Self-Supervised Sentence Polishing by Adding Engaging Modifiers

Zhexin Zhang[1], **Jian Guan**[1], **Xin Cui**[2], **Yu Ran**[2], **Bo Liu**[2] **and Minlie Huang**[1*]
[1]The CoAI group, DCST; [1]Institute for Artificial Intelligence; [1]State Key Lab of Intelligent Technology and Systems;
[1]Beijing National Research Center for Information Science and Technology; [1]Tsinghua University, Beijing 100084, China.
[2]Platform and Content Group, Tencent Technology Co., Ltd.
{zx-zhang22,j-guan19}@mails.tsinghua.edu.cn, {kimicui,ryanran,ustbliu}@tencent.com
aihuang@tsinghua.edu.cn

Abstract

Teachers often guide students to improve their essays by adding engaging modifiers to polish the sentences. In this work, we present the first study on automatic sentence polishing by adding modifiers. Since there is no available dataset for the new task, we first automatically construct a large number of parallel data by removing modifiers in the engaging sentences collected from public resources. Then we fine-tune LongLM (Guan et al., 2022) to reconstruct the original sentences from the corrupted ones. Considering that much overlap between inputs and outputs may bias the model to completely copy the inputs, we split each source sentence into sub-sentences and only require the model to generate the modified sub-sentences. Furthermore, we design a retrieval augmentation algorithm to prompt the model to add suitable modifiers. Automatic and manual evaluation on the auto-constructed test set and real human texts show that our model can generate more engaging sentences with suitable modifiers than strong baselines while keeping fluency. We deploy the model at http://coai.cs.tsinghua.edu.cn/static/polishSent/. A demo video is available at https://youtu.be/Y6gFHOgSv8Y.

1 Introduction

Teachers' guidance is necessary for students to improve their essays in primary and secondary writing education. For example, teachers can point out potential logical errors and incoherence issues, and polish sentences to improve the engagingness of the essays. A typical way to polish sentences is to add engaging modifiers (e.g., from "*I ate a pear*" to "*I ate a big pear enjoyably*"), which usually are adjectives or adverbs that enhance the meaning of a sentence (Witte and Faigley, 1981). Since an essay usually contains tens of sentences, it is a heavy burden for teachers to polish each one. To reduce teachers' workload and enable students to improve their essays independently, we present a new study on *automatic sentence polishing*, which requires polishing a given sentence given its context. The goal of polishing a sentence is to make the sentence more expressive, attractive and engaging. We only consider inserting modifiers for polishing in this work, and leave other types of polishing to future work (e.g., replacing words or rephrasing the sentence). The challenges of the new task mainly manifest in the following two folds: **(1)** finding the words that can be modified; and **(2)** deciding suitable modifiers for those words.

Original Sentences	**Corrupted Sentences**
先是轻盈的雨滴轻轻地滴落，发出悦耳的"叮咚"声，就像是乐曲的前奏。(First, light raindrops drip softly, emitting a pleasant "ding-dong" sound, like a prelude to the music.)	先是雨滴轻轻地滴落，发出"叮咚"声，就像是乐曲的前奏。(First, raindrops drip softly, emitting a "ding-dong" sound, like a prelude to the music.)
情不自禁地哼起那首《乡间的小路》，抛开了一切繁重的心事，直到夜幕降临。(I can't help but hum the song *Country Road*, leaving aside all heavy thoughts until nightfall.)	哼起那首《乡间的小路》，抛开了一切繁重的心事，直到夜幕降临。(I hum the song *Country Road*, leaving aside all heavy thoughts until nightfall.)

Table 1: Examples of automatically constructed data. After collecting original sentences, we corrupt them to construct less engaging sentences by removing the modifiers that are in a vocabulary of engaging words (marked in red).

Considering that there are no available parallel data for this new task, we propose a self-supervised learning approach using automatically constructed training data. We firstly collect a large number of engaging sentences from public books and student essays in Chinese, and then corrupt the sentences to construct less engaging ones by removing the modifiers in them, as exemplified in Table 1. We

[*]Corresponding author

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 3: System Demonstrations, pages 499–507
July 10-12, 2023 ©2023 Association for Computational Linguistics

learn a generation model for sentence polishing by training it to reconstruct the original sentences from the corrupted ones. To alleviate the model's tendency to completely copy inputs as generation outputs, we train the model to generate only the changed sub-sentences split by commas (e.g., "I can't help but hum the song Country Road" in the second example). Furthermore, we propose a novel retrieval augmentation algorithm to improve the correctness of added modifiers by retrieving suitable pairs of modifiers and modified words from the training set as additional inputs.

Automatic and manual evaluation on the auto-constructed test set and real human texts show that our model can generate more engaging sentences with suitable modifiers and comparable fluency than strong baselines. Furthermore, we build a website to enable real-time interaction with our deployed model, where a user can upload a Chinese sentence with its context and get the retrieval result along with the polished sentence.

2 Related Work

2.1 Constrained Text Generation

Automatic sentence polishing can be regarded as a kind of constrained text generation task (Garbacea and Mei, 2022), which requires generating coherent text that meets given constraints. Typical constrained generation tasks span from machine translation (Yang et al., 2020), summarization (Paulus et al., 2018), sentence generation from input concepts (Lin et al., 2020a), story generation from input phrases (Rashkin et al., 2020) or events (Ammanabrolu et al., 2020). Previous studies usually adopt the encoder-decoder framework (Sutskever et al., 2014) equipped with the attention mechanism (Bahdanau et al., 2015) to deal with constrained generation tasks. Recently, large-scale pretraining models based on the Transformer model (Vaswani et al., 2017) such as BART (Lewis et al., 2020) and LongLM (Guan et al., 2022) achieve more surprising performance (Lin et al., 2020b) although they are still far from humans (Lin et al., 2020a).

2.2 Text-Editing Models

There is significant overlap between inputs and outputs in many constrained generation tasks such as grammatical error correction (Omelianchuk et al., 2020) and sentence polishing in this work. When applying the vanilla encoder-decoder framework

	Train	**Val**	**Test$_{\text{Auto}}$**	**Test$_{\text{Real}}$**
# Examples	143,185	17,898	1000	1000
Avg. M **Len**	29.33	29.41	29.48	28.71
Avg. S **Len**	37.89	37.76	38.31	41.92
Avg. N **Len**	29.22	29.38	27.89	27.37
Avg. T **Len**	42.40	42.22	42.79	N/A

Table 2: Statistics of the dataset. *Len* is the abbreviation of *Length*. **Train**, **Val** and **Test**$_{\text{Auto}}$ are the auto-constructed training, validation and test sets, respectively. **Test**$_{\text{Real}}$ is the test set from real human-written sentences. We compute the length by counting the number of Chinese characters.

to such tasks, the models tend to directly copy the input without modification and it seems wasteful to generate the whole output text from scratch (Malmi et al., 2019). Text-editing models are proposed to address this issue, which usually conduct token-wise prediction for how to edit the token. LaserTagger (Malmi et al., 2019) presented three editing types including retaining the token, deleting the token, and inserting tokens before the token using a fixed phrase vocabulary obtained from the training set. Felix (Mallinson et al., 2020) further adopted a pointer network to learn to reorder the input tokens, and utilized a pretrained masked language model to predict inserted tokens. Seq2Edits (Stahlberg and Kumar, 2020) proposed a span-level editing type that allowed to generate a span as insertion or replacement. Lewis (Reid and Zhong, 2021) used a two-step editor which first predicted coarse editing types and then filled in replacements and insertions. EdiT5 (Mallinson et al., 2022) was a semi-auto-regressive approach with non-auto-regressive text labeling and auto-regressive decoding. It first decided the subset of input tokens to be retained using an encoder, then reordered the tokens with a pointer module, and finally infilled the missing tokens using a decoder. In this work, we proposed a simple but effective approach to address the copy issue by only decoding the changed sub-sentences.

3 Dataset Construction

We formulate our task as follows: given three consecutive sentences M, S, N, the model should output a polished sentence T that is more engaging than S while maintaining the original meaning of S and the coherence along with M and N. Since there are no available data for this task, we construct a new dataset through automatic annotation.

Firstly, we use an off-the-shelf OCR tool to col-

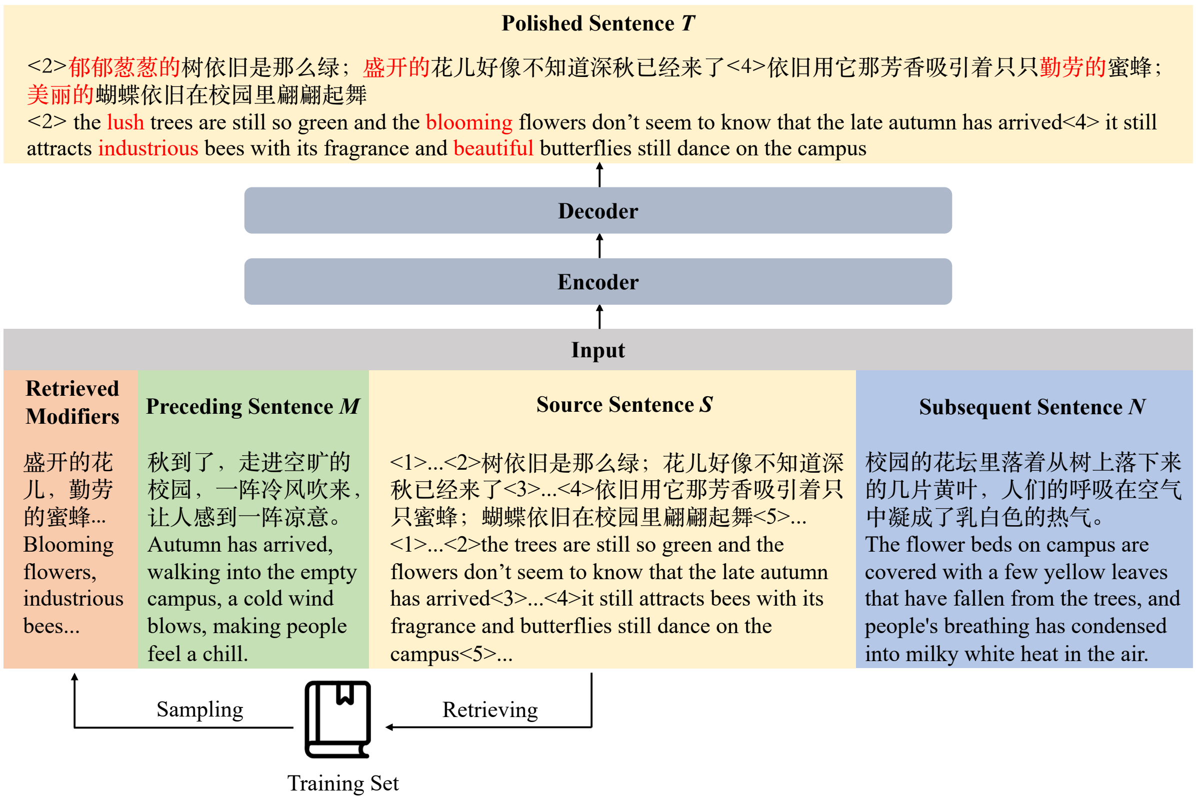

Figure 1: Model overview. We split the source sentence S into sub-sentences by commas and only generate the modified sub-sentences. We also retrieve relevant modifiers from the training set, which are taken as input. The modifiers added by the model are marked in red.

lect about 50k engaging sentences and a vocabulary of 61k engaging words from several books[1], and collect 26k high-quality Chinese student essays from public resources[2] that describe the scenery and thus potentially contain lots of engaging sentences. Then we take every triple of adjacent three sentences in these texts collected from books and websites as M, T and N[3], respectively. We obtain S by removing modifiers in T. Specifically, we adopt a public Chinese NLP toolkit LTP [4] to identify attribute and adverbial words in T, and regard those words included in the vocabulary of engaging words as the modifiers that can be removed. Note that we also remove the structural particle words including "的" and "地" following the removed modifiers to ensure the fluency of the final sentence S. If there is no removed modifier in T, we will discard the example.

Table 2 shows the statistics of our dataset. Considering that the auto-constructed inputs may be different from real texts, we construct an additional test set, i.e., $\text{Test}_{\text{Real}}$, where the inputs are original sentences whose modifiers are not removed. We set the size of both test sets to 1000 to balance the estimation error and the inference time.

4 Methodology

An overview of our model is shown in Figure 1. We build our model on $\text{LongLM}_{\text{Large}}$ (Guan et al., 2022) for the sentence polishing task, which is an encoder-decoder model pretrained on Chinese novels with 1 billion parameters. Considering the significant overlap between the source sentence S and the polished sentence T, we split S into several sub-sentences by commas and require the model to generate only the modified sub-sentences. We describe the detailed input-output format in §4.1. Moreover, we retrieve relevant modifiers from the training corpus, which are taken as input for both training and inference to help the model find suitable modifiers. We show the retrieval augmentation

[1]《1000篇好词好句好段（初中）》，《小学生好词好句好段手册（新课标教材版）》，《书通网》，《黄冈作文-小学生好词好句好段》。

[2]https://www.leleketang.com/zuowen/list10-0-0-1-1.shtml

[3]An engaging text example collected from books may contain less than three sentences. In this case, M or N can be missing.

[4]https://github.com/HIT-SCIR/ltp

algorithm in §4.2.

4.1 Input-Output Format

In our pilot experiments, we take the concatenation of M, S and N as input[5] and train the model to minimize the log-likelihood of the whole target output T. We observe that the model tends to directly copy S as the generation result. We conjecture this is because most tokens in T overlap with S during training, making the model take the shortcut of copying instead of generating new tokens. To alleviate this issue, we split the source sentence S into several sub-sentences by commas (S without commas is not split), and train the model to decode only the modified sub-sentences. As shown in Figure 1, the source sentence S is split into 5 sub-sentences by commas, and the decoder only needs to decode the second and the fourth sub-sentence since the left three sub-sentences remain unchanged. This training strategy not only reduces the ratio of generated tokens that completely copy from the source inputs, but also improves the generation speed. Finally, we use the generated sub-sentences to replace the original ones in S to obtain the whole output sentence.

4.2 Retrieval Augmentation

We observe that the model trained with the framework described in §4.1 sometimes adds unsuitable modifiers (e.g., using "colourful" to modify "sun"). To alleviate this problem, we propose a retrieval augmentation algorithm to prompt the model to find suitable modifiers.

To this end, we first collect all pairs of modifiers and corresponding modified words from the engaging sentences in the training corpus, including attribute words paired with the modified nouns, and adverbial words paired with the modified verbs. We identify the attribute and adverbial words in each sentence through the dependency parsing toolkit of LTP. Furthermore, we restrict that the identified attribute and adverbial words are included in the vocabulary of engaging words. In this way, we obtain a dictionary that can map a noun or verb to a list of its suitable modifiers.

During inference, we first find all nouns and verbs in the source sentence S that are included in the dictionary and do not have engaging modifiers. Then, we randomly sample at most five modifiers for each noun or verb from its corresponding list of modifiers. The sampled modifiers along with the nouns and verbs are inserted before the original input. **During training**, considering that conditioning on too many modifiers that are not used for generating may make the model tend not to use the retrieved modifiers, we drop the retrieved modifiers corresponding to the nouns or verbs that do not have any modifiers in the target output with a probability p_1. Furthermore, to avoid excessive dependence on the retrieved modifiers, for those nouns or verbs that have modifiers in the target output, we drop all corresponding modifiers with a probability p_2, randomly sample at most five modifiers with the probability p_3, or randomly sample at most four modifiers along with the ground-truth modifier with a probability p_4. Note that $p_2 + p_3 + p_4 = 1$. Finally, we insert the selected modifiers with the nouns and verbs before the original input.

5 Experiments

5.1 Compared Models

We compare our model with the following three variants: **(1) Retrieve,** which randomly samples modifiers from the retrieved phrases, and then adds the sampled modifiers to the sentence without using neural language model. **(2) LongLM$_{\text{vanilla}}$,** which generates the whole polished sentence instead of only generating the modified sub-sentences; **(3) LongLM$_{\text{no-retrieve}}$,** which generates the modified sub-sentences without retrieval augmentation.

5.2 Experiment Settings

We initialize our model using the pretrained checkpoint of LongLM$_{\text{Large}}$. For retrieval augmentation, the probability p_1, p_2, p_3, p_4 is set to $0.75, 0.25, 0.25, 0.5$, respectively. And the sampled modifiers are dynamically changed at different epochs during training. We run our experiments on 6 Tesla V100 GPUs (32GB memory). We use DeepSpeed[6] with mixed precision to train our model, which helps significantly reduce the memory usage. We set learning rate to 5e-5 and batch size per GPU to 8. The maximum input length is set to 384 and the maximum output length is set to 128. We train the model for 10 epochs and select the best checkpoint that has the lowest perplexity on the validation set. During inference, we combine beam search (beam size = 10) (Graves, 2012), top-k sampling ($k = 50$) (Fan et al., 2018) and top-p

[5] M, S and N are separated by <sep>.

[6] https://github.com/microsoft/DeepSpeed

sampling ($p = 0.9$) (Holtzman et al., 2020) for decoding. We apply these settings to all models.

5.3 Automatic Evaluation

We evaluate the models on both $\text{Test}_{\text{Auto}}$ and $\text{Test}_{\text{Real}}$. We adopt the following two metrics: **(1) Copy ratio:** It calculates the ratio of samples whose output and input are exactly the same.**(2) # Added Modifiers**: It calculates the averaged number of added modifiers in the outputs. These two metrics aim to measure the differences between the inputs and outputs.

Models	Copy Ratio	# Added Modifiers
$\text{LongLM}_{\text{vanilla}}$	3.8% / 38.6%	1.16 / 0.40
$\text{LongLM}_{\text{no-retrieve}}$	0.2% / 8.5%	1.27 / 0.94
Ours	0.3% / 7.1%	1.23 / 0.92

Table 3: Automatic evaluation result. The two values separated by "/" indicate the performance on $\text{Test}_{\text{Auto}}$ and $\text{Test}_{\text{Real}}$, respectively.

The automatic evaluation result is shown in Table 3. We do not report the results of the Retrieve model because it adds as many modifiers as possible without considering fluency. $\text{LongLM}_{\text{vanilla}}$ has the highest copy ratio and adds the fewest modifiers among the three models. Moreover, all three models tend to copy more from inputs on $\text{Test}_{\text{Real}}$ than $\text{Test}_{\text{Auto}}$, and $\text{LongLM}_{\text{vanilla}}$ shows a larger margin than other two models which only generate the modified sub-sentences. The result suggests the worse generalization ability of $\text{LongLM}_{\text{vanilla}}$. Besides, we find that our model has a lower copy ratio than $\text{LongLM}_{\text{no-retrieve}}$ when tested on $\text{Test}_{\text{Real}}$, indicating that the retrieval augmentation module helps improve generalization to real texts by providing references for adding modifiers.

Aspects	Scores	Descriptions
Fluency	0	The output is obviously not fluent.
	1	The output is a little bit not fluent.
	2	The output is fluent.
Correctness	0	The modifiers in the output are incorrect.
	1	The correctness of the modifiers in the output is ambiguous.
	2	The modifiers in the output are correct.
Engagingness	1	The Engagingness of the output drops.
	2	The engagingness of the output is unchanged.
	3	The engagingness of the output improves slightly.
	4	The engagingness of the output improves.
	5	The engagingness of the output improves significantly.

Table 4: Scoring rules in manual evaluation.

Models	Fluency (κ)	Correctness (κ)	Engagingness (κ)
Retrieve	0.82 (0.42)	0.61 (0.50)	2.00 (0.64)
$\text{LongLM}_{\text{vanilla}}$	1.93 (0.69)	1.87 (0.52)	2.91 (0.75)
$\text{LongLM}_{\text{no-retrieve}}$	1.81 (0.55)	1.73 (0.52)	3.21 (0.76)
Ours	1.88 (0.57)	1.84 (0.57)	3.44 (0.85)

Table 5: Manual evaluation result. We show Fleiss's kappa value κ in the parentheses to measure the inter-annotator agreement.

5.4 Manual Evaluation

Considering there may be many plausible modifications for the same input, it is hard to automatically evaluate the quality of the added modifiers. Therefore, we resort to manual evaluation in terms of three aspects including: **(1) Fluency (0-2):** whether the polished sentence is fluent in terms of grammatical quality; **(2) Correctness (0-2):** whether the added modifiers in the polished sentence are suitable to modify the corresponding nouns, verbs, etc.; **(3) Engagingness (1-5):** whether the engagingness of the polished sentence improves compared with the source sentence. We show the detailed scoring rules in Table 4. We first randomly sample 100 inputs from $\text{Test}_{\text{Real}}$. Then we use the Retrieve model, $\text{LongLM}_{\text{vanilla}}$, $\text{LongLM}_{\text{no-retrieve}}$ and our model to generate polished sentences for the sampled inputs. For each generated sample, we hire three well-trained professional annotators to give a score for each of the three evaluation aspects. Note that these aspects are evaluated independently. We directly average the scores given by three annotators to get the final scores.

Table 5 shows the evaluation results. All results show moderate or better ($\kappa > 0.4$) inter-annotator agreement. By comparing $\text{LongLM}_{\text{vanilla}}$ and $\text{LongLM}_{\text{no-retrieve}}$, we can see that only generating the modified sub-sentences helps improve the engagingness of the polished sentence due to the lower copy ratio. However, the drop of copy ratio also brings a higher risk of adding unsuitable modifiers. Our retrieval augmentation algorithm improves the correctness of the polished sentences by providing multiple possible modifier candidates. Moreover, the suitable modifiers make the polished sentences more fluent and engaging. However, if we remove LongLM and only utilize the retrieved modifiers, it is hard to create fluent and coherent sentences as the result shows, which suggests the necessity to integrate the contextualization ability of generation models. In summary, our model

M	*S*	*N*	LongLM$_{\text{no-retrieve}}$	Ours
从远处看，桃花星星点点，似一群娇小可爱的小女孩，从近处看，这桃花恰似那闭月羞花的少女，在风中轻歌曼舞，柔美的身姿随风摇摆。(From a distance, the peach blossoms are dotted, like a group of petite and cute little girls. From a close point of view, this peach blossom is just like the girl who closes the moon and shy flowers, singing and dancing softly in the wind, and the feminine posture sways with the wind.)	冬天，桃花树养精蓄锐，吮吸着冬天仅有的丝丝阳光，它们饱经风霜，静待那一声撕破寒冬的惊雷；终于，明媚的春天来了，当迎春花开过，它们便向世人展示出那闭月羞花、沉鱼落雁的姿态。(In winter, peach blossom trees nourish themselves, sucking in the only shred of sunlight in winter. They are full of wind and frost, waiting for the thunder that tears through the cold winter. Finally, the bright spring has come, and when the spring flowers have bloomed, they show the world their beautiful posture.)	我觉得不是“梅花香自苦寒来，”还是“桃花艳自苦寒来。”(I think it's not just “plum blossom fragrance comes from bitter cold", but also“peach blossoms come from bitter cold".)	冬天，桃花树养精蓄锐，无助地吮吸着冬天仅有的丝丝阳光，它们饱经风霜，静待那一声撕破寒冬的惊雷；终于，明媚的春天来了，当迎春花开过，它们便向世人展示出那闭月羞花、沉鱼落雁的姿态。(In winter, peach blossom trees nourish themselves, helplessly sucking in the only shred of sunlight in winter. They are full of wind and frost, waiting for the thunder that tears through the cold winter. Finally, the bright spring has come, and when the spring flowers have bloomed, they show the world their beautiful posture.)	冬天，桃花树养精蓄锐，贪婪地吮吸着冬天仅有的丝丝阳光，它们饱经风霜，静待那一声撕破寒冬的沉闷的惊雷；终于，明媚的春天来了，当热情的迎春花开过，它们便骄傲地向世人展示出那闭月羞花、沉鱼落雁的姿态。(In winter, peach blossom trees nourish themselves, greedily sucking in the only shred of sunlight in winter. They are full of wind and frost, waiting for the dull thunder that tears through the cold winter. Finally, the bright spring has come, and when the enthusiastic spring flowers have bloomed, they proudly show the world their beautiful posture.) **Retrieved Phrases:** 沉闷的惊雷，贪婪地吮吸，热情的迎春花，充分地展示，火辣辣的阳光... (Dull thunder, greedily suck, enthusiastic spring flowers, fully show, fiery sunlight...)
秋风一阵阵吹来，一层层桔色的“海浪”迎面“扑”来，感觉像看四维电影一样。(The autumn wind blows in waves, and the layers of orange "waves" "flutter" in the face, feeling like watching a four-dimensional movie.)	仔细一看，像一大团桔红色的火焰在燃烧，花蕊被花瓣紧紧团住，最大的有爸爸拳头那么大。(If you look closely, it looks like a large orange-red flame burning, and the flower buds are tightly held by the petals, the largest of which is the size of Daddy's fist.)	花是桔红色的，绿色的叶子把花瓣裹住。(The flowers are orange-red, and the green leaves wrap the petals.)	*Same as the input source sentence S.*	仔细一看，像一大团桔红色的火焰在燃烧，花蕊被鲜艳的花瓣紧紧团住，最大的有爸爸拳头那么大。(If you look closely, it looks like a large orange-red flame burning, and the flower buds are tightly held by the brightly colored petals, the largest of which is the size of Daddy's fist.) **Retrieved Phrases:** 鲜艳的花瓣，纤细的花蕊... (brightly colored petals, slender flower buds...)

Table 6: Cases generated by different models on Test$_{\text{Real}}$. M, S and N are the preceding, source and subsequent sentences, respectively. LongLM$_{\text{vanilla}}$ copies the source sentences for both cases and we omit the generation results. We mark the added modifiers generated by LongLM$_{\text{no-retrieve}}$ in orange. In the generation result of ours, we mark the added modifiers that have been retrieved in red, and others in blue.

can improve the engagingness significantly[7] while keeping fluency and correctness comparable with baselines.

5.5 Case Study

We show two cases in Table 6. Our model can add suitable modifiers in multiple sub-sentences with the help of various retrieved modifiers. In contrast, LongLM$_{\text{no-retrieve}}$ uses "helplessly" to modify "sucking", which is reasonable in isolation but is incoherent with the context, thus decreasing the engagingness of the sentence. We additionally show the result of the Retrieve model in Table 7 in the appendix.

6 Demonstration

We have deployed our model online to automatically polish the source sentence given its context. Figure 2 shows a screenshot of our demo website. Users need to enter the source sentence and its preceding and subsequent sentences. Note that the source sentence is mandatory but its context can be empty. Then users can submit the request and the result will be returned after a few seconds. We show the polished sentence at the bottom of the page, and the retrieved modifiers for reference.

Figure 2: A screenshot of our demo website.

7 Conclusion

We propose a new task named sentence polishing, which requires polishing a given sentence while

[7] $p < 0.01$ when compared with LongLM$_{\text{vanilla}}$ (Wilcoxon signed-rank test).

maintaining fluency and coherence with the context. To this end, we construct about 160k parallel examples by removing modifiers in collected engaging sentences. Then we fine-tune LongLM to reconstruct the original sentences from the corrupted ones by generating the modified sub-sentences. We also propose a retrieval augmentation algorithm to retrieve engaging modifiers from the training set, which can help generate suitable modifiers. Automatic and manual evaluation demonstrate strong performance of our model to generate engaging sentences. We have deployed our model online for public use. Although we focus on adding modifiers in this paper, the perturbation-and-reconstruction framework can be potentially adapted to other polishing techniques such as adding metaphors, which is left as future work. Moreover, although we train our model on collected Chinese data, we believe the method can be easily transferred to other languages.

References

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara J Martin, and Mark O Riedl. 2020. Story realization: Expanding plot events into sentences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7375–7382.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Cristina Garbacea and Qiaozhu Mei. 2022. Why is constrained neural language generation particularly challenging? *arXiv preprint arXiv:2206.05395*.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Jian Guan, Zhuoer Feng, Yamei Chen, Ruilin He, Xiaoxi Mao, Changjie Fan, and Minlie Huang. 2022. Lot: A story-centric benchmark for evaluating chinese long text understanding and generation. *Transactions of the Association for Computational Linguistics*, 10:434–451.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

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.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020a. Commongen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840.

Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. 2020b. Pre-training multilingual neural machine translation by leveraging alignment information. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2649–2663.

Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Edit5: Semi-autoregressive text-editing with t5 warm-start. *arXiv preprint arXiv:2205.12209*.

Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. 2020. Felix: Flexible text editing through tagging and insertion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1244–1255.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. Plotmachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295.

Machel Reid and Victor Zhong. 2021. Lewis: Levenshtein editing for unsupervised text style transfer. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3932–3944.

Felix Stahlberg and Shankar Kumar. 2020. Seq2edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Stephen P Witte and Lester Faigley. 1981. Coherence, cohesion, and writing quality. *College composition and communication*, 32(2):189–204.

Shuoheng Yang, Yuxin Wang, and Xiaowen Chu. 2020. A survey of deep learning techniques for neural machine translation. *arXiv preprint arXiv:2002.07526*.

A Case Study

We additionally show two cases of the Retrieve model in Table 7. We can see that the Retrieve model can add unsuitable modifiers such as *lightsome* or duplicated modifiers (e.g., "一团团" after "一大团"). Moreover, the sentence generated by Retrieve is less fluent than the sentence generated by our model.

M	*S*	*N*	**Retrieve**	**Ours**
从远处看，桃花星星点点，似一群娇小可爱的小女孩，从近处看，这桃花恰似那闭月羞花的少女，在风中轻歌曼舞，柔美的身姿随风摇摆。 (From a distance, the peach blossoms are dotted, like a group of petite and cute little girls. From a close point of view, this peach blossom is just like the girl who closes the moon and shy flowers, singing and dancing softly in the wind, and the feminine posture sways with the wind.)	冬天，桃花树养精蓄锐，吮吸着冬天仅有的丝丝阳光，它们饱经风霜，静待那一声撕破寒冬的惊雷；终于，明媚的春天来了，当迎春花开过，它们便向世人展示出那闭月羞花、沉鱼落雁的姿态。 (In winter, peach blossom trees nourish themselves, sucking in the only shred of sunlight in winter. They are full of wind and frost, waiting for the thunder that tears through the cold winter. Finally, the bright spring has come, and when the spring flowers have bloomed, they show the world their beautiful posture.)	我觉得不是“梅花香自苦寒来，”还是“桃花艳自苦寒来。” (I think it’s not just “plum blossom fragrance comes from bitter cold", but also“peach blossoms come from bitter cold".)	冬天，桃花树养精蓄锐，尽情吮吸着冬天仅有的丝丝清澈的阳光，它们饱经风霜，静待那一声撕破凛冽的寒冬的沉闷的惊雷；终于，明媚的春天轻盈地来了，当许许多多的迎春花热热闹闹开过，它们便向世人努力展示出那闭月羞花、沉鱼落雁的轻盈的一种姿态。 (In winter, peach blossom trees nourish themselves, enjoyably sucking in the only shred of clear sunlight in winter. They are full of wind and frost, waiting for the dull thunder that tears through the nippy and cold winter. Finally, the bright spring has come airily, and when a lot of spring flowers have bloomed with high spirits, they show the world their beautiful and lightsome posture.)	冬天，桃花树养精蓄锐，贪婪地吮吸着冬天仅有的丝丝阳光，它们饱经风霜，静待那一声撕破寒冬的沉闷的惊雷；终于，明媚的春天来了，当热情的迎春花开过，它们便骄傲地向世人展示出那闭月羞花、沉鱼落雁的姿态。 (In winter, peach blossom trees nourish themselves, greedily sucking in the only shred of sunlight in winter. They are full of wind and frost, waiting for the dull thunder that tears through the cold winter. Finally, the bright spring has come, and when the enthusiastic spring flowers have bloomed, they proudly show the world their beautiful posture.) **Retrieved Phrases:** 沉闷的惊雷，贪婪地吮吸，热情的迎春花，充分地展示，火辣辣的阳光... (Dull thunder, greedily suck, enthusiastic spring flowers, fully show, fiery sunlight...)
秋风一阵阵吹来，一层层桔色的“海浪”迎面“扑”来，感觉像看四维电影一样。 (The autumn wind blows in waves, and the layers of orange "waves" "flutter" in the face, feeling like watching a four-dimensional movie.)	仔细一看，像一大团桔红色的火焰在燃烧，花蕊被花瓣紧紧团住，最大的有爸爸拳头那么大。 (If you look closely, it looks like a large orange-red flame burning, and the flower buds are tightly held by the petals, the largest of which is the size of Daddy’s fist.)	花是桔红色的，绿色的叶子把花瓣裹住。 (The flowers are orange-red, and the green leaves wrap the petals.)	仔细一看，像一大团桔红色的粗犷的火焰在一团团燃烧，漂亮的花蕊被娇嫩的小花瓣紧紧团住，最大的有爸爸拳头那么大。 (If you look closely, it looks like a large and clouds of orange-red and rough flame burning, and the beautiful flower buds are tightly held by the delicate petals, the largest of which is the size of Daddy’s fist.)	仔细一看，像一大团桔红色的火焰在燃烧，花蕊被鲜艳的花瓣紧紧团住，最大的有爸爸拳头那么大。 (If you look closely, it looks like a large orange-red flame burning, and the flower buds are tightly held by the brightly colored petals, the largest of which is the size of Daddy’s fist.) **Retrieved Phrases:** 鲜艳的花瓣，纤细的花蕊... (brightly colored petals, slender flower buds...)

Table 7: Cases generated by the Retrieve model on the same test examples as Table 6. We mark the added modifiers generated by the Retrieve model in orange. We also show the generation result of our model for reference.