Preserve Context Information for Extract-Generate Long-Input Summarization Framework

Ruifeng Yuan[1], **Zili Wang**[2], **Ziqiang Cao**[3] *, **Wenjie Li**[1] *
[1] The Hong Kong Polytechnic University
[2] Xiaohongshu Inc
[3] Institute of Artificial Intelligence, Soochow University, China

Abstract

The Extract-generate framework has been a classic approach for text summarization. As pretrained language models struggling with long-input summarization for their high memory cost, extract-generate framework regains researchers' interests. However, the cost of its effectiveness in dealing with long-input summarization is the loss of context information. In this paper, we present a context-aware extract-generate framework (CAEG) for long-input text summarization. It focuses on preserving both local and global context information in an extract-generate framework with little cost, and can be applied to most of existing extract-generate summarization models. CAEG generates a set of context-related text spans called context prompts for each text snippet and use them to transfer the context information from the extractor and generator. To find such context prompts, we propose to capture the context information based on the interpretation of the extractor, where the text spans having the highest contribution to the extraction decision are considered as containing the richest context information. We evaluate our approach on both long-document and long-dialogue summarization datasets: arXiv and QMSum. The experiment results show that CAEG achieves the-state-of-art result on QMSum and outperforms other extract-generate based models in arXiv.

Introduction

Text summarization aims to generate a concise version of the source document while preserve its salient information. There are mainly two types of approaches for text summarization, extractive summarization and abstractive summarization. As a combination of the two approaches, extract-generate framework has been widely used in text summarization. It first extracts salient text snippets from the source document with an extractor model and then compresses the extracted text as the summary with a generator model. As pretrained language models struggling with long-input summarization due to the high memory complexity of full self-attention, extract-generate summarization framework regains researchers' interests.

Instead of focusing equally on the whole source document like other long-input summarization models including sparse attention transformers and hierarchical models,

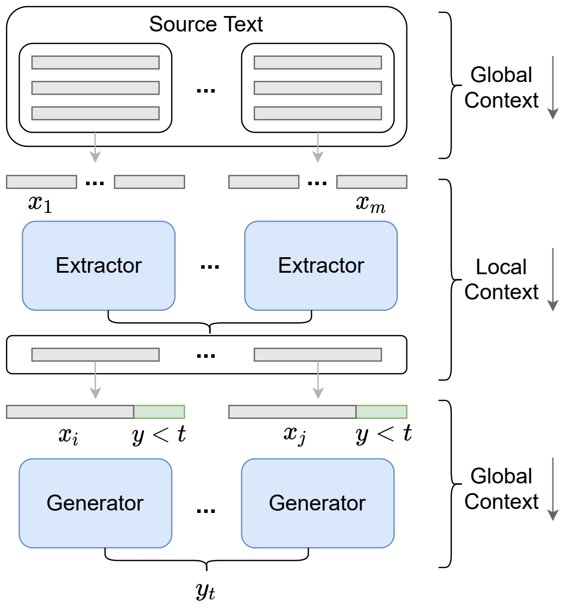

Figure 1: The extract-generate framework for long-input summarization. The grey strips stands for text snippets such as sentences or utterances from the source document. This figure shows that in the extract generate summary framework, both local and global context information is lost.

the extract-generate summarization framework is based on the assumption that only a part of salient source document is useful for the summarization. This not only makes the framework follow the human intuition when dealing with long input-summarization, but also allows it to handle the summarization input at any length. Hence, the extract-generate summarization framework achieves a success in long-input text summarization (Pilault et al. 2020; Zhao, Saleh, and Liu 2020; Bajaj et al. 2021; Zhang et al. 2021; Mao et al. 2021). However, as shown in Figure 1, the cost of its effectiveness in handling long-input summarization is the loss of context information. One the one hand, there exist a gap between the extractor and generator, where the context information can not be transferred from the continuous rep-

*Corresponding authors

resentation in extractor to the dispersed snippets in input of the generator. On the other hand, the framework leverages a chunking strategy in both extractor and generator to process a longer input, which obstructs the access to global context information between the chunks. Some previous works (Xiao and Carenini 2019; Cui and Hu 2021) has noticed this problem and propose context-aware extractive models to capture the global context information in extractor. Unfortunately, they do not solve this problem in the perspective of the whole framework and the context information still can not be effectively transferred from extractor to generator.

In this paper, we aim to investigate the influence of the context information and propose a context-aware extract-generate framework (CAEG) for long-input summarization. It aims to preserve both local and global context information from the extractor to the generator and utilize them to enhance the generation process. To build a bridge between the extractor and generator, CAEG generates a set of context-related text spans called context prompts for each text snippets and use them to transfer the context information. To generate such context prompts, we propose to capture the context information through the interpretation of the extractor. Considering that context information is one of the decisive factors of the extractive summarization, we assume that the text spans having the highest contribution to the extraction decision is considered as containing the richest context information. Here, we adopt a attention-based interpretation approach called attention rollout (Abnar and Zuidema 2020) to generate the context prompts. Then we add the context prompts to the generation process through simple concatenation. In terms of local context information, we concatenate the extracted snippets with their corresponding context prompts. In terms of global context information, each extracted snippet are concatenated with context prompts from its most related snippet to capture the global dependency. In this case, CAEG can be easily applied to most of existing summarization models based on extract-generate framework without largely increasing the complexity or memory cost of the model.

We conducted experiments on two long-input summarization datasets: arXiv (Cohan et al. 2018) for long-document summarization, and QMSum (Zhong et al. 2021) for long-dialogue summarization. Taking a recent proposed extract-generate summarization model DYLE (Mao et al. 2021) as the backbone, our approach achieves improvement on both datasets compared to the base model and obtains the state-of-the-art result on QMSum. These experiments suggest the effectiveness of CAEG in preserving the context information. We conclude our contributions as follows:

- We firstly investigate the influence of context information loss for extract-generate framework in long-input summarization and propose the CAEG for this problem.
- We introduce a new approach for capturing the context information based on the interpretation of the extractor.
- The experiment result shows that CAEG is capable of effectively preserving the context information from the extractor to the generator without largely increasing the complexity or memory cost of the model.

Related Work

Extract-Generate Text Summarization

Coarse-to-fine frameworks containing multiple stages are used in many text generation tasks, such as text summarization, dialogue state tracking, and neural storyline generation. In text summarization, the two-stage extract-generate framework is commonly used. This framework first extracts important text snippets from the input, followed by generating an overall summary. Some researchers (Mehdad et al. 2013; Lebanoff et al. 2019) propose to extract a set of similar sentence clusters and then fuse each cluster to obtain a summary sentence. More researchers (Pilault et al. 2020; Zhao, Saleh, and Liu 2020; Bajaj et al. 2021; Zhang et al. 2021) apply extract-generate framework to the long-input summarization to overcome the capability limitation of transformer models such as BERT or PEGASUS. DYLE (Mao et al. 2021) further extends the idea and presents a dynamic latent extraction approach that generate dynamic snippet-level attention weights during decoding. However, these models seldom focus on the loss of the context information in the extract-generate framework. Hence, we aim to enhance these existing extract-generate summarization models with the context information using a cost-efficient approach in this paper.

Long-Input Summarization

Long-input summarization has attracted more attention recently in different domains such as news, science paper, dialogue and etc. One solution is to adopt transformers with the sparse attention mechanism. Longformer (Beltagy, Peters, and Cohan 2020) combines sliding windows with global attention patterns, while BigBird (Zaheer et al. 2020) uses sliding windows and random blocks. Reformer (Kitaev, Kaiser, and Levskaya 2020) adopts the locality-sensitive hashing to replace the dot-product attention. Despite focusing on self-attention, some researchers (Huang et al. 2021) proposes head-wise positional strides to improve the efficiency of the cross-attention. Some hierarchical models also have been proposed for long-input summarization. HAT-Bart (Rohde, Wu, and Liu 2021) presents a transformer with hierarchical attention that utilizes information from both sentence and paragraph-level. HMNet (Zhu et al. 2020) proposes a hierarchical structure that captures discourse-level information and speaker roles for dialogue summarization. As mentioned above, the extract-generate summarization framework also becomes a major solution for handling longer input. Compared with the other two types of models, we believe models based on this framework achieve a good balance between performance and computational cost.

Method

The extract-generate framework has been widely used in long-input summarization. It is usually composed of two transformer-based models, an extractor that extracts salient snippets from the source document and a generator that compress the extracted snippets into summaries. In this paper, we aim to enhance such framework by preserving previously lost context information. An overview of our proposed approach CAEG is shown in Figure 2. In the first subsection, we briefly introduce the extractor-generator framework for long-input summarization. In the second subsection, we introduce how we extract the context prompts based on the interpretation of the extractor and use it to transfer the local context information. The context prompt can also be used on preserving the global context information, which we elaborate on in the third subsection. The application of CAEG is shown in the last subsection.

Extract-Generate Framework

In an extract-generate summarization framework, the input is composed of m text snippets, $X = (x_1, ..., x_m)$, and an optional query q if it is a query-focused summarization task. The output is a summary y containing T tokens. In terms of document summarization, we take each sentence as a snippet. In terms of dialogue summarization, dialogue utterances are regarded as snippets. The framework aims to generate a sequence of summary tokens y given the input text X and the generated tokens in previous steps $y < t$ with an extractor E_η and a generator G_θ:

$$P_\theta(y \mid q, X) = \prod_{t=1}^{T} P_{\theta+\eta}(y_t \mid q, X, y_{<t}) \tag{1}$$

The goal for extractor is to output a score s_i for each text snippet x_i taking the source text and the optional query as input. However, limited by GPU memory, it is impractical to encode the full source text within in one language model. Hence, the text snippets are divided into multiple chunks $X = (c_1, ..., c_u)$, each containing consecutive snippets. We feed each chunk and the optional query to the extractor and compute score for each snippet in the chunk. Then top-N snippets X_N with the highest scores are extracted from the document X:

$$X_K = topN(E_\eta(q, x_i, c_j), x_i \in c_j, c_j \in X) \tag{2}$$

Note that even the extracted snippets in long-input summarization may still exceed the capability of a generation model. Some researchers adopt a strategy called fusion in decoder. It allows the generator first encode each extracted snippet independently and then concatenate the hidden states of all snippets as the input of the decoder. DYLE (Mao et al. 2021) proposes to predict the generation probability and the dynamic weight on each snippet and obtains the final generation probability by marginalizing over all extracted snippets.

In this paper, we adopt the same generator proposed by DYLE. It feeds each extracted snippet to the model and obtains the hidden state h_i^t and generation probability on every decoding time step. An additional MLP is used to map the hidden state to a scalar weight. The dynamic weight allows the generator to focus on different snippets at different time steps. The final generation probability at time step t is computed by:

$$y_t = \sum_{x_i \in X_N} G_\theta(q, x_i, y_{<t})\, MLP(h_i^t) \tag{3}$$

Local Context Information

Local context refers to the neighboring snippets of a target snippet within one chunk. Such information is fully utilized by the extractor to extract the salient snippets. However, due to the extracted snippets are transformed back to discrete representation after the extraction, the local context information can not be transferred to the generator. For example, we assume there is a snippet x_i mentioned entity A and entity B and its local context mentioned entity A multiple times. An extractor can easily extract the snippet x_i, since entity A is considered salient information. When snippet x_i is fed into the generator, the model can no longer identify whether entity A or entity B is the crucial one. In this case, the loss of local context information creates negative effects on the summarization.

The problem lies ahead is how to obtain the local context information from the extractor and apply it to the generator. An intuitive solution is to extend each extracted snippet by concatenating its neighboring snippets with it, which inevitably brings noise and efficiency drop. Another solution is to transfer the snippet representations in the extractor to the generator. However, these representations may not be effective, since the extractor and generator are usually independent models. A wildly accepted idea is that extractive summarization mainly depends on the context information. Inspired by this, we make an assumption that the text spans that contribute the most to the snippet extraction contain the richest context information. We named these text spans context prompts. In CAEG, we generate these context prompts through the interpretation of the extractor, and use them to represent the local context information.

A commonly used interpretation approach for a transformer-based model is the attention distribution. However, information originating from the input tokens gets increasingly mixed across layers of the Transformer, which makes attention weights unreliable as explanations. Hence, we adopt attention rollout (Abnar and Zuidema 2020) to interpret the extractor. Its glob is to quantify the flow of information in self-attention layers by simulating the information propagated from the input layer to the higher layers. Attention rollout assumes information propagation in a transformer as a directed acyclic graph, where the nodes refer to the token representations at each layer and edges denote the attention. The attention weight is regarded as the proportion of information transferred between two nodes. In this case, the information propagated between two nodes through a certain path is computed by multiplying the weights of all edges in the path. Considering there exist multiple paths between two nodes, the total amount of information is the sum of all possible paths between two nodes. At the implementation level, the attentions rollout score at i layer is computed by recursively multiplying the attention weight matrices in all layers below:

$$\tilde{A}(l_i) = \begin{cases} A(l_i)\,\tilde{A}(l_{i-1}) & if \quad i > 0 \\ A(l_i) & if \quad i = 0 \end{cases} \tag{4}$$

where $A = 0.5W_{att} + 0.5I$ refer to the raw attention updated by residual connections, $\tilde{A}$ represents the attention

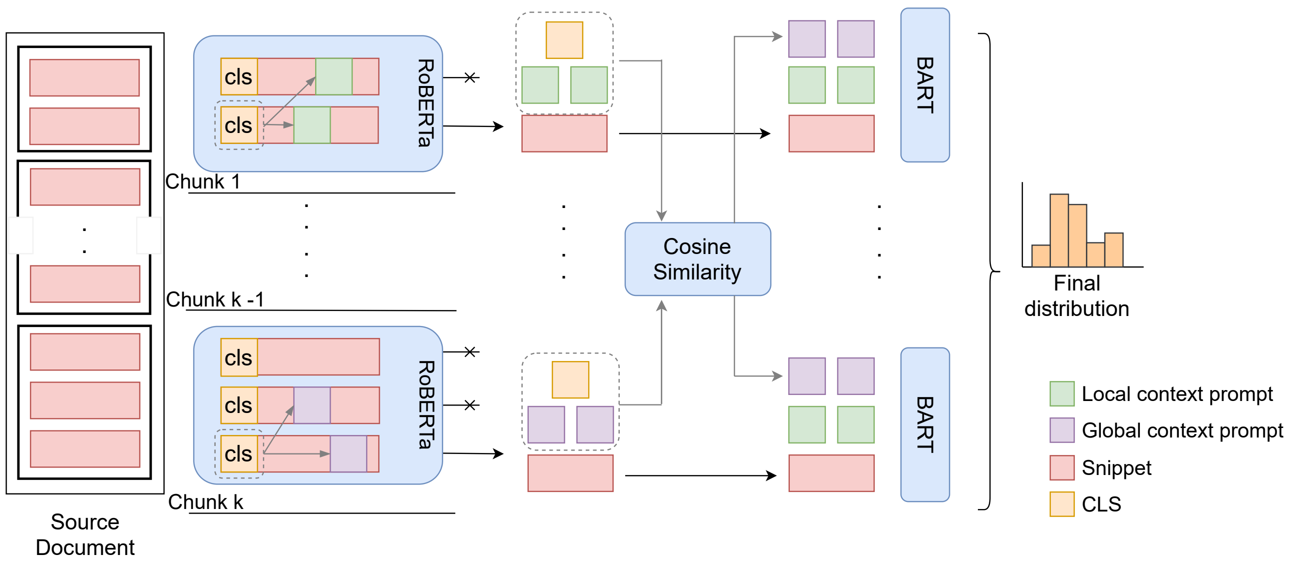

Figure 2: The framework of our proposed CAEG framework. The extractor we used is RoBERTa-base, and the generator is BART-large. Here, we use the blocks with various colors to represent the different types of information in the model and adopt the dot lines to emphasize the information flow added for preserving the context information. To obtain a clear observation, we simplified the structure of the generator in the figure.

rollout, and the multiplication operation denotes matrix multiplication.

As shown in Figure 2, for an extracted snippet x_i, we use the attention rollout score of its [CLS] token to extract K context prompts $L_i = [l_i^1, l_i^2, .., l_i^k]$. Here, we mask the attention rollout scores for all [CLS] tokens, since [CLS] token itself does not contain any effective context information. A sliding window strategy is adopted to extract the text spans containing the highest average attention rollout score, which are considered as the explanation for extracting the snippet. These text spans are used as the context prompts for the extracted snippet x_i. In the application, the window size is set to 8 tokens. One thing that is worth noticing is that the context prompts are not limited to the extracted snippet, but also disperse in its neighboring snippets.

These extracted context prompts are used to enhance the generation process in the same way as prompt-based language models. Given an extracted snippet x_i, we concatenate it with its corresponding context prompts L_i as the input for the generator:

$$y_t = \sum_{x_i \in X_N} G_\theta(q, L_i, x_i, y_{<t}) MLP(h_i^t) \tag{5}$$

Note that all the component of the input are not directly concatenated together, instead we use a set of special tokens to separate these components.

Global Context Information

The extract-generate framework split the input of the extractor and the generator into multiple chunks to handle a longer input. The cessation of the information interaction between chunks inevitably leads to the loss of information that captures the long-term dependency, which leads to loss

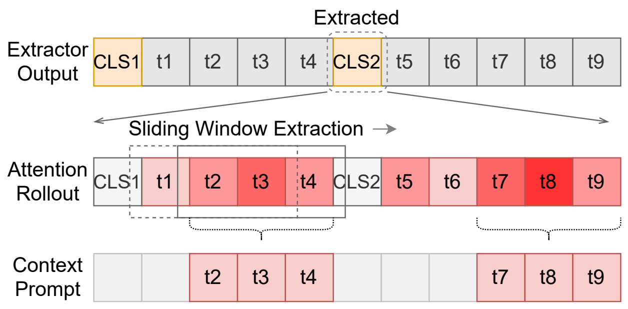

Figure 3: The generation of context prompts. The yellow squares are the [cls] tokens and the grey squares denote the text tokens. The darker red stands for a higher attention rollout score.

of global context information. Previous works (Xiao and Carenini 2019; Cui and Hu 2021) mainly focus on capturing such context information in the extractor. However, similar to the local context information, these information is difficult to be transferred to the generator.

A commonly used method for capturing the global context information of a snippet is to search for its related snippets in the source document. Here, we adopt a similar strategy but in the form of the context prompt. For an extracted snippet x_i, we adopt the context prompts from its most related snippet as its global context prompts G_i. We use the cosine similarity between the snippet representations R in the extractor to select the most related snippet for x_i. Considering a search space of all source snippets inevitably brings noise, in practice, we reduce the search space to top-score

Dataset	Type	Domain	Size	Source length	Target length	Query
QMsum	dialogue	meeting	1808	9070	70	$\surd$
Arxiv	document	science paper	200000	6030	273	$\times$

Table 1: The statistics and comparison of datasets.

snippets that have already been chosen in the snippet extraction stage. Follow the same way used for local context information, global context prompts are used to enhance the generation process:

$$y_t = \sum_{x_i \in X_N} G_\theta(q, G_i, x_i, y_{<t}) MLP(h_i^t) \tag{6}$$

Application

Instead of adding more neural network architecture, CAEG preserves context information by adjusting the input and output of the extractor and the generator. This allows it to be easily applied to any trained extract-generate summarization model as long as it has a transformer-based extractor. In this paper, we adopt DYLE (Mao et al. 2021) as the base model. It is worth noting that we still need to finetune the generator, which makes it adapt to additional contextual information input. To utilize both local and global context information in the framework, we concatenate the two types of context prompts and add them to the generator.

Experiment

Dataset

We evaluate our proposed methods in the context of long-input summarization. Two datasets from different domains are adopted as evaluation benchmarks. The detailed comparison is shown in Table 1. **arXiv**(Cohan et al. 2018) is a dataset for long-input single-document summarization. It collects scientific articles from arXiv.org and takes the abstracts of these articles as the target summaries. Compared to previous news summarization datasets, it has significantly longer input and output. **QMSum**(Zhong et al. 2021)is a benchmark for query-focused dialogue summarization. The dataset is composed of meeting records from three different domains. Since only a small proportion of the source document is correlated to the query, QMSum has a higher compression rate than other long-input summarization datasets.

Baseline

We compare our method with some commonly used pretrained models and previous state-of-the-art methods designed for long-input summarization. The pretrained models include Bart-large (Lewis et al. 2019) and PEGASUS (Zhang et al. 2020). There are three kinds of methods for long-input summarization: transformers with sparse attention, hierarchical transformers, and models based on the extract-generate framework, which are shown below:

- **Transformers with sparse attention:** We adopt various sparse-attention transformers for comparison including Longformer (Beltagy, Peters, and Cohan 2020), BigBird (Zaheer et al. 2020), and LSH(Huang et al. 2021). DialogLM (Zhong et al. 2022) is a pretrained model for dialogue understanding, and we display it with both full attention and sinkhorn sparse attention.
- **Hierarchical transformers:** HMNet (Zhu et al. 2020) designed a hierarchical structure for dialogue summarization, which includes discourse-level information and speaker roles. HAT-BART (Rohde, Wu, and Liu 2021) proposes a Transformer-based model with hierarchical attention. It is capable of capturing information among sentence and paragraph-level.
- **Models based on extract-generate framework:** Dyle (Mao et al. 2021) is the state-of-the-art summarization model based on extract-generate framework, which is also our base model. BM25+Bart refers to a baseline model taking BM25 as extractor and Bart as the generator, which is drawn from (Zhang et al. 2021). DANCER (Gidiotis and Tsoumakas 2020) propose a divide-and-conquer approaches. Moreover, we also report two extractive summarization models for long-input summarization: ExtSum-LG (Xiao and Carenini 2019) and SSN-DM (Cui and Hu 2021).

Model	R-1	R-2	R-L
Bart-large (3072)	32.16	8.01	27.72
HMNet (8192)	32.29	8.67	28.17
Longformer (8192)	31.60	7.80	20.50
DialogLM (5120)	34.02	9.19	29.77
DialogLM - Sparse (8192)	33.69	9.32	30.01
BM25+Bart (dyn)	32.9	9.0	22.0
DYLE (dyn)	34.42	9.71	30.10
CAEG-local (dyn)	**36.50**	11.15	**30.50**
CAEG-global (dyn)	36.11	11.22	30.24
CAEG-all (dyn)	36.41	**11.41**	30.21

Table 2: Results on QMSum.

Model	R-1	R-2	R-L
PEGASUS (3072)	44.21	16.95	38.83
BigBird-PEGASUS (3072)	46.63	19.02	41.77
LSH (7168)	**48.24**	**20.26**	41.78
HAT-BART (3072)	46.68	19.07	42.17
ExtSum-LG (dyn)	44.01	17.79	39.09
DANCER-PEGASUS (dyn)	45.01	17.60	40.56
SSN-DM (dyn)	45.03	19.03	32.58
DYLE (dyn)	46.41	17.95	41.54
CAEG-local (dyn)	46.69	18.48	42.04
CAEG-global (dyn)	46.86	18.61	**42.20**
CAEG-all (dyn)	46.81	18.58	42.16

Table 3: Results on arXiv.

	QMSum						arXiv					
	Local			Global			Local			Global		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
K=1	35.46	10.37	29.56	35.38	10.36	29.34	46.56	18.44	41.88	46.69	18.51	42.01
K=2	35.79	10.77	29.79	36.09	10.88	29.83	46.62	18.46	41.94	**46.86**	**18.61**	**42.20**
K=3	36.05	10.78	30.15	**36.11**	**11.22**	**30.24**	**46.69**	**18.48**	**42.04**	46.72	18.50	42.05
K=4	**36.50**	**11.15**	**30.50**	35.15	10.28	29.15	46.56	18.42	41.90	46.47	18.30	41.88

Table 4: Analysis of the number of context prompts.

We add the maximum input sequence length of the model in the brackets after the model, and "dyn" represents dynamic denoting there is no limitation for the input length.

Implementation Details

Taking the state-of-the-art extract-generate summarization model DYLE as the backbone, we adopt Roberta-base (Liu et al. 2019) as the extractor and Bart-large (Lewis et al. 2019) as the generator. Both models are initialized by the checkpoint given by DYLE. The implementation of our code is based on transformers from Hugging Face. Adam algorithm is used for optimization and the learning rate is set to $2 - e^{-6}$. The batch size for the training is set to 1, and gradient accumulation steps are set to 8. We conduct the validation for every 100 steps and train the model for a maximum of 20000 steps. The experiments are run on a single V100 GPU. In terms of the inference stage, we adopt a beam search size 4 for arXiv and a beam search size 1 for QMSum. The length limitation is 150 to 450 for arXiv and 50 to 100 for QMSum.

Experiment Results

The evaluation results are summarized in Table 2 and Table 3. In the experiment, following previous works, we adopt ROUGE 1.5.5 (Lin 2004) including Rouge-1 (R-1), Rouge-2 (R-2), and Rouge-L (R-L) as evaluation metrics. There are three types of variants of our approach. (1) CAEG-local: We only add local context prompts to the generator; (2) CAEG-global: We only add global context prompts to the generator; (3) CAEG-all: We add both local context prompts and global context prompts to the generator, and the number of context prompt K is set to 2.

On QMSum, CAEG achieves the new state-of-the-art performance. Compared with the base model DYLE, all three variants of our model yield a clear improvement, which shows that CAEG outperforms previous extract-generate summarization models whose context information is ignored. These results suggest the importance of context information in the long dialogue summarization. CAEG's better performance can be attributed to its effectiveness in preserving context information between the extractor and the generator.

On arXiv, CAEG outperforms the other models based on the extract-generate framework, but fails to achieve the best result. We believe there are two reasons for this. On the one hand, CAEG is an approach that enhance extract-generate summarization model by utilizing the context information, so the final performance is partly dependent on the base model itself. Hence, even if we achieved improvement on the base model DYLE, it can not fill the natural gap between DYLE and LSH. On the other hand, the performance enhancement brought by context information on arXiv is not as large as QMSum. This might be because the structural patterns also play an important role in identifying salient information in the summarization of science papers. In this case, the context information becomes less useful.

	R-1	R-2	R-L
Context prompt	**36.50**	**11.15**	**30.50**
Adjacent snippet	35.74	10.78	29.46
Snippet embedding	33.26	8.66	27.36
No context(DYLE)	34.42	9.71	30.10

Table 5: The comparison between different approaches in preserving local context information on QMSum dataset.

It is a surprise that CAEG-all fails to further improve the performance after adding both local and global context information. A possible reason is that the generator cannot effectively distinguish the two types of context information. It is our future work to find a optimal way for this problem.

Analysis and Discussion

Effect of number of context prompts As suggested in the Method Section, for each extracted snippet, we generate K context prompts to preserve its context information. Here, we vary the value of the hyperparameter K and test it on both QMSum dataset and arXiv dataset in Table 4. The largest K we show in the table is 4, since a greater value leads to a clear drop in performance.

We can observe that the performance of the model increase when the value of K increase until it reach a upper bound around 3. This suggests that the context information requires multiple context prompts to be effectively represented. Moreover, the model achieves its best performance when k=4 for the local context and k=3 for the global context in QMSum, while the best K value are smaller for arXiv. This is expected as the long text snippets (utterance) in QMSum require more context information than the ones (sentence) in arXiv.

Effect of different forms in transferring context information We are interested in the most effective form to transfer the context information from the extractor to the generator. To investigate this, we evaluate the effectiveness of three different forms in transferring the local context information in Figure 5. (1) Context prompt: We report the best result of

QMSum	
Context Snippet	Suzy Davies Am: Do you think it would be better for us as scrutinisers of this act if we could see the draft changes to cps guidance on the public interest test before we make our final decision?
Extracted Snippet	Barry Hughes : I honestly don't think that would necessarily be helpful. I've had some discussions with Kwame, who would have an involvement in this. What we would envisage is that we would simply want to take the present public interest factors, which are set out, in my view, very clearly in the code for crown prosecutors ... And we'd need to work that up as we go along, and I think you'd run a risk of putting the cart before the horse, if I may put it like that.
Context Snippet	Suzy Davies Am: It's just that, personally, I think the public interest test is critical in all this...
Prompt w Context	[”see the draft changes to cps guidance”, ”want to take the present public interest factors”]
Prompt w/o Context	[”Barry Hughes : I honestly do”, ”if I may put it like that”]
arXiv	
Context Snippet	One may calculate the penetration probability numerically by using the path integral method or the wkb approximation.
Extracted Snippet	However, it is highly desirable to have an analytical expression for the barrier penetrability when one introduces an energy-dependent one-dimensional potential barrier @xcite or barrier distribution functions @xcite.
Context Snippet	In the present work, we derived a new barrier penetration formula based on the wkb approximation...
Prompt w Context	[”we derived a new barrier penetration formula based”, ”or barrier distribution functions @xcite”]
Prompt w/o Context	[”barrier distribution functions @xcite.”, ”to have an analytical expression for the barrier”]

Table 6: The case study of the context prompts on QMSum and arXiv. For each extracted snippet, we display two generated context prompts. For a better observation, we use the underline to emphasize the position of the generated context prompts in the source text.

our approach using the local context information. (2) Adjacent snippet: We concatenate each extracted snippet with its neighboring snippets and feed them into the generator. (3) Snippet embedding: For each extracted snippet, we can obtain its representation from the its corresponding [cls] token in extractor. The representation are fed to the generator by adding it before the input embeddings. Since the word dimension of the extractor and generator are not the same, we use a MLP to map the snippet embedding from the dimension of the extractor to the generator.

The results show that context prompt outperforms the other two ways in transferring the local context information. Although directly concatenating the neighboring snippets achieves a strong result, it leads to a huge increase in the GPU memory cost of the generator. Meanwhile, due to the noise brought by the large amount of context, it underperforms our proposed context prompt. As for using snippet embeddings, the results suggest that it can not effectively reflect the context information, especially when the distribution between the extractor and generator are not the same. Conducting a end-to-end training may solve this problem, but it will largely increases the computational cost.

Case study of context prompts To have a more clear understanding about how context prompts work, we display the context prompts of some extracted snippets in the Table 6. Here, we display the extracted snippet, the neighboring snippets of the extracted snippet (Context Snippet), and the generated context prompts (Prompt w Context). For a better comparison, we also show the salient text spans obtained in the same ways as context prompt but without context information (Prompt w/o Context). This is achieved by feeding the extracted snippet solely to the extractor.

In terms of the example from QMSum, we find the ”public interest test” is the important information among the context. The context prompts successfully capture such information and highlight a related topic ”changes to cps guidance”, while the prompt with no context information fails to achieve this. In terms of the example from arXiv, ”barrier” has been mentioned multiple times in the extracted snippet, which indicate its importance. However, the context prompt can not further focus on ”barrier penetration” without the context information. These examples not only suggest the importance of the local context information, but also show the effectiveness of the context information captured through the interpretation of the extractor.

Conclusions

In this paper, we firstly propose to focus on context information preservation in an extract-generate summarization framework for long input. To address the challenge of context information loss from the extractor to the generator, we generate a set of context-related text spans called context prompts for each extracted snippet and feed them into the generator through concatenation. A novel approach is proposed to generate the context prompts based on the interpretation of the extractor. Hence, our approach can be applied to most extract-generate summarization models at a low cost. The experiments further show its effectiveness in capturing and preserving the context information in the extract-generate summarization framework.

Acknowledgements

The work described in this paper was supported by Research Grants Council of Hong Kong (PolyU/15203617 and PolyU/5210919), National Natural Science Foundation of China (61672445, 62076212, 62106165).

References

Abnar, S.; and Zuidema, W. 2020. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*.

Bajaj, A.; Dangati, P.; Krishna, K.; Kumar, P. A.; Uppaal, R.; Windsor, B.; Brenner, E.; Dotterrer, D.; Das, R.; and McCallum, A. 2021. Long Document Summarization in a Low Resource Setting using Pretrained Language Models. *arXiv preprint arXiv:2103.00751*.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Cohan, A.; Dernoncourt, F.; Kim, D. S.; Bui, T.; Kim, S.; Chang, W.; and Goharian, N. 2018. A discourse-aware attention model for abstractive summarization of long documents. *arXiv preprint arXiv:1804.05685*.

Cui, P.; and Hu, L. 2021. Sliding Selector Network with Dynamic Memory for Extractive Summarization of Long Documents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 5881–5891.

Gidiotis, A.; and Tsoumakas, G. 2020. A divide-and-conquer approach to the summarization of long documents. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28: 3029–3040.

Huang, L.; Cao, S.; Parulian, N.; Ji, H.; and Wang, L. 2021. Efficient attentions for long document summarization. *arXiv preprint arXiv:2104.02112*.

Kitaev, N.; Kaiser, Ł.; and Levskaya, A. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Lebanoff, L.; Song, K.; Dernoncourt, F.; Kim, D. S.; Kim, S.; Chang, W.; and Liu, F. 2019. Scoring sentence singletons and pairs for abstractive summarization. *arXiv preprint arXiv:1906.00077*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mao, Z.; Wu, C. H.; Ni, A.; Zhang, Y.; Zhang, R.; Yu, T.; Deb, B.; Zhu, C.; Awadallah, A. H.; and Radev, D. 2021. Dyle: Dynamic latent extraction for abstractive long-input summarization. *arXiv preprint arXiv:2110.08168*.

Mehdad, Y.; Carenini, G.; Tompa, F.; and Ng, R. 2013. Abstractive meeting summarization with entailment and fusion. In *Proceedings of the 14th European Workshop on Natural Language Generation*, 136–146.

Pilault, J.; Li, R.; Subramanian, S.; and Pal, C. 2020. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 9308–9319.

Rohde, T.; Wu, X.; and Liu, Y. 2021. Hierarchical learning for generation with long source sequences. *arXiv preprint arXiv:2104.07545*.

Xiao, W.; and Carenini, G. 2019. Extractive summarization of long documents by combining global and local context. *arXiv preprint arXiv:1909.08089*.

Zaheer, M.; Guruganesh, G.; Dubey, K. A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33: 17283–17297.

Zhang, J.; Zhao, Y.; Saleh, M.; and Liu, P. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, 11328–11339. PMLR.

Zhang, Y.; Ni, A.; Yu, T.; Zhang, R.; Zhu, C.; Deb, B.; Celikyilmaz, A.; Awadallah, A. H.; and Radev, D. 2021. An Exploratory Study on Long Dialogue Summarization: What Works and What's Next. *arXiv preprint arXiv:2109.04609*.

Zhao, Y.; Saleh, M.; and Liu, P. J. 2020. Seal: Segment-wise extractive-abstractive long-form text summarization. *arXiv preprint arXiv:2006.10213*.

Zhong, M.; Liu, Y.; Xu, Y.; Zhu, C.; and Zeng, M. 2022. Dialoglm: Pre-trained model for long dialogue understanding and summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 11765–11773.

Zhong, M.; Yin, D.; Yu, T.; Zaidi, A.; Mutuma, M.; Jha, R.; Awadallah, A. H.; Celikyilmaz, A.; Liu, Y.; Qiu, X.; et al. 2021. QMSum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*.

Zhu, C.; Xu, R.; Zeng, M.; and Huang, X. 2020. A hierarchical network for abstractive meeting summarization with cross-domain pretraining. *arXiv preprint arXiv:2004.02016*.