentences in the test set, however, are questions, so it is improper to use generic language models (e.g., trained on One-Billion Corpus) to judge the likelihood of a question. Instead, we trained language models on all the training samples that do not appear in the validation and test sets. The language models are of the same structure as the ones for keywords-to-sentence generation, except that we reduce the dictionary size to 30k because of fewer training samples.

Previous work uses BLEU score (Papineni et al. 2002) against a ground truth reference (denoted as BLEU-ref) to evaluate the quality of the quality of generated paraphrase (Gupta et al. 2017). We observe that it is insufficient because simply copying the input sentence itself yields the highest BLEU-ref score (Table 4). We thus propose to use BLEU score against the original input sentence (denoted as BLEU-orig) as an auxiliary measure. Ideally, BLEU-ref should be high, whereas BLEU-ori should be low. We tried different initialization methods, including using exact or corrupted original sentences. We also attempted to start from a totally random state. As a lot of samples are generated, we choose the first sentence with BLEU-ori score lower than 55 to compare with other models. (The number is chosen empirically in order to get paraphrases with obvious literal difference.)

We compare our approach with supervised methods, including sequence-to-sequence models, VAE-SVG and VAE-SVG-eq (Gupta et al. 2017). VAE-SVG is a VAE conditioned on the original sentence, and VAE-SVG-eq is a variant of VAE-SVG which shares parameters between encoder and decoder.

We would also like to compare paraphrase generator in the unsupervised setting as our CGMH. However, we cannot find an existing dedicated model. We find it possible to train a variational autoencoder (VAE) with non-parallel corpus and sample sentences from the variational latent space (Bowman et al. 2016).

Table 4 shows that our methods, compared with VAE, achieve a fairly close BLEU-ref score to the best supervised approaches. Moreover, CGMH even outperforms the supervised methods when the training set is not large enough ($\leq$50k).

Admittedly, CGMH has higher BLEU-ori scores than supervised methods, indicating that the generated samples are closer to the input. This, however, makes much sense because CGMH samples sentences from a distribution specified in an unsupervised fashion, as opposed to rewriting words and phrases in an *ad hoc* fashion to make the expressions different, as is learned in the supervised setting. However, we only consider paraphrases with BLEU-ori less than 55, which has assured a significant literal difference. Future research could address this problem by designing proper heuristics to manipulate the stationary distribution, which is beyond the scope of our paper (but shows the flexibility of our model).

Table 5 shows examples of generated paraphrases. We see qualitatively that CGMH yields fairly good samples in terms of both closeness in meaning and difference in expressions. Table 6 gives a real example of the paraphrase generation process with CGMH.

Unsupervised Error Correction

We evaluated our method on JFLEG (Napoles, Sakaguchi, and Tetreault 2017),[4] a newly released dataset for sentence correction. It contains 1501 sentences (754 for validation and 747 for test), each with 4 revised references. We use GLEU (Napoles et al. 2015), which measures sentence fluency and grammaticality as the evaluation metric.

[4] https://github.com/keisks/jfleg

Model	#parallel data	GLEU
AMU	2.3M	44.85
CAMB-14	155k	46.04
MLE	720k	52.75
NRL	720k	**53.98**
CGMH	0	45.5

Table 7: Results of different models on sentence correction.

Ori	Even if **we are failed** , We have to try to get **a new things** .
Ref	Even if we all failed , we have to try to get new things .
Gen	Even if we are failing , We have to try to get some new things .
Ori	In the world **oil price very high** right now .
Ref	In today 's world , oil prices are very high right now .
Gen	In the world , oil prices are very high right now .

Table 8: Examples of sentence correction by CGMH.

This benchmark dataset does not contain training samples. Various studies have not only proposed new models, but also collected parallel data for themselves, each containing millions of samples, which is shown in Table 7. However, we used none of them.

We adopted the same language models (trained on One-Billion-Word) as in keywords-to-sentence generation to approximate sentence probabilities. For MH sampling, we start from the original sentence with mistakes, and simply output the 100th sample. As the original erroneous sentence has low probability from a language model perspective, the goal of sentence correction can be formulated as jumping to a nearby sentence with high probability. We would like to encourage MH to explore more probable states by further rejecting proposals if the likelihood is becoming too small.

The performance of CGMH on error correction is surprisingly promising, as in Table 7. CGMH achieves comparable results to state-of-the-art supervised systems including a rule-based system, CAMB14 (Felice et al. 2014). CGMH even outperforms the AMU system (Junczys-Dowmunt and Grundkiewicz 2016), which is built on phrase-based machine translation with 2.3M parallel training data and intensively engineered linguistic features. We observe a small performance gap between CGMH and MLE (Maximal Likelihood Estimation) as well as NRL (Neural Reinforcement Learning) (Sakaguchi, Post, and Van Durme 2017). Nevertheless, our initial success of CGMH shows a promising direction of unsupervised error correction.

Model Analysis

Despite successful applications in previous experiments, we now analyze CGMH in more detail.

Acceptance rate and ergodicity. As is known, it is difficult to quickly traverse the states of a high-dimensional Markov chain, where low acceptance rate is a major problem. Table 9 shows the acceptance rate in the paraphrase generation task. We see that the word replacement has 100% acceptance rate as it is essentially a Gibbs step, guaranteed by Equation 7. For word deletion and addition, the acceptance rate is lower, but still in a reasonable range; it allows the sampler to generate sentences with different lengths, as opposed to Gibbs sampling. In our experiment, it takes about

Model	Rep	Add	Del	Mean
CGMH *w/o matching*	100	9.2	5.1	32.9
w/ KW	100	8.0	4.4	32.5
w/ KW + WVM	100	10.8	2.9	32.7

Table 9: Acceptance rate (%) in the paraphrase generation task. Word replacement has 100% acceptance rate as shown in Equation (8).

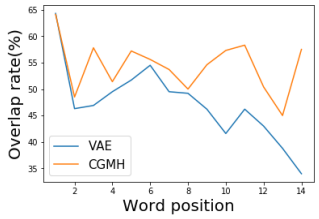

Figure 2: Overlap rates of CGMH and VAE for each word position of sentences.

150 steps to obtain a fluent sentence from a sequence of keywords. For paraphrase generation, it takes less than 50 steps for more than 20% of words being changed, showing that CGMH is efficient for practical use.

Comparison with VAE. We would like to compare CGMH, which samples from the sentence space, with VAE, which samples from the latent space. VAE is a probabilistic model that imposes a prior distribution on the latent space and then decodes a sentence in a deterministic manner by a RNN. In practice, we observe the variance of VAE samples will increase as the generation proceeds. This is shown by the blue curve in Figure 2, as the word overlap rate (the ratio of the reference sentences containing words at a specific position of the generated ones) goes down for words far from sentence beginning. This is possibly because RNN can be thought of as an autoregressive Bayesian network generating words conditioned on previous ones. Hence error will accumulate during generation. However, CGMH does not severely suffer from this problem, because there is not an explicit generation direction (order) for CGMH. At the same time, CGMH has the ability to self-correct, which is shown in the experiment of error correction.

Conclusion

In this paper, we present CGMH for constrained sentence generation by Metropolis-Hastings (MH) sampling, where we use word operations including replacement, insertion, and deletion as proposals, and design several stationary distributions for different tasks. We evaluated our results on keywords-to-sentence generation, paraphrase generation, and error correction. Our CGMH framework not only makes unsupervised learning feasible in these applications, but also achieves high performance close to state-of-the-art supervised approaches.

Acknowledgement

This work was done during Ning Miao's internship at ByteDance AI lab and is supported by the National Key Research and Development Program of China (No.

2017YFC0804001) and the National Science Foundation of China (No. 61876196, No. 61672058). Corresponding author: Hao Zhou and Rui Yan.

References

Anderson, P.; Fernando, B.; Johnson, M.; and Gould, S. 2017. Guided open vocabulary image captioning with constrained beam search. In *EMNLP*.

Berglund, M.; Raiko, T.; Honkala, M.; Kärkkäinen, L.; Vetek, A.; and Karhunen, J. T. 2015. Bidirectional recurrent neural networks as generative models. In *NIPS*.

Bowman, S. R.; Vilnis, L.; Vinyals, O.; Dai, A.; Jozefowicz, R.; and Bengio, S. 2016. Generating sentences from a continuous space. In *CoNLL*.

Chelba, C.; Mikolov, T.; Schuster, M.; Ge, Q.; Brants, T.; Koehn, P.; and Robinson, T. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Chollampatt, S.; Hoang, D. T.; and Ng, H. T. 2016. Adapting grammatical error correction based on the native language of writers with neural network joint models. In *EMNLP*.

Felice, M.; Yuan, Z.; Andersen, Ø. E.; Yannakoudakis, H.; and Kochmar, E. 2014. Grammatical error correction using hybrid systems and type filtering. In *CoNLL*.

Gelman, A.; Carlin, J. B.; Stern, H. S.; Dunson, D. B.; Vehtari, A.; and Rubin, D. B. 2013. *Bayesian Data Analysis*. CRC Press.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Gupta, A.; Agarwal, A.; Singh, P.; and Rai, P. 2017. A deep generative framework for paraphrase generation. *arXiv preprint arXiv:1709.05074*.

Guu, K.; Hashimoto, T. B.; Oren, Y.; and Liang, P. 2018. Generating sentences by editing prototypes. *TACL*.

Harrison, B.; Purdy, C.; and Riedl, M. 2017. Toward automated story generation with markov chain monte carlo methods and deep neural networks. In *AAAI*.

Hasler, E.; De Gispert, A.; Iglesias, G.; and Byrne, B. 2018. Neural machine translation decoding with terminology constraints. *arXiv preprint arXiv:1805.03750*.

Hinton, G. E.; Osindero, S.; and Teh, Y.-W. 2006. A fast learning algorithm for deep belief nets. *Neural Computation*.

Hokamp, C., and Liu, Q. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *ACL*.

Hu, Z.; Yang, Z.; Liang, X.; Salakhutdinov, R.; and Xing, E. P. 2017. Toward controlled generation of text. In *ICML*.

Junczys-Dowmunt, M., and Grundkiewicz, R. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. *arXiv preprint arXiv:1605.06353*.

Kiros, R.; Zhu, Y.; Salakhutdinov, R. R.; Zemel, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-thought vectors. In *NIPS*.

Li, Z.; Jiang, X.; Shang, L.; and Li, H. 2017. Paraphrase generation with deep reinforcement learning. *arXiv preprint arXiv:1711.00279*.

Li, J.; Jia, R.; He, H.; and Liang, P. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *NAACL*.

Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; and Teller, E. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics*.

Mou, L.; Yan, R.; Li, G.; Zhang, L.; and Jin, Z. 2015. Backward and forward language modeling for constrained sentence generation. *arXiv preprint arXiv:1512.06612*.

Mou, L.; Song, Y.; Yan, R.; Li, G.; Zhang, L.; and Jin, Z. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *COLING*.

Napoles, C.; Sakaguchi, K.; Post, M.; and Tetreault, J. 2015. Ground truth for grammatical error correction metrics. In *ACL*.

Napoles, C.; Sakaguchi, K.; and Tetreault, J. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. *arXiv preprint arXiv:1702.04066*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Post, M., and Vilar, D. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. *arXiv preprint arXiv:1804.06609*.

Prabhumoye, S.; Tsvetkov, Y.; Salakhutdinov, R.; and Black, A. W. 2018. Style transfer through back-translation. In *ACL*.

Prakash, A.; Hasan, S. A.; Lee, K.; Datla, V.; Qadir, A.; Liu, J.; and Farri, O. 2016. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*.

Rose, S.; Engel, D.; Cramer, N.; and Cowley, W. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*.

Sakaguchi, K.; Post, M.; and Van Durme, B. 2017. Grammatical error correction with neural reinforcement learning. *arXiv preprint arXiv:1707.00299*.

Shen, T.; Lei, T.; Barzilay, R.; and Jaakkola, T. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*.

Su, J.; Xu, J.; Qiu, X.; and Huang, X. 2018. Incorporating discriminator in sentence generation: a gibbs sampling method. In *AAAI*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Yin, J.; Jiang, X.; Lu, Z.; Shang, L.; Li, H.; and Li, X. 2016. Neural generative question answering. In *IJCAI*.