Knowledge-Based Systems 00 (2023) 110886

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

End-to-end hard constrained text generation via incrementally predicting segments

Jinran Nie [a,b], Xuancheng Huang [c], Yang Liu [c], Cunliang Kong [a,b], Xin Liu [a,b], Liner Yang [a,b,*], Erhong Yang [a,b]

[a] National Language Resources Monitoring and Research Center for Print Media, Beijing Language and Culture University, China
[b] School of Information Science, Beijing Language and Culture University, China
[c] Department of Computer Science and Technology, Tsinghua University, China

ARTICLE INFO

Article history:
Received 20 September 2022
Received in revised form 1 March 2023
Accepted 3 August 2023
Available online 9 August 2023

Keywords:
End-to-end
Incrementally predicting segments
Lexical constraints

ABSTRACT

Hard-constrained text generation is an important task that requires generating fluent sentences to include several specific keywords. It has numerous real-world applications, such as advertisement generation, keyword-based summary generation, and query rewriting. Although previous plug-and-play approaches like enhanced beam search and stochastic search have been proven effective, they lack time efficiency and may reduce the quality of generated sentences. While end-to-end methods based on seq2seq models are superior in speed, they cannot guarantee that outputs satisfy all constraints. In this work, we propose a novel end-to-end method for lexically constrained text generation via incrementally predicting segments (IPS) between every two adjacent lexical constraints using seq2seq models. Our approach guarantees that all constrained keywords will be included in the generated sentence. The experimental results show that our method not only satisfies all lexical constraints but also achieves state-of-the-art performance. Our code and data will be available at https://github.com/blcuicall/IPS.

1. Introduction

Recently, hard-constrained text generation [1–5] has attracted increasing attention from the community. It aims to incorporate some specific keywords or phrases into generated sentences and has many related scenarios in the real world. For example, it can create an advertisement [2] or a story by giving several specified keywords [6–9]. Moreover, it can rewrite a search query as a fluent sentence [10–12].

As shown in Table 1, previous work for hard-constrained generation can be divided into three branches: *enhanced beam search* [13,14], *stochastic search* [2–5], and *seq2seq* based methods [15].

The *enhanced beam search* is not an end-to-end method, which enforces to reserve those candidates containing keywords and costs much time compared to beam search. It may result in poor or even failed generation because of not being aware of the constraint words or phrases in training [4,15]. The mainstream method of hard-constrained text generation is the *stochastic search* [2–5] which needs many iterations to modify the sentence. Though some attempts make the searching process more efficient or faster [4,5], they are still too slow for real-world applications.

In contrast, *seq2seq* based methods have a faster text generation speed than those two kinds of methods mentioned above because of the end-to-end manner. Quite simply, the seq2seq based methods map keywords to a sentence directly [15] with the challenge that it is hard to satisfy all the constraints. Wang et al. [15] proposed to use mention flags incorporated in the decoder to encourage the model to meet the constraints, but the satisfaction still cannot achieve 100%.

Therefore, we can predict the segments between every two keywords to realize 100% constraints satisfaction inspired by the masked language model which masked sampled span like T5 [16] or BART [17]. However, if all segments predicted at once with the given keywords, there is much information need to be predicted. As shown in Fig. 1, we propose a new manner in which segments between every two keywords will be predicted incrementally at inference time. The training objective of T5 is to predict all segments at once, while our method is to predict segments one by one. So, our method gives more information at the encoder input and requires less information at the decoder output, which is easier for predicting. Our method can guarantee that *all the keywords* are incorporated in the final generated text by placing the keywords beforehand then filling the segments between them. In addition, we design a decoding algorithm for IPS to improve the

* Corresponding author at: National Language Resources Monitoring and Research Center for Print Media, Beijing Language and Culture University, China.
E-mail address: lineryang@gmail.com (L. Yang).

https://doi.org/10.1016/j.knosys.2023.110886
0950-7051/

Table 1
Comparisons with previous work.

Method	End-to-End	Constraint (100%)
Enhanced Beam Search	✗	✓
Stochastic Search	✗	✓
Seq2seq based	✓	✗
IPS	✓	✓

Keywords : _____ **like** _____ **basketball** _____

Stage 1 : *Jack and Bob* **like** _____ **basketball** _____

Stage 2 : Jack and Bob **like** *playing* **basketball** _____

Stage 3 : Jack and Bob **like** playing **basketball** *in the park.*

Fig. 1. An example of incrementally predicting segments. Given two keywords "**like**" and "**basketball**", the generating process includes three stages, in which the three segments "*Jack and Bob*", "*playing*", and "*in the park.*" will be generated step by step. Then we get a complete sentence "*Jack and Bob* **like** *playing* **basketball** *in the park.*" including the two keywords.

generation speed by dynamically updating the batch of decoding inputs. Experimental results demonstrate that our method has 100% *constraint satisfaction* (the ratio of satisfied lexical constraints to given lexical constraints) and achieves state-of-the-art performance on hard-constrained text generation.

Our main contributions include:

- We propose a novel end-to-end method for lexically constrained text generation via incrementally predicting segments which can ensure the constrained keywords appear in the output sentence.
- We design a decoding algorithm for incrementally segments predicting through dynamically updating the batch of decoding inputs at different stages.
- Experimental results show that our method achieves state-of-the-art performance on hard-constrained text generation.

2. Related work

There are three branches in previous work for hard-constrained text generation: enhanced beam search [13,14,18], stochastic search [2–5] and seq2seq based methods [15].

Enhanced beam search. In enhanced beam search methods, grid beam search (GBS) algorithm [13] conducts beam search by adding another dimension to search for the candidate sentences that satisfy the given lexical constraints. Dynamic beam allocation (DBA) [14,18] is the extension of GBS which aim to accelerate the inference process. Although they work well in tasks with limited search space such as machine translation because of the alignment information between source and target, it will cost a lot of time on searching candidate sentences or even fail in general text generation tasks when there is a much larger search space [4,5]. Therefore, more work focus on stochastic search methods. Unlike the methods of enhanced beam search, our method does not only consider lexical constraints in the prediction process but takes constraints into account in both the training and prediction process. As the input of our model, constraint words are encoded and understood by the model encoder in the process of training and prediction to help model generation. Therefore, our method should theoretically have better generation quality.

Stochastic search. In stochastic search methods, early research used Gibbs sampling to generate sentences from the sentence space directly [19], which was extended to BERT [20]. Compared with Gibbs sampling, CGMH [2] is more superior to generate sentences through Metropolis–Hastings sampling. However, previous MCMC-based models typically conduct many invalid and redundant refinements because of the randomly chosen actions and positions [5]. Several works [3–5] have focused on addressing this issue. He and Li [5] proposed to use a pre-trained classifier [21] to predict the position and operation. Moreover, Sha [4] utilized the gradient to help determine which token in the sequence should be changed. To accelerate the iteration speed, POINTER [3] was proposed to insert new tokens between existing tokens in a parallel manner. Though these methods can improve the iteration precision and speed, insertion-based methods need many iterations to find a fluent sentence which cost too much time in real-world applications. Our method is an end-to-end method which is completely different from the methods of stochastic search. Instead of iteratively modifying sentences, our model generates text end-to-end. Therefore, our model generation speed is much faster than the stochastic search methods.

Seq2seq based. The end-to-end method for lexically constrained text generation utilizes a seq2seq model by giving the constraints as input and the target sentence as output for training. It is superior in speed compared to enhance beam search methods and stochastic search methods. But it is hard to satisfy all constrained lexical. Mention flag [15] was proposed to increase the proportion of satisfaction for end-to-end lexically constrained generation, which has been used in common sense generation task [22–25]. However, it still cannot achieve 100% constrained satisfaction. Therefore, our method aims to generate fluent text by satisfying all given lexical constraints by seq2seq model, for which the generation speed is significantly faster than previous insertion-based methods. Different from the common end-to-end generation approach, our model is divided into multi-stage prediction segments. The goal is to ensure that the generated text contains all constraint words. Our method of incremental prediction segment can fully guarantee the inclusion of all constraint words.

In summary, our method is a kind of seq2seq-based method, which is different from stochastic search methods because there is no iteration to modify the sentence step by step. Thus, our method has an absolutely fast speed compared to the stochastic search methods. The keywords constraints are considered both in training and inference time in our method. But it does only considered in inference time for enhanced beam search, which will impair the quality of generation. In addition, our method can achieve 100% constrained satisfaction compared to previous seq2seq methods.

3. Method

3.1. Model overview

Hard-constrained text generation aims to incorporate the given lexical constraints (we call them keywords) into the output [3]. Given the n keywords $\mathbf{k} = \{k_1, k_2, \ldots, k_n\}$, this task is defined as follows:

$$S^* = \arg\max_S P\left(S|k_1, k_2, \ldots, k_n\right) \tag{1}$$

where S is the sentence containing the given lexical constraints. For the hard-constrained text generation task, the goal is to generate a complete text sequence given a set of keywords as constraints, where the keywords have to be exactly included in the final generated sequence in the same order.

The seq2seq based method directly uses keywords as input and complete sentences as model output [15]. This method has two disadvantages. First, the output sentence does not be forced to contain all keywords because the method samples each word

from the whole vocabulary. It is unable to guarantee 100% constrained satisfaction due to random sampling. Second, the model needs a lot of information to predict the complete sentences, but the input keywords only provide limited information, making the learning process more difficult.

The method that we proposed can not only suitably address these two issues but also incrementally predicts each segment one by one. First of all, we use the keywords-to-text generation to fill in the segment between every two keywords. Hence the model predicts a segment in order at each stage, and fills the prediction result in the next stage. Because the keywords are fixed at the beginning of the generation process, the model does not need to predict keywords but only the segments between keywords. Therefore, it can be guaranteed that the keywords are included in the generated sentences. Second, this method allows the model to predict only one segment at one stage rather than the entire sentence. Compared with the method that directly uses keywords as input and complete sentences as output, our method fuses more information from the source. Furthermore, the target end needs less information to predict. Thus the model is easier to learn from data.

We will give the formulaic description of our method below. The denotation of S is given by

$$S = \left\{ \mathbf{s}_1, k_1, \mathbf{s}_2, k_2, \ldots, \mathbf{s}_i, k_i, \ldots, \mathbf{s}_n, k_n, \mathbf{s}_{n+1} \right\} \tag{2}$$

where k_i denotes the ith keyword, n denotes the number of given keywords, and $\mathbf{s}_i$ denotes the ith segment which is between k_{i-1} and k_i.

We divide the generation process into $n+1$ stages and every stage predict one segment by the order. The formulation is:

$$P(\mathbf{s}_i|\mathbf{s}_{<i}, \mathbf{k}) = \prod_{j=1}^{|\mathbf{s}_i|} P(s_{i,j}|\mathbf{s}_{i,<j}, \mathbf{s}_{<i}, \mathbf{k}) \tag{3}$$

where $\mathbf{s}_i$ denotes the ith segment, $\mathbf{s}_{<i}$ denotes the segments before the ith segment, $|\mathbf{s}_i|$ denotes the length of ith segment, $s_{i,j}$ denotes the jth token in the ith segment and $\mathbf{s}_{i,<j}$ denotes the tokens before the jth token in the ith segment.

To implement the above modeling and prediction, we develop a training and prediction process. First, the training data needs to be designed elaborately with specific tags after extracting keywords. Then, given the constructed paired data, a seq2seq model is trained by the autoregressive pattern. In inference time, we design a multi-stages prediction and decoding algorithm. We will describe the details of these steps in the following subsections.

3.2. Specific tags

Given the tokens of $\mathbf{s}_{i,<j}$, $\mathbf{s}_{<i}$, $\mathbf{k}$ alone, the model cannot distinguish between segments and keywords as well as which segment needed to be predicted. Therefore, we adopt the method of adding specific tags to explicitly promote the model to learn where the segment needs to be predicted at the current stage and where the segments are vacant and need to be predicted in the following stages. As shown in Fig. 2, which will be described in detail below, the specific token `[pred]` is placed at the position of the segment that needs to be predicted at the current stage. The vacant segments at the current stage (need to be predicted in the following stages) are represented by `[blank]`.

3.3. Training data preparation

To accomplish the above modeling process, we need to construct the corresponding training data. First, we extract some keywords from an unlabeled sentence. Then the unlabeled sentence is divided into multiple segments according to the number of keywords and their positions in the sentence. The segments are the consecutive tokens before the first keyword, after the last keyword, and between every two keywords. If the two keywords are adjacent, the segment is replaced with a space while its position is still reserved.

Keywords extraction. We use the keywords extracting tool YAKE[1] [26] to get keywords from a sentence. This tool will try its best to extract more keywords from one sentence, but following previous work [5], we only preserve $1 \sim 4$ keywords. To limit the number of keywords, we set a rule that is proportional to the length of the sentence, and the maximum count is 4. More details are in Section 4.

Paired data construction. According to the position of the segments in the sentence, we replace each segment with `[pred]` in turn and replace segments behind them with `[blank]`. In detail, we replace the first segment with `[pred]` and replace all the segments after it with `[blank]` to get the source of the first paired data. And the target is the first segment concatenated with `[bos]` before it and `[eos]` after it. Hence, we will get the first paired data.

Then, to construct the second paired data, we replace the second segment with `[pred]` and the segments after it with `[blank]` but keep the original tokens of the first segment without any changes. Similarly, the target of the second paired data consists of `[bos]`, the second segment, and `[eos]`. If there are n keywords, we process them in a similar way to construct $n+1$ paired data.

Fig. 2 is an example. From the sentence "Jack and Bob like playing basketball in the park ". we get two keywords: "like" and "basketball". Then we divide the sentence into three segments "Jack and Bob", "playing", and "in the park ". according to the position of two keywords. First, we replace the first segment "Jack and Bob" with `[pred]`, the second and third segments with `[blank]`. And we get the first pair: {X^1: `[pred]` like `[blank]` basketball `[blank]`, Y^1: `[bos]` Jack and Bob `[eos]` }. Then we replace the second segment "playing" with `[pred]`, the segment after it (i.e., "in the park ".) with `[blank]`, and the segment before it (i.e., "Jack and Bob") remain unchanged. And we get the second pair {X^2: Jack and Bob like `[pred]` basketball `[blank]`, Y^2: `[bos]` playing `[eos]` }. Finally, we replace the third segment "in the park ". with `[pred]` and the segments before it (i.e., "Jack and Bob" and "playing") remains unchanged. Then we get third paired data {X^3: "Jack and Bob like playing basketball `[pred]`", Y^3: `[bos]` in the park `[eos]` }. Therefore, for one sentence with n keywords, we will get $n+1$ pairs $\{\langle X^1, Y^1\rangle, \langle X^2, Y^2\rangle, \ldots, \langle X^{n+1}, Y^{n+1}\rangle\}$ for training. Through the process mentioned above, we construct training data from all unlabeled text in the corpus.

It is worth noting that since the replaced segment may be empty (when the two keywords are adjacent in the sentence), the target segment is added with `[bos]` at the beginning and `[eos]` at the end, while the `[bos]` and `[eos]` tags are removed from the output result during generation.

3.4. Model training and inference

For each unlabeled sentence, we construct several paired data as described in Section 3.3. Then, we mix all paired data and shuffle it to obtain a final training set D_d. We optimize the transformer model [27] using the following objective:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left\{ \sum_{\langle X,Y\rangle \in D_d} \log P(Y|X, \boldsymbol{\theta}) \right\} \tag{4}$$

[1] https://github.com/LIAAD/yake

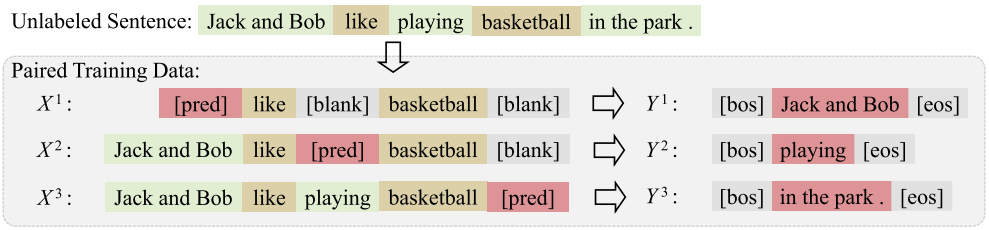

Fig. 2. Training data preparation. Given one sentence which contains two keywords "like" and "basketball", we will construct three pairs of data, one for each segment.

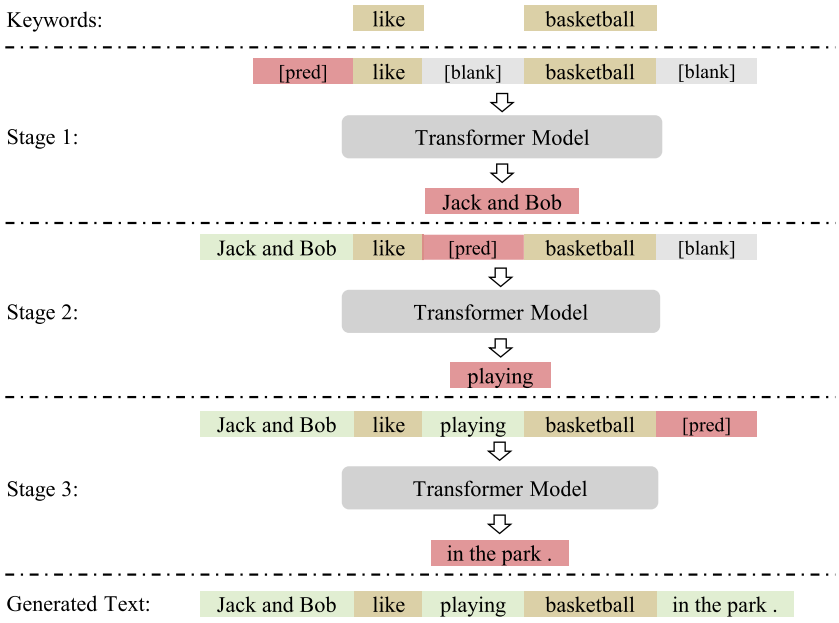

Fig. 3. Illustration of generation. Given two keywords "like" and "basketball", there are three stages to predict three segments incrementally. In each stage, the tag [pred] in the source will be replaced with the output segment and the first [blank] in the input will be replaced with [pred] to build the input of the next stage. Finally, a complete sentence "Jack and Bob like playing basketball in the park". will be generated.

where $\boldsymbol{\theta}$ denotes the model parameters, $\langle X, Y \rangle$ denotes one training instance. During training, we apply teacher-forcing to train the model to learn predicting segments conditioned on partial sentences. As the inference procedure illustrated in Fig. 3, we fill the generated segment into the `[pred]` of the previous stage. Then, we continue to predict the next segment, and incrementally append the segment until all the segments are filled, thereby generating a complete sentence.

Decoding algorithm. We design a parallel decoding algorithm to promote inference speed for IPS. The challenge of batched inference is that the number of given keywords of each sentence in one batch is different, leading to early finish of some sentences then the efficiency will be reduced. Hence we utilize a dynamic batching strategy to maintain the batch size. The details are described in Algorithm 1. Given t groups of keywords in the test set, each group has several keywords which will be incorporated in the generated sentence, and it will generate t sentences by mini-batch. First, we initialize the model and get the input data I of the first stage by adding `[pred]` and `[blank]` tags, from which we get the initialized mini-batch input B. Then, we get the segments ϕ from the model. Before the next stage, we need to find which samples have been finished in the mini-batch input and add the same number of samples into the mini-batch. Next, we will replace the `[pred]` with the generated segment and replace the next `[blank]` with `[pred]`. Then, continue the next stage of segments generation in the loop. Finally, return all the finished samples when the loop end.

4. Experiments

4.1. Experimental setup

Algorithm 1 IPS batched decoding.

input: The t groups of keywords **K** in test set
output: Generated t sentences **S**
Initialize parameters of model $\rightarrow$ **MODEL**;
Add `[pred]` and `[blank]` to **K** $\rightarrow$ *Inp*
$B \leftarrow$ Get batch data from I
while I is not **None do**
  $\phi \leftarrow$ **MODEL**(B)
  for s_i, b in ϕ, B **do**
    $s_i \leftarrow$ remove `[bos]` and `[eos]` from s_i
    $s_i \leftarrow$ replace `[pred]` in b with s_i
    if [blank] in s_i **then**
      $s_i \leftarrow$ replace next `[blank]` with `[pred]`
    else
      S.append(s_i)
      B.append(pop(*Inp*))
    end if
  end for
end while
return S

Table 2
Automatic evaluation results on One-billion-word dataset.

Method	Avg. Len.	Constraint (%)	Entropy	NIST (%)		BLEU (%)		METEOR (%)	PPL
				N-2	N-4	B-2	B-4		
Ground Truth	27.14	100.00	12.24	–	–	–	–	–	76.53
X-MCMC-C	16.86	**100.00**	**11.69**	1.94	1.96	11.10	3.39	13.69	163.57
Trans+DBA	22.60	**100.00**	11.04	2.88	3.02	18.12	7.80	15.41	196.65
Trans-K2S	19.01	98.91	11.13	2.32	3.03	19.33	9.26	16.01	95.84
T5	22.22	**100.00**	11.43	4.13	4.25	21.36	9.76	16.62	81.94
IPS	24.71	**100.00**	11.59	**4.35**	**4.49**	**22.35**	**10.21**	**16.72**	**58.89**

Table 3
Automatic evaluation results on News dataset.

Method	Avg. Len.	Constraint (%)	Entropy	NIST (%)		BLEU (%)		METEOR (%)	PPL
				N-2	N-4	B-2	B-4		
Ground Truth	29.86	100.00	11.77	–	–	–	–	–	37.45
X-MCMC-C	16.49	**100.00**	11.07	1.09	1.09	8.01	2.07	11.94	160.38
Trans+DBA	26.95	**100.00**	10.08	3.86	3.93	18.80	7.09	15.23	65.28
Trans-K2S	24.96	99.02	10.67	3.77	3.85	18.59	7.43	15.24	37.90
T5	25.22	**100.00**	10.82	3.89	3.97	19.32	7.90	15.58	35.34
IPS	28.14	**100.00**	**11.10**	**4.01**	**4.09**	**20.25**	**8.13**	**15.89**	**29.80**

Datasets and pre-processing. Following the previous work [3,5], we evaluate our model on two datasets: One-Billion-Word and News. The first one contains 10M sentences as the training set, 60 K sentences as the validation set, and 10 K sentences as the test set, which are randomly picked from One-Billion-Word Corpus. Second, we use the monolingual English News (2018) dataset from ACL 2019 WMT, which contains 18M sentences. We pick our data from those sentences of which the length is limited from 10 to 100 words. We get 5M sentences as the training set, 50 K sentences as the validation set, and 5k sentences as the test set.

Then we use the keywords extracting tool to get the keywords-sentence pair from the two datasets above. To overcome the weakness of the tool, we set a rule to constrain the number of keywords depending on the length of the sentence. The longer the sentence, the more keywords. We set the number of keywords as 15% of the sentence length, and the max number of keywords is 4. Thus the number of keywords varies from 1 to 4. Because the length of most sentences is long, most sentences have 4 keywords. Finally, we constructed our training and test data on the keywords-sentence pair above by the method described in Section 3.3.

Baselines. We compare our approach (IPS) with three state-of-the-art methods from the three branches in hard-constrained text generation mentioned in Section 1: (1) X-MCMC-C [5] which is a relatively new method in the category of stochastic search, (2) DBA [14] which is a constraint decoding method, and we set it on transformer model (Trans+DBA), (3) the transformer model which put the keywords as input and sentence as output (Trans-K2S), and (4) We use the pre-training objective of T5 [16] but do not use it for initialization. As far as we know, we are the first to utilize T5 in this way in hard-constrained text generation to satisfy all constraints. Then we apply our method to the transformer model to compare with these three methods. The input and output of our method are different from the input and output in T5. Our method will provide more information in the source and predict one segment in the target, while T5 model will predict all segments at once. So, our method gives more information at the encoder input and requires less information to predict at the decoder output.

Evaluation metrics. Our automatic evaluation has four aspects: consistency with the ground truth, fluency, diversity, and constraint satisfaction. Following the previous work [3,5], we perform automatic evaluations using commonly adopted text generation metrics, including BLEU [28], METEOR [29], and NIST [30] to measure the differences from the references. Following Zhang et al. [3],He and Li [5], to assess the fluency of generated sentences, we also report the perplexity (PPL) over the test set using the pre-trained GPT-2 [31] large model.[2] We use 4-gram Entropy [32] to evaluate lexical diversity. And we set the ratio of keywords contained in the output sentence as a metric to evaluate the constraint satisfaction. Following previous studies [3], our human evaluation will perform from three aspects: the consistency between keywords and generated sentence, the fluency, and the informativeness of generated sentences.

Model setup. We employ the model architecture from the typical Transformer [27]. We use a learning rate of 3e-4 and set a warming-up schedule with 4000 steps for the training procedures. The optimization algorithm is Adam [33]. We train our model on One-billion-word and News for around 20 epochs. At inference time, we utilized a beam search algorithm and set the beam size as 5. By the way, we also set the length penalty as 2 in beam search by fairseq.[3] We set the maximum number of input tokens as 8192, which is the same with transformer-based baselines (Trans+DBA and Trans-K2S).

4.2. Experimental results

Results on one-billion-word dataset. Quantitative results by automatic metrics are summarized in Table 2. The automatic evaluation results show that our method outperforms baselines in several metrics (NIST, BLEU, METEOR, Entropy-4, and PPL). The NIST, BLEU, and METEOR evaluate the similarity between the generated sentence and human references. A higher score indicates a model can generate sentences more similar to human references. Another essential evaluation metric is PPL which indicates the fluency of the generated sentences. One of the most critical points of the hard-constrained text generation task is the fluency of the generated sentences. As shown in Table 2, our method gets the best PPL score compared to all baseline even is better than human references. Because the X-MCMC-C is a stochastic searching method, its diversity (Entropy score) is the best. In fact, when decoding by beam search, our method gets the highest score on Entropy-4. Using the beam search algorithm and giving length

[2] https://huggingface.co/transformers
[3] https://github.com/pytorch/fairseq

Table 4
Human Evaluation on two datasets for semantics, fluency and informativeness.

Semantics: A and B, which is more semantically meaningful and consistent?									
	One-Billion-Word dataset					News dataset			
	System A	Neutral		System B		System A	Neutral		System B
IPS	**71.3**%	11.0%	17.7%	X-MCMC-C	IPS	**57.5**%	5.5%	37.0%	X-MCMC-C
IPS	**37.7**%	37.3%	28.0%	Trans+DBA	IPS	**48.3**%	13.7%	38.0%	Trans+DBA
IPS	**47.2**%	32.0%	20.8%	T5	IPS	40.2%	18.2%	**41.6**%	T5
Fluency: A and B, which is more grammatical and fluent?									
	One-Billion-Words dataset					News dataset			
	System A	Neutral		System B		System A	Neutral		System B
IPS	**46.3**%	35.4%	18.3%	X-MCMC-C	IPS	**56.8**%	12.7%	30.5%	X-MCMC-C
IPS	36.3%	**44.4**%	19.3%	Trans+DBA	IPS	**50.2**%	15.3%	34.5%	Trans+DBA
IPS	**48.8**%	29.5%	21.7%	T5	IPS	42.0%	14.8%	**43.2**%	T5
Informativeness: A and B, which is more informative?									
	One-Billion-Words dataset					News dataset			
	System A	Neutral		System B		System A	Neutral		System B
IPS	22.7%	9.3%	**68.0**%	X-MCMC-C	IPS	**56.0**%	10.1%	33.9%	X-MCMC-C
IPS	**48.0**%	23.3%	30.4%	Trans+DBA	IPS	**54.5**%	14.3%	32.7%	Trans+DBA
IPS	34.5%	22.2%	**43.3**%	T5	IPS	**48.3**%	18.7%	33.0%	T5

Table 5
Speed comparisons. Inference time is computed on 5K test examples in News dataset.

Method	Inference time
X-MCMC-C	21 h
Trans+DBA	3.6 h
Trans-K2S	126 s
T5	126 s
IPS	228 s

penalty in decoding can further improve most automatic metrics we evaluated. We choose the comprehensive results to show in Table 2 considering all evaluation metrics.

Results on news dataset. We further evaluate our method on the News dataset. On the News dataset, our method outperforms all baselines on all automatic metrics. From Table 3, the generated sentences from our model are more coherent with references (see NIST, BLEU, and METEOR) and more fluent (see PPL) compared with the baseline methods. For diversity, we also achieve the highest Entropy score. And stochastic search method (X-MCMC-C) has better diversity than the Trans+DBA, Trans-K2S, and T5. The result is similar to what is in the last dataset, in which the beam search decoding and length penalty still can help to promote the performance of our method, and we only give comprehensive results.

Human evaluation results. We conducted a human evaluation of 300 randomly sampled outputs (out of test set) of X-MCMC-C, Trans+DBA, T5 and our baseline method with greedy decoding. Note that we remove all tricks like beam search and length penalty to compare with baselines. Following the previous work [3], our human evaluation is performed from three aspects: semantics, fluency, and informativeness. Three experts vote for which system is better from the three aspects, and the final results are averaged percentages of the total "vote" of the three experts. As shown in Table 4, compared to the three baselines (X-MCMC-C, Trans+DBA and T5) which can satisfy all constraints, our method has a greater percentage of sentences with higher fluency and semantic scores on the One-billion-word dataset. And our method outperforms the two baselines (X-MCMC-C and Trans+DBA) in the News dataset. And on the News dataset, we have better diversity than T5, while the semantics and fluency are similar.

Analysis. From automatic and human evaluation results, the method we adopt has better performance not only on fluency and consistency than the baselines on both datasets but also shows better diversity on the News dataset. Our method only needs to predict a segment at each stage rather than predict the whole sentence at once. It is easier for the model to predict a segment conditioned on the partial sentence than predict a complete sentence conditioned on keywords only because the information gap between source and target is smaller for the former one. In addition, the roles of two tags (`[pred]` and `[blank]`) are very important. The `[pred]` and `[blank]` tags tell the model which segment needs to be predicted and which segments are unknown in the current stage, respectively.

Inference running-time comparison. Compared to stochastic search or enhanced beam search methods, end-to-end methods has a huge advantage in decoding speed. We compare the inference time for X-MCMC-C [5], Trans+DBA [14] and IPS on the News dataset, and summarize the results in Table 5. The experiment is performed on a single Nvidia TITAN-XP GPU. Note that our method in Table 5 uses our parallel decoding algorithm. Our method (IPS) is much faster than baselines because of the seq2seq manner. The stochastic search method, X-MCMC-C, needs several iterations to find a fluent sentence for a group of keywords. Hence it costs much time in the generation. And enhanced beam search method Trans+DBA needs to expand beam size. Thus it is computationally expensive.

Case study. Some examples for all baselines and ours are provided in Tables 6 and 7. We find that compared with baseline generations, the sentences generated by our method contain more details (e.g., "WTO") and are more vivid with some metaphors (e.g., "in good faith" and "very happy") while ensureing language fluency simultaneously. According to these examples, the sentences that our method generated are slightly longer than the baselines and are more diverse.

5. Conclusion and future work

We have proposed a novel end-to-end method for hard-constrained text generation by incrementally predicting segments using the seq2seq model. Through automatic and manual evaluation, it is able to not only generate sentences including all the given keywords but also achieve state-of-the-art performance compared with previous methods. In addition, compared with

Table 6
Generated examples from the One-Billion-Word dataset.

Method	*Keywords*: World; Health; Organisation; Uganda
X-MCMC-C	The **World Health Organisation** said **Uganda** did not yet have any new swine flu cases.
Trans+DBA	According to the **World Health Organisation**, **Uganda** has the highest number of confirmed cases of swine flu in the world.
Trans-K2S	According to the **World Health Organisation**, **Uganda** has the highest number of confirmed cases of swine flu in the world.
T5	The **World Health Organisation** said the virus was spreading to **Uganda**, where it was confirmed that it was spreading to the UK.
IPS	The **World Health Organisation** (WHO) said last week that **Uganda**, the world's largest producer of swine flu vaccine, had become the first country in the world to report cases of swine flu.

Table 7
Generated examples from the News dataset.

Method	*Keywords*: Thursday; morning; Trump; Singapore
X-MCMC-C	On **Thursday morning** US President Donald **Trump** visited **Singapore**
Trans+DBA	**Thursday morning Trump Singapore** said he would not be able to meet with the US president again.
Trans-K2S	On **Thursday morning**, **Trump** said he would meet with Kim in **Singapore** "to discuss the denuclearization of the Korean peninsula."
T5	On **Thursday morning** , Mr. **Trump** said he was "very happy " to meet with Mr. Xi in **Singapore**.
IPS	On **Thursday morning**, **Trump** said he would be leaving **Singapore** "in good faith" and that he would be "very happy" with the outcome.

the methods of enhanced beam search and stochastic search, our method has superior generation speed. Our method consists of novel training and inference approach that can be generalized to different models, including seq2seq models and pre-trained language models. In the future, we will try to perform it on pre-trained models through fine-tuning. The proposed method can also be used as a learning objective of the pre-trained model, that is, to predict fragments given constraints. This is a challenging training objective that is not yet taken into account by existing pre-trained models. In the future, we will try to apply our method to the pre-trained large-scale language models, which may achieve better performance on different tasks. This will contribute to the development of pre-trained language models.

CRediT authorship contribution statement

Jinran Nie: Conceptualization, Methodology, Software, Writing – original draft. **Xuancheng Huang:** Writing – review & editing. **Yang Liu:** Conceptualization, Investigation, Project administration, Supervision, Writing – review & editing. **Cunliang Kong:** Validation. **Xin Liu:** Resources. **Liner Yang:** Conceptualization, Investigation, Project administration, Supervision, Writing – review & editing. **Erhong Yang:** Conceptualization, Investigation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the funds of Research Project of the National Language Commission No. ZDI145-24.

References

[1] L. Mou, Y. Song, R. Yan, G. Li, L. Zhang, Z. Jin, Sequence to backward and forward sequences: a content-introducing approach to generative short-text conversation, in: Proceedings of the International Conference on Computational Linguistics, 2016.
[2] N. Miao, H. Zhou, L. Mou, R. Yan, L. Li, Cgmh: constrained sentence generation by metropolis-hastings sampling, in: Proceedings of the AAAI Conference on Artificial Intelligence, 33, (01) 2019, pp. 6834–6842.
[3] Y. Zhang, G. Wang, C. Li, Z. Gan, C. Brockett, B. Dolan, Pointer: constrained progressive text generation via insertion-based generative pre-training, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2020.
[4] L. Sha, Gradient-guided unsupervised lexically constrained text generation, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020, pp. 8692–8703.
[5] X. He, V.O. Li, Show me how to revise: improving lexically constrained sentence generation with xlnet, in: Proceedings of the AAAI Conference on Artificial Intelligence, 35, (14) 2021, pp. 12989–12997.
[6] A. Fan, M. Lewis, Y. Dauphin, Hierarchical neural story generation, in: Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2018.
[7] L. Yao, N. Peng, R. Weischedel, K. Knight, D. Zhao, R. Yan, Plan-and-write: towards better automatic storytelling, in: Proceedings of the AAAI Conference on Artificial Intelligence, 33, (01) 2019, pp. 7378–7385.
[8] J. Guan, F. Huang, Z. Zhao, X. Zhu, M. Huang, A knowledge-enhanced pretraining model for commonsense story generation, Transactions of the Association for Computational Linguistics 8 (2020) 93–108.
[9] P. Xu, M. Patwary, M. Shoeybi, R. Puri, P. Fung, A. Anandkumar, B. Catanzaro, Controllable story generation with external knowledge using large-scale language models, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2020.
[10] S. Yu, J. Liu, J. Yang, C. Xiong, P. Bennett, J. Gao, Z. Liu, Few-shot generative conversational query rewriting, in: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020, pp. 1933–1936.
[11] A.K. Mohankumar, N. Begwani, A. Singh, Diversity driven query rewriting in search advertising, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021.
[12] X. Liu, J. Hu, Q. Shen, H. Chen, Geo-bert pre-training model for query rewriting in poi search, in: Findings of the Association for Computational Linguistics, 2021, pp. 2209–2214.
[13] C. Hokamp, Q. Liu, Lexically constrained decoding for sequence generation using grid beam search, in: Annual Meeting of the Association for Computational Linguistics, 2017.
[14] M. Post, D. Vilar, Fast lexically constrained decoding with dynamic beam allocation for neural machine translation, in: North American Chapter of the Association for Computational Linguistics, 2018.
[15] Y. Wang, I. Wood, S. Wan, M. Dras, M. Johnson, Mention flags (mf): constraining transformer-based text generators, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, 2021, pp. 103–113.
[16] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, The Journal of Machine Learning Research 21 (1) (2020) 5485–5551.
[17] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, Bart: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7871–7880.
[18] J.E. Hu, H. Khayrallah, R. Culkin, P. Xia, T. Chen, M. Post, B. Van Durme, 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, 2019, pp. 839–850.
[19] M. Berglund, T. Raiko, M. Honkala, L. Kärkkäinen, A. Vetek, J.T. Karhunen, Bidirectional recurrent neural networks as generative models, in: Advances in Neural Information Processing Systems, 2015, pp. 856–864.
[20] A. Wang, K. Cho, Bert has a mouth, and it must speak: bert as a markov random field language model, in: Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation, 2019, pp. 30–36.

[21] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R.R. Salakhutdinov, Q.V. Le, Xlnet: generalized autoregressive pretraining for language understanding, in: Advances in neural information processing systems, 2019.
[22] Y. Liu, Y. Wan, L. He, H. Peng, S.Y. Philip, Kg-bart: knowledge graph-augmented bart for generative commonsense reasoning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 6418–6425.
[23] H. Wang, Y. Liu, C. Zhu, L. Shou, M. Gong, Y. Xu, M. Zeng, Retrieval enhanced model for commonsense generation, in: Findings of the Association for Computational Linguistics, 2021, pp. 3056–3062.
[24] Z. Fan, Y. Gong, Z. Wei, S. Wang, Y. Huang, J. Jiao, X.-J. Huang, N. Duan, R. Zhang, An enhanced knowledge injection model for commonsense generation, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 2014–2025.
[25] Y. Liu, L. Zhang, W. Han, Y. Zhang, K. Tu, Constrained text generation with global guidance–case study on commongen, arXiv preprint arXiv:2103.07170 (2021).
[26] R. Campos, V. Mangaravite, A. Pasquali, A.M. Jorge, C. Nunes, A. Jatowt, Yake! collection-independent automatic keyword extractor, in: European Conference on Information Retrieval, Springer, 2018, pp. 806–810.
[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the Advances in neural information processing systems, 2017, pp. 5998–6008.
[28] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.
[29] A. Lavie, A. Agarwal, Meteor: an automatic metric for mt evaluation with high levels of correlation with human judgments, in: Proceedings of the second workshop on statistical machine translation, 2007, pp. 228–231.
[30] G. Doddington, Automatic evaluation of machine translation quality using n-gram co-occurrence statistics, in: Proceedings of the second international conference on Human Language Technology Research, 2002, pp. 138–145.
[31] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (8) (2019) 9.
[32] Y. Zhang, M. Galley, J. Gao, Z. Gan, X. Li, C. Brockett, B. Dolan, Generating informative and diverse conversational responses via adversarial information maximization, in: Advances in Neural Information Processing Systems, 2018.
[33] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).