Coherent Long Text Generation by Contrastive Soft Prompt

Guandan Chen†
{chenguandan1}@163.com

Jiashu Pu†**, Yadong Xi and Rongsheng Zhang***
Fuxi AI Lab, NetEase Inc., Hangzhou, China
{pujiashu,xiyadong,zhangrongsheng}@corp.netease.com

Abstract

Improving the coherence of long text generation is an important but challenging task. Existing models still struggle to generate a logical and coherent sentence sequence. It is difficult for a model to plan long text generation and avoid generating incoherent texts from a high-level semantic perspective. We conjecture that this is due to two factors: (1) current training methods mainly rely on maximum likelihood estimation computed from token-level probability prediction; (2) the role of incoherent texts has been largely under-explored, thus the noised generated texts with errors are out-of-distribution for the model. To address these issues, in this paper, we propose a **C**ontrastive **S**oft **P**rompt (**CSP**) model for improving the coherence of long text generation. It learns text representations in the hidden space for better planning long text generation. To this end, it jointly learns to generate a text representation close to representations of coherent texts and away from incoherent ones, and then generates long text taking this representation as the soft prompt. We conduct experiments on two public story generation datasets, and experimental results show that our method can generate more coherent stories than the state-of-the-art model.

1 Introduction

Generating coherent long text plays a key role in many applications, e.g. news report generation (Leppänen et al., 2017), story generation (Guan et al., 2021), text adventure games (Hausknecht et al., 2020). Taking story generation as an example, it requires the model to generate a reasonable story for a given prompt or a given leading context.

In recent years, pre-trained language models (Lewis et al., 2020; Radford et al., 2019) have demonstrated their scalability to large-capability and datasets, becoming a de-facto standard for text generation tasks. These state-of-the-art models already closely resemble humans in the generation of short sentences (Pu et al., 2022). However, as table 1 shows, even with a pre-trained language model, it is still difficult to generate a coherent long text, which indicates that generating a coherent and logical long text is a challenging task. It is observed that pre-trained language models are capable of generating related keywords and achieving good intra-sentence coherence, but still struggle to generate coherent long texts, suffering from generating repetitive plots (Xi et al., 2021), unrelated events, or conflicting logic (Holtzman et al., 2019), e.g. "*for her boyfriend's birthday*" but "*wasted her birthday*" in table 1. We conjecture that above mentioned issues are mainly caused by two reasons. Firstly, current training methods mainly rely on maximum likelihood estimation which is computed from token-level probability prediction. It hinders the model from understanding and planning the generation in the entire text perspective. Secondly, the role of negative samples has been largely under-explored, especially hard negative samples. Thus, the noised generated texts with errors are out-of-distribution for the model.

Input (Leading Sentence):
FEMALE baked a cake for her boyfriend's birthday.
Output 1:
She put the cake in the oven. When the cake was done, **she frosted it. Then she frosted it.** She forgot to put sugar in it.
Output 2:
She spent the morning preparing the cake and putting it in the oven. She left the oven on too long. When she came back the cake was ruined. FEMALE was very sad she 'd **wasted her birthday**.

Table 1: Some stories written by text generation models (The name is replaced with "FEMALE".). The generated stories suffer from incoherence issues (in **bold**), i.e. repeating "*frosted it*", "*for her boyfriend 's birthday*" but "*wasted her birthday*".

To alleviate the above issues, in this paper, we

† Equal Contribution. * Corresponding Author.

Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM), pages 445 - 455

propose **CSP**, a **C**ontrastive **S**oft **P**rompt based text generation model. It learns long text representations in a hidden space, and the model can plan long text generation and distinguish coherent long text and incoherent texts in this hidden space. To this end, we design losses to jointly learn long text representations as well as the ability to plan text generation and distinguish coherent and incoherent texts from the hidden space, specifically, (1) contrastive loss for distinguishing positive samples which are human written texts and negative samples which are generated by applying perturbations to positive ones; (2) contrastive loss for distinguishing different texts; (3) the generation loss for surface realization in text taking representations in hidden space as the soft prompt (Lester et al., 2021). These losses are designed to help the model to learn text representations useful for planning coherent long-text generation. In addition, by taking the generated representations as the soft prompt for text generation, we adopt a language model to learn text representations, and the generated representations serve as extra information for the language model to condition on (Lester et al., 2021). **Our contributions are twofold**. **First**, we propose a novel generation model named CSP for improving the coherence of long text generation. CSP learns high-level representations for long text in hidden space, and jointly learns to plan text generation and distinguish between coherent and incoherent texts utilizing the high-level representations. **Secondly**, we conduct extensive experiments on two-story generation tasks. Experimental results demonstrate that our method can generate more coherent stories than the state-of-the-art model.

2 Method

Our task aims at generating a multi-sentence text $Y = (y_1, y_2, ..., y_m)$, given a text input $X = (x_1, x_2, ..., x_n)$. Figure 1 shows the structure of our proposed model. Our proposed model consists of three parts, prompt generator, prompt posterior generator, and text generator. The prompt generator aims at generating a soft prompt that represents the hidden representation of the text to be generated. The prompt posterior generator is used to help train the prompt generator, which provides the hidden representations of positive and negative samples so that the prompt generator is trained to generate soft prompts close to positive samples and away from negative ones. The text generator generates the text using the soft prompt and the input.

2.1 Prompt Generator

Soft prompt serves as extra information for the language model to condition on (Lester et al., 2021). The prompt generator aims at generating a soft prompt which is used by the text generator for generation. To improve the coherence of long text generation, the model learns to generate soft prompts close to the representations of coherent texts and away from incoherent texts in hidden space, which is introduced in the following subsections.

The prompt generator takes the concatenation of X and a special token sequence $P = ([P_1], [P_2], ..., [P_k])$ as the input, and outputs a soft prompt $S = s_1, s_2, ..., s_k$, where P represents the placeholders for generating the soft prompt, $S \in R^{k \times d}$ is the soft prompt, k is the length of soft prompt, d is the hidden dimension. The prompt generator can be any sequence-to-sequence model, e.g. a Transformer model (Vaswani et al., 2017). In our experiment, we use GPT-2 (Radford et al., 2019) as the backbone for the prompt generator. Specifically, let X' be the concatenation of X and P, and we denote the GPT-2 as a function f. We take the hidden states of GPT-2 as the output, which is a sequence with length $n + k$, we use the subsequence of the output sequence corresponding to P as the soft prompt, i.e.

$$\begin{aligned} X' &= X \parallel P \\ Y' &= f(X') \\ S &= Y'_{n+1:n+k} \end{aligned} \tag{1}$$

where $\parallel$ is the concatenation operation, and $Y'_{n+1:n+k}$ represents the sub-sequence from $n+1$ to $n+k$ elements, i.e. the hidden states of f corresponding to P.

2.2 Posterior Prompt Generator

In analogy to the prompt generator, the posterior prompt generator is also a seq2seq model. Its input is the concatenation of three parts, (1) the input X; (2) the output Y or a negative sample Y_-, and we will introduce the construction of negative examples in the following subsection; (3) a special token sequence $Q = ([Q_1], [Q_2], ..., [Q_k])$. The posterior prompt is computed from

$$\begin{aligned} X''_+ &= X \parallel Y \parallel Q \\ Y''_+ &= f(X''_+) \\ S_+ &= Y''_{n+m+1:n+m+k} \end{aligned} \tag{2}$$

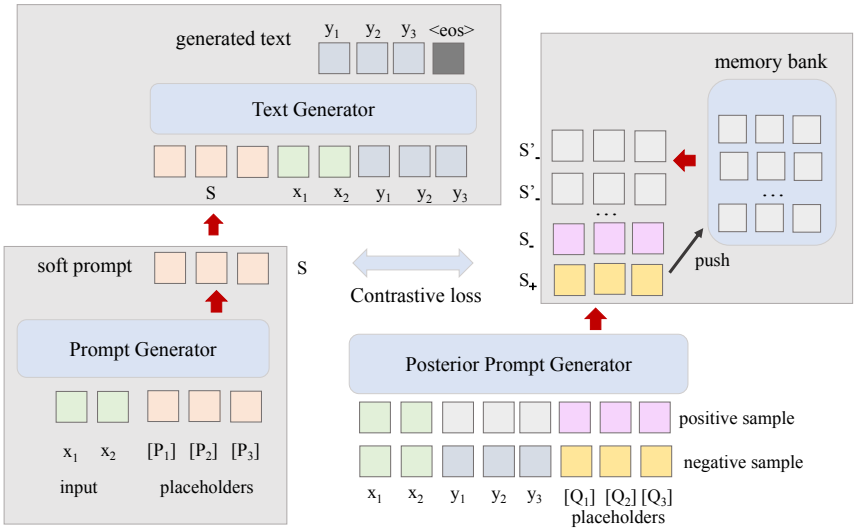

Figure 1: The structure of our proposed model. Prompt generator aims at generating a soft prompt closed to coherent texts and away from incoherent texts in the hidden space. Posterior prompt generator encodes positive and negative samples into representations in the hidden space, and helps to train prompt generator. Memory bank stores the text representations produced in previous training steps, and also help to train prompt generator. Note that the posterior prompt generator and memory bank are only used during training and can be dropped during inference. The prompt generator and posterior prompt generator share most of parameters except for embeddings of $[P_1]$, $[P_2]$, ...,$[P_k]$, and $[Q_1]$, $[Q_2]$, ...,$[Q_k]$. Text generator aims at generating text given the soft prompt and the input in auto-regressive manner. The special token <eos> represents the end of the text.

where $Y''_{n+m+1:n+m+k}$ is the hidden states of f corresponding to Q, S_+ represents the posterior soft prompt for a sample in the training dataset. The posterior soft prompt for a negative sample S_- can be generated similarly. Both S_+ and S_- will be used to train the prompt generator, which will be introduced in the subsection of contrastive loss.

2.3 Negative Sample Generation

We construct negative samples to cover unfavored incoherent texts. To enable the model to learn to distinguish both intra-sentence and inter-sentence errors, we construct negative samples from both the N-gram level and sentence level. To construct negative samples, we randomly apply the following perturbations to texts:

Repetition. In recent studies, it is observed that many NLG models produce repeated text contents, particularly with maximum-likelihood-based decoding strategies (Holtzman et al., 2019). Thus we generate negative samples by randomly repeating N-grams or sentences, aiming at telling the model to not repeat content when generating texts.

Deletion. We randomly delete sentences or N-grams from the original text, to let the model distinguish a coherent text and an incoherent text with missing content.

Insertion. We randomly insert sentences or N-grams into the random positions of the original text. In this way, the model learns to distinguish negative samples with extra text contents.

Substitution. We randomly substitute sentences or N-grams with a random sentence or N-gram in the training corpus.

Reorder. We randomly shuffle the order of the sentences in a text. This will produce stories with wrong temporal order or wrong causality.

2.4 Contrastive Loss

Given the soft prompt S and positive posterior soft prompt S_+ and negative posterior soft prompt S_-, we design the contrastive loss to let S be closed to the positive posterior soft prompt and away from negative ones, to avoid generating incoherent texts. We also adopt extra negative samples from a memory bank inspired by previous contrastive learning methods (Wu et al., 2018; He et al., 2020). We sample extra negative samples S'_- from a memory bank B. After each training step, S and S_+ are stored in the memory bank B. We map S, S_+, S_-, and S'_- to a hidden space, e.g. for S, we have

$$v = W_{proj} pool(S) \quad (3)$$

Dataset	#Input tokens	#Output tokens	#Train sample	#Val sample	#Test sample
ROCStories	14.5	56.3	88344	4908	4909
WritingPrompts	30.0	185.7	26758	2000	2000

Table 2: Statistics of the datasets.

where pool [2] represents a function which average the matrix with dimension $k \times d$ into a vector with dimension d. We use a W_{proj} to map the origin $pool(S)$ into a hidden space. Similarly, we map S_+, S_-, S'_- to v_+, v_-, v'_- respectively.

In this paper, infoNCE (Oord et al., 2018) is considered as the contrastive loss function:

$$L_c = -\log \frac{exp(\frac{v \cdot v_+}{\tau})}{\sum_{v_-} exp(\frac{v \cdot v_-}{\tau}) + \sum_{v'_-} exp(\frac{v \cdot v'_-}{\tau})} \tag{4}$$

where τ is a temperature hyperparameter.

2.5 Text Generator

Given a soft prompt S and the input X, the text generator aims at generating a text Y. We also use GPT-2 as the backbone of the text generator. Following GPT-2, we learn the text generation in an auto-regressive manner. It is learned from cross-entropy loss, i.e.

$$\begin{aligned} Z &= f(S \parallel X \parallel Y) \\ H &= Z_{k+n+1:k+n+m} \\ P(y_t|y<t, X) &= softmax(H_t W + b) \\ L_{ce} &= -\sum_{t=1}^{m} logP(y_t|y<t, X) \end{aligned} \tag{5}$$

where Z is the hidden states of the text generator, and H is the hidden states corresponding to the output, W and b are trainable parameters.

We also introduce a text reconstruction loss similar to autoencoder, i.e.

$$\begin{aligned} Z^{ae} &= f(S_+ \parallel X \parallel Y) \\ H^{ae} &= Z^{ae}_{k+n+1:k+n+m} \\ P^{ae}(y_t|y<t, X) &= softmax(H^{ae}_t W + b) \\ L_{ae} &= -\sum_{t=1}^{m} logP^{ae}(y_t|y<t, X) \end{aligned} \tag{6}$$

[2]We also tried to flatten operation in our experiments, it has similar performance with pool operation while flatten represents a function which reshapes the matrix with dimension $k \times d$ into a vector.

By optimizing L_{ae}, the model tries to learn a hidden representation for a whole text, and the text content can be reconstructed from this hidden representation.

The loss function for our model combines L_c, L_{ce}, and L_{ae}, i.e.

$$L = \lambda_c L_c + \lambda_{ce} L_{ce} + \lambda_{ae} L_{ae} \tag{7}$$

where λ_c, λ_{ce} and λ_{ae} are hyper-parameters.

3 Experiments

3.1 Dataset

We evaluate our model on two publicly available story generation datasets, named ROCStories (Mostafazadeh et al., 2016) and WritingPrompts (Fan et al., 2018). We use the same preprocessing method as the previous work (Guan et al., 2020, 2021), i.e. all the names are replaced with special placeholders for better generalization. For ROCStories, we use the first sentence as the input and expect the model to generate the remaining content of the story. For WritingPrompts, the input is the writing prompt, and the model is expected to generate a story according to the writing prompt. We use the same filter strategy and validation and test split as the previous work (Guan et al., 2021). Table 2 shows the statistics of these two datasets.

3.2 Baselines

We compare our method with several baselines, including fine-tuning pre-trained language models, the previous state-of-the-art method, and variants of our method.

BART (Lewis et al., 2020): It is fine-tuned on the ROCStories and WritingPrompts datasets based on the publicly available BART model checkpoint.

GPT-2 (Radford et al., 2019): It is fine-tuned on the ROCStories and WritingPrompts datasets based on the pre-trained GPT-2 model.

HINT (Guan et al., 2021): It is the previous state-of-the-art method on ROCStories and WritingPrompts datasets, which continue the pretraining of BART on book corpus with two additional objectives, i.e. inter-sentence semantic similarity and

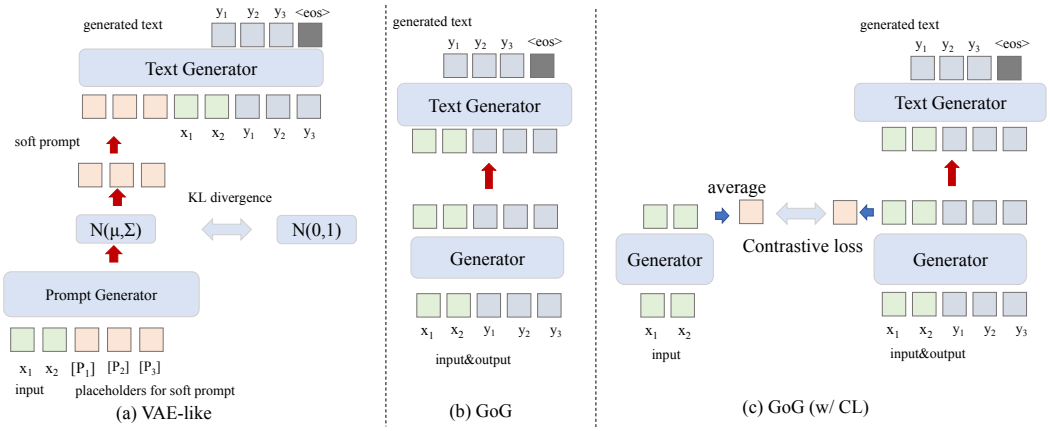

Figure 2: The structures of some baseline methods, including VAE-like, GOG, and GOG (w/ CL).

distinguishing between normal and randomly shuffled sentence orders.

VAE-like: Similar to VAE (Kingma and Welling, 2013), it uses Gaussian distribution as the prior probability distribution for the soft prompt S. Compared to our method, it replaces the contrastive loss with the ELBO loss function. The left of figure 2 shows the structure of VAE-like.

GoG: It stacks two GPT-2 models, and is fine-tuned on ROCStories and WritingPrompts datasets. The middle of figure 2 shows the structure of GoG.

GoG (w/ CL): It also stacks two GPT-2 models, and is fine-tuned on the downstream datasets with language modeling objective and contrastive loss. Compared to our method, the hidden representations used to compute contrastive loss are the linear mapping of the average of hidden representations. The right of figure 2 shows the structure of GoG (w/ CL).

CSP (w/o CL): It is a variant of our model, which removes the contrastive loss L_c.

CSP (w/o AE): It is a variant of our model, which removes the reconstruction loss L_{ae}.

CSP (w/ PT): It has the same structure and loss function as our method, but based on model parameters which are continue pre-trained on book corpus. We use the same loss as our model during continue pretraining.

3.3 Automatic Evaluation

Evaluation Metrics We adopt several commonly used metrics to evaluate the performance, including (1) **UNION**: It is a learnable metric proposed by Guan and Huang (2020), which adopts a classifier trained from human-written texts and negative samples constructed by applying perturbations to human-written texts. The UNION score is the average classifier score of texts and measures the coherence and context-relatedness of the generated texts. (2) **Orderness**: It is also a learnable metric. It relies on a classifier trained to distinguish human-written texts and randomly shuffled sentences. This metric reflects the degree to which texts are in reasonable sentence order. Both UNION and Orderness are trained on the training sets of ROCStories or WritingPrompts. (3) **Perplexity (PPL)**: It measures how well a probability model predicts the ground-truth samples. (4) **BLEU (B-n)**: It reflects the n-gram overlap ratio between generated texts and ground-truth texts (Papineni et al., 2002). (5) **Lexical Repetition (LR-n)**: It is the percentage of generated texts which repeat 4-gram at least n times (Shao et al., 2019a). (6) **Distinct-4 (D-4)**: It is the ratio of distinct 4-grams in all 4 grams in the texts.

Results on ROCStories. As shown in the table 3, our proposed method outperforms all the baselines on 6 out of 7 metrics on the ROCStories dataset. As for the structure and coherence of generated texts, our model achieves the best UNION and Orderness metrics. Our model has higher PPL compared to GPT-2, and it is because the contrastive loss L_c provides a regularization, and the model may allocate more probability to diverse texts. This conjecture can be supported by diversity-related metrics. Our model will generate more diverse texts and thus has higher distinct and lower lexical repetition. Although our model improves diversity, it still has better BLEU than baselines, indicating that our model is inclined to generate diverse but reasonable n-grams. With continued pretraining on book corpus, our model improves UNION and distinct metrics further and decreases repetition metrics. As both HINT and CSP (w/PT)

Models	UNION↑	orderness↑	PPL↓	B-1↑	B-2↑	LR-2↓	D-4↑
BART	0.684	0.906	10.72	0.293	0.131	0.312	0.651
GPT-2	0.767	0.935	**8.72**	0.315	0.143	0.239	0.695
HINT	0.772	0.920	9.20	0.331	0.154	0.263	0.678
VAE-like	0.805	0.938	9.16	0.316	0.143	0.231	0.693
GoG	0.828	0.946	9.32	0.318	0.146	0.197	0.710
GoG (w/ CL)	0.838	0.945	9.30	0.322	0.148	0.200	0.706
CSP	0.853	**0.952**	12.18	0.332	**0.158**	0.186	0.747
CSP (w/o CL)	0.837	0.945	9.31	0.315	0.144	0.202	0.714
CSP (w/o AE)	0.848	0.949	12.64	0.327	0.149	0.172	0.757
CSP (w/ PT)	**0.863**	0.950	11.72	**0.333**	0.151	**0.160**	**0.772**

Table 3: Automatic evaluation results on the ROCStories dataset.

Models	UNION↑	orderness↑	PPL↓	B-1↑	B-2↑	LR-5↓	D-4↑
BART	0.302	0.909	32.02	0.219	0.079	0.381	0.409
GPT-2	0.325	0.769	**27.11**	0.209	0.075	0.558	0.418
HINT	0.353	0.909	30.71	0.221	0.083	0.323	0.445
VAE-like	0.387	0.935	27.98	0.242	0.090	0.384	0.463
GoG	0.394	0.926	28.71	0.245	0.091	0.326	0.483
GoG (w/ CL)	0.393	0.924	28.99	0.244	0.092	0.361	0.460
CSP	0.520	0.936	32.01	0.255	0.097	0.230	0.555
CSP (w/o CL)	0.382	0.929	27.76	0.245	0.091	0.385	0.454
CSP (w/o AE)	0.449	0.932	33.26	0.246	0.092	0.274	0.537
CSP (w/ PT)	**0.711**	**0.940**	32.08	**0.278**	**0.102**	**0.154**	**0.682**

Table 4: Automatic evaluation results on the WritingPrompts dataset.

use book corpus, the comparison of these two models further shows the effectiveness of our model.

Results on WritingPrompts. Table 4 shows the results on the WritingPrompts dataset. Texts in the WitingPrompts dataset are longer than texts in ROCStories, and most baseline methods tend to repeat texts, with low distinct and high repetition metrics. However, our model can alleviate this issue evidenced by lexical repetition and distinct metrics. Our model can increase the distinct metric up to nearly 10 percents. In addition, the improvement of UNION is much more significant than on ROCStories, with about 50% relative improvement compared to HINT, although HINT also adopts specially designed loss for improving coherence. Our model also achieves 1 point BLEU improvement compared to best baseline method. With continue pretraining on book corpus, our model can further improve coherence and diversity, specifically, improve UNION from 0.520 to 0.711 and B-1 from 0.255 to 0.278, and also significantly improve distinct and decreases repetition metrics.

The comparison of GOG and CSP shows that our model structure for computing text representation is more effective than simply averaging hidden states. Furthermore, the experimental results of CSP (w/o CL) and CSP (w/o AE) show that the contrastive loss and reconstruction loss can improve the performance of our model.

3.4 Manual Evaluation

For manual evaluation, we conduct a pair-wise comparison on two aspects, namely *fluency* and *coherence*, following recent studies (Guan et al., 2021; Xu et al., 2020). The metric *fluency* measures linguistic quality while *coherence* focuses on logicality, e.g. causality and temporal relationship. We randomly sample 100 generated texts from the test set of ROCStories and invite three annotators (they are all volunteers) to give a preference concerning fluency and coherence respectively (win, lose or tie). As table 5 shows, our model outperforms all the baselines on both *fluency* and *coherence* aspects. We use Fleiss's kappa (Fleiss, 1971) to measure the inter-annotator agreements, most of the results are moderate ($0.4 \leq \kappa \leq 0.6$) or substantial ($0.6 \leq \kappa \leq 0.8$). However, our model still generates some incoherent texts and is judged to

Models	Fluency				Coherence			
	Win	Lose	Tie	κ	Win	Lose	Tie	κ
CSP vs. GPT-2	27	3	70	0.75	58	5	37	0.76
CSP vs. HINT	22	7	71	0.72	57	4	39	0.61
CSP vs. VAE-like	15	4	81	0.69	62	10	28	0.59
CSP vs. GOG	20	5	75	0.66	57	9	34	0.55
CSP vs. GOG (w/ CL)	35	3	62	0.84	52	6	42	0.79
CSP vs. CSP (w/o CL)	14	2	84	0.64	62	6	32	0.60

Table 5: Manual evaluation results on the ROCStories dataset. κ is the Fleiss' kappa (Fleiss, 1971), measuring the inter-annotator agreement (most of them are moderate or substantial).

Noise type	UNION↑	orderness↑	PPL↓	B-1↑	B-2↑	LR-2↓	D-4↑
N-gram	0.853	0.951	**11.40**	0.324	0.149	0.185	0.747
Sentence	**0.869**	0.951	11.43	0.320	0.147	**0.166**	**0.749**
N-gram+Sentence	0.853	**0.952**	12.18	**0.332**	**0.158**	0.186	0.747

Table 6: Effectiveness of different types of noise on the ROCStories dataset.

"lose" compared to baselines.

3.5 Effectiveness of different negative samples

Table 6 shows the effectiveness of different types of noise on the ROCStories dataset. Using negative samples constructed from sentence noise is more effective than N-gram noise in lexical repetition and UNION metrics, and achieves similar PPL, BLEU, distinct and Orderness metrics. Combining N-gram and sentence noise will achieve about 1 point BLEU improvement.

3.6 Influence of different memory bank sizes

Table 7 shows the performances of our models with different memory bank sizes. When the memory bank size is 0, the model only needs to distinguish between human-written texts and negative samples constructed by applying N-gram noise and sentence noise, which has better PPL and distinct metrics. We also observed that without a memory bank, the soft prompts mainly lie in two areas, one for positive samples and the other for negative samples. By adding a memory bank, the model learns better text representation that could encode the differences between different texts, and achieves better UNION and BLEU metrics. However, PPL increases when we use a memory bank, and we conjecture that it is mainly because the model allocates more probability to some other reasonable stories.

3.7 Case Study

We present a case in table 8 to demonstrate that CSP can generate texts with better coherence than the previous SOTA model HINT.

4 Related Work

Long Text Generation Many recent long text generation studies try to tackle the incoherence problem by designing model structures, training methods and incorporating extra knowledge. Li et al. (2015) propose a hierarchical RNN model to learn the sentence-level representation. Fan et al. (2018) propose a hierarchical CNN model, and they also adopt a gated multi-scale self-attention mechanism to capture long-range context information. These two methods focus on modeling both word-level and sentence-level representations aiming at capturing long-range dependency. Another line of work adopts plan-then-generate methods (Yao et al., 2019; Shao et al., 2019b; Tan et al., 2021; Goldfarb-Tarrant et al., 2020). They first generate a high-level plan, and then generate the whole text according to the plan. The main problem of these works is that the models are biased to the plans extracted from human-written texts during training and lack of exposure to generated plans (Tan et al., 2021). There are also some works that try to incorporate knowledge base into text generation models (Guan et al., 2020; Xu et al., 2020) to improve the ability to generate commonsense stories. However, these methods mainly focus on commonsense stories, and may not be effective for generating other types of stories. Guan et al. (2021) propose the HINT model, which is the previous state-of-the-art method. HINT adopts two extra losses for training, i.e. inter-sentence seman-

Memory bank size	UNION↑	orderness↑	PPL↓	B-1↑	B-2↑	LR-2↓	D-4↑
0	0.832	0.948	**9.21**	0.322	0.147	**0.164**	**0.760**
600	0.851	0.948	11.15	0.319	0.146	0.183	0.735
65536	**0.853**	**0.952**	12.18	**0.332**	**0.158**	0.186	0.747

Table 7: Influence of different memory bank sizes on the ROCStories dataset.

Input (Leading Sentence):
MALE had a roommate.
Output of HINT:
MALE was **a hard worker and a great student**. MALE knew MALE was very **smart and smart**. MALE challenged MALE to **a game of basketball**. MALE beat MALE by a landslide.
Output of CSP:
he was very messy. MALE's roommate's mom said he needed to be more organized. MALE's roommate agreed to do so. MALE's roommate was much more organized.

Table 8: Generated stories by the previous SOTA model HINT and our model CSP (Names are replaced with "MALE"). In this case, CSP generates a more coherent text than HINT (incoherent issues are in **bold**).

tic similarity and discrimination between normal and random shuffled sentence orders. Compared to HINT, our method focuses on learning a better representation in a hidden space for better planning long text generation.

Contrastive Learning In recent years, contrastive learning has made great advances in computer vision (Tian et al., 2020; He et al., 2020; Chen et al., 2020; Misra and Maaten, 2020), natural language processing tasks (Wang et al., 2021a; Pan et al., 2021; Zhang et al., 2021a; Gao et al., 2021; Kim et al., 2021), as well as multi-modal tasks (Radford et al., 2021; Wang et al., 2021b). In the NLP domain, contrastive learning is adopted for sentence representation (Zhang et al., 2021a; Gao et al., 2021; Kim et al., 2021). Pan et al. (2021) use contrastive learning to learn a universal cross-language representation for better multilingual translation performance. Inspired by these works, we adopt contrastive learning to learn a better text representation, aiming at helping the model to plan long text generation from a high level and avoid generating incoherent texts.

Prompt Tuning Recently, some works have shown the effectiveness of prompt tuning in zero-shot and few-shot tasks (Brown et al., 2020; Gao et al., 2020). By designing or automatically searching templates and demonstrations, prompt tuning provides effective techniques for fine-tuning language models using only a few examples. Furthermore, the soft prompt is proposed as a parameter-efficient finetuning method (Li and Liang, 2021; Liu et al., 2021; Lester et al., 2021; Zhang et al., 2021b), i.e. the parameters of the pretrained language model remain fixed, and we add only a few trainable parameters as a prefix to the input sequence. Our work is inspired by recent soft prompt works, however, these works mainly focus on parameter-efficient finetuning, while our work aims at improving the coherence of long text generation. We jointly learn high level text representations in hidden space and take the representation as the soft prompt for better long text generation.

Learning from Negative Examples Welleck et al. (2019) have tried to avoid the model generating repetitive, dull text by unlikelihood training. Li et al. (2019) use unlikelihood training to generate text consistent with persona information. Unlikelihood loss is computed from tokens, and our method mainly focuses on high level representations in hidden space. Another line of studies on employing negative examples is adversarial learning (Yu et al., 2017; Li et al., 2017), which plays a minimax game between a generative model and a discriminative model to generate texts that can not be distinguished from human-written texts. We anticipate that unlikelihood training and adversarial training are largely complementary to our method.

5 Conclusion

In this paper, we propose a contrastive soft prompt method for improving the coherence of long text generation. It learns long text representations in the hidden space for better planning long text generation. To this end, it jointly learns to generate a text representation close to representations of coherent texts and away from incoherent ones and generates long text taking this representation as the soft prompt. We conduct experiments on two public story generation datasets, and experimental results show that our method can generate more coherent stories than the state-of-the-art model.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Seraphina Goldfarb-Tarrant, Tuhin Chakrabarty, Ralph Weischedel, and Nanyun Peng. 2020. Content planning for neural story generation with aristotelian rescoring. *arXiv preprint arXiv:2009.09870*.

Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Transactions of the Association for Computational Linguistics*, 8:93–108.

Jian Guan and Minlie Huang. 2020. Union: An unreferenced metric for evaluating open-ended story generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9157–9166.

Jian Guan, Xiaoxi Mao, Changjie Fan, Zitao Liu, Wenbiao Ding, and Minlie Huang. 2021. Long text generation by modeling sentence-level and discourse-level coherence. *In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7903–7910.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2021. Self-guided contrastive learning for bert sentence representations. *arXiv preprint arXiv:2106.07345*.

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

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. 2017. Data-driven news generation for automated journalism. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 188–197.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. 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*, pages 7871–7880.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y-Lan Boureau, Kyunghyun Cho, and Jason Weston. 2019. Don't say that! making inconsistent dialogue unlikely with unlikelihood training. *arXiv preprint arXiv:1911.03860*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Ishan Misra and Laurens van der Maaten. 2020. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of

commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Xiao Pan, Mingxuan Wang, Liwei Wu, and Lei Li. 2021. Contrastive learning for many-to-many multilingual neural machine translation. *arXiv preprint arXiv:2105.09501*.

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.

Jiashu Pu, Ziyi Huang, Yadong Xi, Guandan Chen, Weijie Chen, and Rongsheng Zhang. 2022. Unraveling the mystery of artifacts in machine generated text. In *Proceedings of the Language Resources and Evaluation Conference*, pages 6889–6898, Marseille, France. European Language Resources Association.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*.

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

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019a. Long and diverse text generation with planning-based hierarchical variational model. *arXiv preprint arXiv:1908.06605*.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019b. Long and diverse text generation with planning-based hierarchical variational model. *arXiv preprint arXiv:1908.06605*.

Bowen Tan, Zichao Yang, Maruan Al-Shedivat, Eric Xing, and Zhiting Hu. 2021. Progressive generation of long text with pretrained language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4313–4324.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer.

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.

Dong Wang, Ning Ding, Piji Li, and Hai-Tao Zheng. 2021a. Cline: Contrastive learning with semantic negative examples for natural language understanding. *arXiv preprint arXiv:2107.00440*.

Mengmeng Wang, Jiazheng Xing, and Yong Liu. 2021b. Actionclip: A new paradigm for video action recognition. *arXiv preprint arXiv:2109.08472*.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742.

Yadong Xi, Jiashu Pu, and Xiaoxi Mao. 2021. Taming repetition in dialogue generation. *arXiv preprint arXiv:2112.08657*.

Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. 2020. Megatron-cntrl: Controllable story generation with external knowledge using large-scale language models. *arXiv preprint arXiv:2010.00840*.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7378–7385.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Dejiao Zhang, Shang-Wen Li, Wei Xiao, Henghui Zhu, Ramesh Nallapati, Andrew O Arnold, and Bing Xiang. 2021a. Pairwise supervised contrastive learning of sentence representations. *arXiv preprint arXiv:2109.05424*.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2021b. Differentiable prompt makes pre-trained language models better few-shot learners. *arXiv preprint arXiv:2108.13161*.

A Appendix

A.1 Details of experiments

The model structure of the prompt generator, posterior prompt generator, and text generator are the same with GPT-2, and the parameter weights are initialized with a pre-trained GPT-2 checkpoint. The soft prompt length is 20 in our experiment.

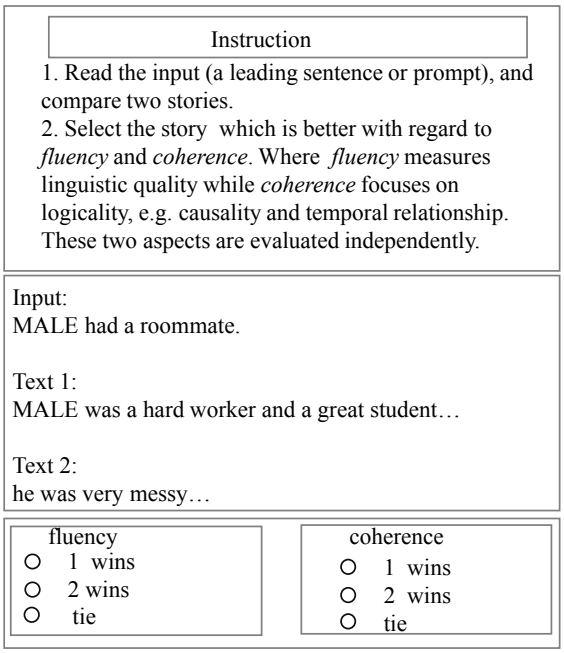

Figure 3: Manual annotation instruction.

The memory bank size is 65536. We set $\lambda_c = 1.0$, $\lambda_{ce} = 0.1$ and $\lambda_{ae} = 0.1$. We finetune the model on ROCStories and WritingPrompts for 22000 steps respectively. We use adam optimizer, and the learning rate is set to $5\mathrm{e}{-5}$, no weight decay, and the batch size is 16. During generation, we use nucleus sampling with p=0.9, and the softmax temperature is 0.7. Our model contains 234 million parameters. We run our experiments on one GeForce RTX 3090, and it takes about 34 and 39 hours for training models on ROCStories and WritingPrompts datasets respectively.

A.2 Annotation Instruction

Figure 3 shows the annotation instruction. we conduct pair-wise comparison on two aspects, namely *fluency* and *coherence*. The metric *fluency* measures linguistic quality while *coherence* focuses on logicality, e.g. causality and temporal relationship. We randomly sample 100 generated texts from the test set of ROCStories and invite three annotators (they are all volunteers) to give a preference about fluency and coherence respectively (win, lose or tie). The comparison pair of texts are presented in random order.