From Reading to Compressing: Exploring the Multi-document Reader for Prompt Compression

Eunseong Choi, Sunkyung Lee, Minjin Choi, June Park, Jongwuk Lee*
Sungkyunkwan University, Republic of Korea
{eunseong, sk1027, zxcvxd, pj00515, jongwuklee}@skku.edu

Abstract

Large language models (LLMs) have achieved significant performance gains using advanced prompting techniques over various tasks. However, the increasing length of prompts leads to high computational costs and often obscures crucial information. Prompt compression has been proposed to alleviate these issues, but it faces challenges in (i) capturing the global context and (ii) training the compressor effectively. To tackle these challenges, we introduce a novel prompt compression method, namely *Reading To Compressing (R2C)*, utilizing the Fusion-in-Decoder (FiD) architecture to identify the important information in the prompt. Specifically, the cross-attention scores of the FiD are used to discern essential chunks and sentences from the prompt. R2C effectively captures the global context without compromising semantic consistency while detouring the necessity of pseudo-labels for training the compressor. Empirical results show that R2C retains key contexts, enhancing the LLM performance by 6% in out-of-domain evaluations while reducing the prompt length by 80%.

1 Introduction

Large language models (LLMs) have recently exhibited remarkable performance gains in various tasks owing to a wide variety of prompting, *e.g.*, Retrieval-Augmented Generation (RAG) (Gao et al., 2023), Chain-of-Thought (CoT) (Wei et al., 2022), and In-Context Learning (ICL) (Dong et al., 2023). The use of rich prompts unlocks the abilities of LLMs, but prompts can be verbose to deliver sufficient information. The lengthy prompts not only increase computational costs but also make LLMs struggle to discern important information. Although the input length limit has recently been extended to a million tokens (Reid et al., 2024), the quadratic increase in computation over the input length is still a substantial burden.

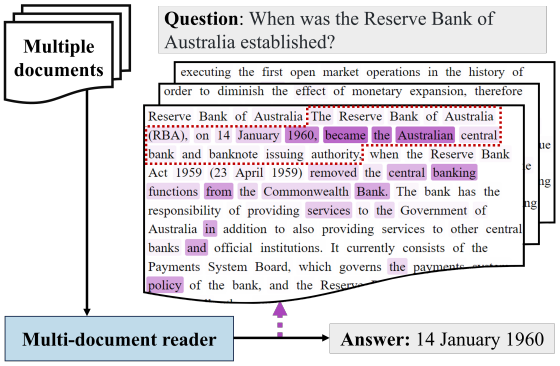


Figure 1: Multi-document reader, *i.e.*, Fusion-in-decoder (FiD), captures core information by learning to generate answers from lengthy inputs, as highlighted with the dotted red box. The darker the purple color, the higher the cross-attention score in FiD-decoder.

Recently, *prompt compression* has garnered significant focus for alleviating this issue. Its goal is to preserve only the essential information to reduce the computational overhead without sacrificing the accuracy of the end task. The most straightforward approach is *token pruning*, removing redundant tokens from the original prompt by entropy-based metrics (Li et al., 2023; Jiang et al., 2023a,b) or predicted scores (Xu et al., 2024; Huang et al., 2023; Pan et al., 2024). They utilize the perplexity of tokens using smaller models or classification scores of trained compressors. They can be used in a model-agnostic manner, delivering the benefit in black-box scenarios without understanding the internal structure of the LLMs.

However, existing prompt compression methods face two challenges in handling lengthy inputs.
(i) *How to extract essential information based on the global context?* The key points in a single paragraph may differ significantly from the main topic of the lengthy document. However, existing works divide the prompt into multiple segments if exceeding the maximum input length of compressors and compress each segment independently (Pan et al., 2024; Huang et al., 2023). Since the model iden-

*Corresponding author

Findings of the Association for Computational Linguistics: EACL 2024, pages 14734–14754

tifies crucial tokens only within the segment, it is limited in capturing essential information across the global context.

(ii) *How to train a compressor?* Since the ground truth of the compressed prompt is non-trivial to define, it is difficult to train the compressor. Recent work detours the issue by utilizing state-of-the-art LLMs, *e.g.*, GPT-4, for generating the pseudo-compressed prompts to train the compressor (Pan et al., 2024). However, as pointed out in Jiang et al. (2023a); Ali et al. (2024), GPT-4 underperforms in prompt compression, suggesting that there is still room for improvement in training compressors.

Addressing these issues, we shed light on one prominent solution to handle multiple documents, *i.e.*, *Fusion-in-Decoder (FiD)* (Izacard and Grave, 2021b). FiD is a question-answering model aggregating evidence from multiple documents to answer a question. It effectively captures the global context over multiple documents by leveraging cross-attention to answer the question, as depicted in Figure 1. It is worth noting that FiD effectively highlights salient parts based on the global context to generate the answer regardless of the length of the whole context.

To this end, we propose a novel prompt compression method, ***R**eading **To** **C**ompressing (**R2C**)*, which fully utilizes the structure and training strategy of FiD to align with prompt compression. First, the prompt compression is connected with the FiD to capture the global context of lengthy input. Specifically, lengthy prompts are divided into multiple chunks as input units of FiD, and global semantics over chunks are aggregated in the decoder. R2C yields efficiency by utilizing the cross-attention scores computed in generating only the first token, avoiding auto-regressive generation. Second, we utilize the *question-answering* task to train the compressor, a natural way to identify key information without relying on pseudo-labels. The cross-attention scores are trained to align with a question-answering process, providing the effective approximation of tokens that the target LLM needs to focus on.

We thoroughly conduct experiments on in-domain, *i.e.*, Natural Questions, and out-of-domain datasets, *i.e.*, LongBench, demonstrating the effectiveness and efficiency of R2C. Notably, R2C yields up to 14.5 times faster compression than existing methods and reduces end-to-end latency by 26% with a 5x compression of the original prompt with minimal performance degradation.

2 Related Work

2.1 Prompt Compression

As more complex prompting techniques are proposed to unlock the capabilities of LLMs, *prompt compression* has been actively studied to handle lengthy input. Prompt compression methods are broadly categorized into soft prompting and two token pruning methods which are abstractive and extractive compression.

Soft Prompting. Soft prompting methods compress texts in embedding space, using a limited number of learnable embeddings to imply input tokens (Chevalier et al., 2023; Mu et al., 2023; Qin and Durme, 2023; Cheng et al., 2024). They achieve high compression ratios with minimal loss of semantics. However, the embedding must be learned for each language model, and it is challenging to apply to API-based LLMs.

Abstractive Compression. Abstractive compression aims to generate the core information of a prompt using a generative model. Wang et al. (2023) introduced sentence-level pseudo-labels and trained the model to generate labeled sentences given the original prompt. Chuang et al. (2024) proposed to optimize the compressor using a reward function considering length constraints. Recently, Ali et al. (2024) proposed to construct a graph with LLMs and reconstruct the prompt using subgraphs within the graph. While abstractive compression effectively compresses prompts by reconstructing them, they suffer from the substantial cost of auto-regressive generation.

Extractive Compression. Extractive compression methods extract only core information from prompts. Representative works proposed by Li et al. (2023); Jiang et al. (2023a) remove redundancy based on the entropy-based metric without any training. Jiang et al. (2023b) additionally leverages question-aware perplexity. However, the entropy-based methods are hardly aligned with the objective of prompt compression, which is to retain only the essential information (Ali et al., 2024). Recently, there have been works on training compressors to extract the salient information from the prompt. Pan et al. (2024); Xu et al. (2024) created pseudo labels and Huang et al. (2023); Jung and Kim (2023) incorporated reinforcement learning for training a compressor. However, they still struggle to capture the global context across whole prompts since each segment of prompts is compressed independently.

2.2 Fusion-in-Decoder (FiD)

Fusion-in-Decoder (Izacard and Grave, 2021b) has been introduced for Open-Domain Question Answering (ODQA), effectively aggregating information from multi-documents to generate an answer. Recent studies have demonstrated the versatility of the FiD structure in various tasks thanks to its ability to handle lengthy input without information loss. Izacard and Grave (2021a) showed that the cross-attention score obtained from the FiD decoder can be utilized as a label for retrieval. In addition, Ye et al. (2023) incorporated the FiD structure for in-context learning with long input. This paper introduces a new method for compressing long prompts utilizing FiD and demonstrates its effectiveness.

3 Proposed Method

This section introduces R2C, a novel prompt compression method using the Fusion-in-Decoder (FiD) architecture. In contrast to the existing compression methods, which consider the local context within each chunk, R2C effectively seizes the global context that lies across chunks in the lengthy prompt.

The prompt $\mathrm{P_C}$ consists of an instruction I, a context C, and a question Q, *i.e.*, $\mathrm{P_C} = (\mathrm{I}; \mathrm{C}; \mathrm{Q})$, where the context is usually the longest component. R2C compresses the context C to $\hat{\mathrm{C}}$ and reduces the overall prompt length from $|\mathrm{P_C}|$ to $|\mathrm{P}_{\hat{\mathrm{C}}}|$.

Figure 2 depicts the overall framework of R2C. First, we divide context C into multiple chunks and feed them into FiD to obtain the importance score for each token (Section 3.1). Next, token scores are aggregated in chunk- and sentence-level for multi-granularity compression (Section 3.2). Finally, we hierarchically compress the context in a coarse-to-fine manner, *i.e.*, chunk-to-sentence order (Section 3.3). Note that the training process is omitted in this paper since R2C is not trained for prompt compression. Instead, R2C utilizes the trained FiD weights for the QA task.

3.1 Identifying Importance in Context

We calculate the token-level importance in the context using FiD (Izacard and Grave, 2021b). The context is divided into smaller chunks as input units, and each chunk is processed individually with multiple encoders. The outputs of multiple encoders are concatenated and utilized as a key-value matrix of cross-attention for decoding. Even if the overall length of a prompt exceeds the maximum input length of language models, FiD discerns the important parts considering the global context. Here, we leverage the attention score as the importance criterion.

Given a prompt $\mathrm{P_C}$, we divide the context C into K chunks: $\mathcal{C} = [\mathrm{C}_1, \mathrm{C}_2, ..., \mathrm{C}_K]$. For each chunk C_i, we input it to the FiD-encoder and get the token embeddings $\mathbf{H}_i \in \mathbb{R}^{M \times h}$, where M is the maximum sequence length of the FiD-encoder, and h is the size of the hidden dimension. Similar to the original QA task, we prepend the question Q to C_i if a question is given in the dataset, otherwise, we set Q as an empty string.

$$\mathbf{H}_i = \text{FiD-encoder}\left(\mathrm{Q} + \mathrm{C}_i\right) \in \mathbb{R}^{M \times h}. \quad (1)$$

We perform it for K chunks and concatenate all token embeddings as $\mathbf{H} = (\mathbf{H}_1; \ldots; \mathbf{H}_K) \in \mathbb{R}^{(K \times M) \times h}$. Then, $\mathbf{H}$ is converted to the key matrix $\mathbf{K}$ through the projection layer $\mathbf{W}_k \in \mathbb{R}^{h \times h}$. The cross-attention score $\mathbf{A} \in \mathbb{R}^{1 \times (K \times M)}$ is calculated by the matrix product using query embedding $\mathbf{q}_{\texttt{[BOS]}} \in \mathbb{R}^{1 \times h}$ and the key matrix $\mathbf{K}$.

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{q}_{\texttt{[BOS]}}\mathbf{K}}{\sqrt{h}}\right), \text{where } \mathbf{K} = \mathbf{W}_k\mathbf{H}^\top. \quad (2)$$

Due to the maximum sequence length, existing works (Jiang et al., 2023a,b) are limited to the local context within chunks. In contrast, R2C effectively captures the global context across all chunks by processing concatenated outputs. We define the token-level importance $t_{i,j}$ of the j-th token in i-th chunk as the sum of the attention scores $\mathbf{A}$ over all layers and heads in the FiD-decoder.

$$t_{i,j} = \sum_{l=1}^{L}\sum_{h=1}^{H} \mathbf{A}_{i,j}^{(l,h)}. \quad (3)$$

Here, L is the number of layers in the decoder, and H is the number of heads. $\mathbf{A}_{i,j}^{(l,h)}$ denotes the attention score for the j-th token in the i-th chunk of h-th head in l-th layer.

3.2 Aggregating Unit Importance

We adopt two compression units with coarser granularity than tokens, *i.e.*, chunks and sentences. A naive way of compression is to prune redundant tokens with low importance scores until the desired compression ratio is reached. However, token-level compression neglects the semantic integrity and the grammatical structure of the text (Jiang et al., 2023a; Ali et al., 2024). Otherwise, R2C adopts two

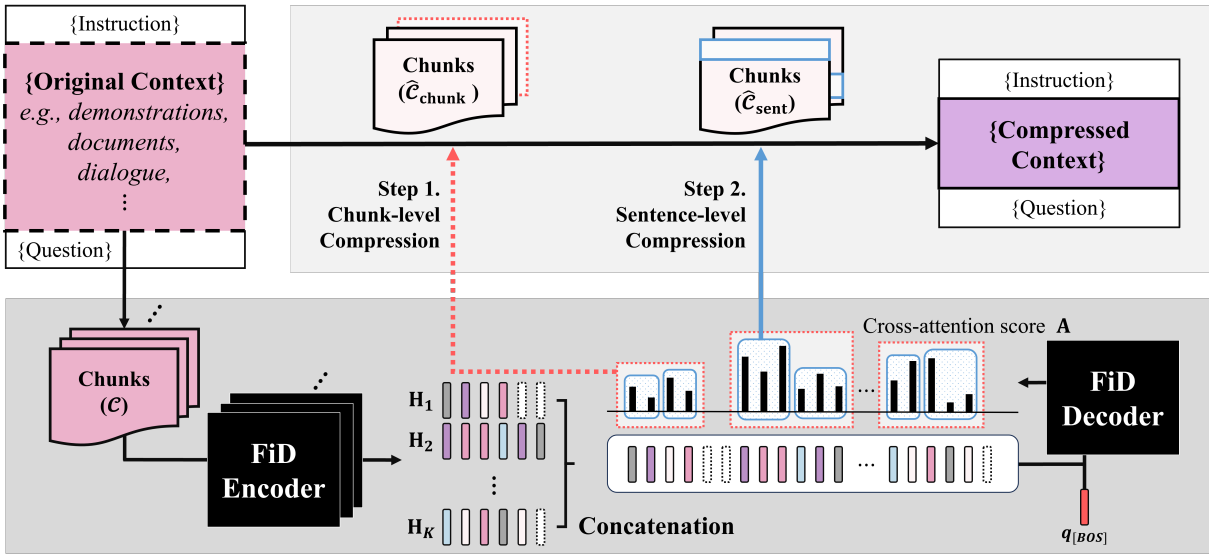


Figure 2: The overall framework of **R**eading **To** **C**ompressing (**R2C**)

levels of granularity, achieving high compression ratios without hindering the semantics.

Chunk-level Importance. We utilize an average pooling to aggregate chunk-level importance following Izacard and Grave (2021a). Note that other pooling operators, *e.g.*, max or sum, are also feasible and please refer to Table 4 for further experimental analysis. The chunk-level importance c_i is averaged over the token-level scores $t_{i,j}$ contained in each chunk C_i.

$$c_i = \frac{1}{|\mathrm{C}_i|} \sum_{j=1}^{|\mathrm{C}_i|} t_{i,j}, \tag{4}$$

where $|\mathrm{C}_i|$ denotes the number of tokens in chunk C_i. We sort the chunks C in descending order of chunk-level scores and re-index them.

Sentence-level Importance. For a fine-grained unit, R2C also utilizes sentence-level importance. A sentence is a basic unit that preserves the meaning of the original input while compression (Wang et al., 2023; Xu et al., 2024).

We split each chunk C_i to the sentence list $\mathcal{S}_i = [\mathrm{S}_{i,1}, \mathrm{S}_{i,2}, \ldots, \mathrm{S}_{i,|\mathcal{S}_i|}]$ using `sent_tokenize` of NLTK (Wagner, 2010). For the m-th sentence in i-th chunk $\mathrm{S}_{i,m}$, we compute the sentence-level importance $s_{i,m}$ as an average over the scores of contained tokens.

$$s_{i,m} = \frac{1}{|\mathrm{S}_{i,m}|} \sum_{j=1}^{|\mathrm{C}_i|} t^m_{i,j}, \tag{5}$$

$$\text{where } t^m_{i,j} = \begin{cases} t_{i,j} & \text{if } j\text{-th token} \in \mathrm{S}_{i,m}, \\ 0 & \text{otherwise.} \end{cases}$$

Here, $|\mathrm{S}^m_i|$ denotes the number of tokens in the sentence S^m_i. Then, similar to chunk-level, we sort and re-index the sentences in $\mathcal{S}_i$ according to sentence-level importance scores.

Token-level Importance. It is possible to utilize token-level importance as the finest granularity. Although we derive the token-level scores in R2C, it is observed that token-level compression may hinder the performance of the target task. Therefore, R2C focuses on utilizing chunk- and sentence-level importance without losing semantic integrity.

3.3 Performing Hierarchical Compression

Given the multi-granularity unit importance, *i.e.*, chunk and sentence, R2C hierarchically compresses the prompt. To preserve semantic integrity (Jiang et al., 2023b; Huang et al., 2023), it performs compression from chunks to sentence excluding token-level. Token-level compression can break the grammatical structure or struggle to generate answers exactly matched to the ground truth[1].

Algorithm 1 describes the compression procedure of R2C. Given sorted chunks $\mathcal{C}$ as input, R2C first performs chunk-level compression and generates $\hat{\mathcal{C}}_{\text{chunk}}$ by retaining only the important chunks (line 1–8). Then, it further performs sentence-level compression and selects only the crucial sentences for each chunk to yield $\hat{\mathcal{C}}_{\text{sent}}$ (line 9–23). We use $\hat{\mathcal{C}}_{\text{sent}}$ for the final compressed context $\hat{\mathrm{C}}$ (line 24),

[1]With token-level compression, "*She moved to Los Angeles, where she studied drama at the Lee Strasberg Theatre and Film Institute*" becomes "*Los Angeles studied drama at the Lee Strasberg Theatre and Film Institute*", destroying the linguistic structure and semantics.

Algorithm 1 R2C prompt compression

Input: sorted context chunks $\mathcal{C} = [\mathrm{C}_1, \mathrm{C}_2, ...\mathrm{C}_K]$, number of chunks K, number of total removing tokens E_{comp}, hierarchical ratio between two-level compression ρ
Output: compressed context $\hat{\mathrm{C}}$

▶ **Step 1. Chunk-level Compression**
1: $E_{\text{chunk}} = \rho \cdot E_{\text{comp}}$
2: **for** $i = 1$ **to** K **do**
3: **if** $E_{\text{chunk}} \geq \sum_{j=i}^{K} |\mathrm{C}_j|$ **then**
4: **Break**
5: **end if**
6: $K' = K' + 1$
7: **end for**
8: $\hat{\mathcal{C}}_{\text{chunk}} \leftarrow [\mathrm{C}_1, \mathrm{C}_2, ...\mathrm{C}_{K'}]$
▶ **Step 2. Sentence-level Compression**
9: Initialize $\hat{\mathcal{C}}_{\text{sent}} \leftarrow [\,]$
10: $E_{\text{sent}} = (1 - \rho) \cdot E_{\text{comp}}$
11: **for** $i = 1$ to K' **do**
12: Calculate $E_{\text{sent},i}$ using Equation (6)
13: $\mathcal{S}_i = \texttt{sent\_tokenize}(\mathrm{C}_i)$
14: Initialize $\hat{\mathcal{S}}_i \leftarrow [\,]$
15: **for** $m = 1$ to $|\mathcal{S}_i|$ **do**
16: **if** $E_{\text{sent},i} \geq \sum_{j=m}^{|\mathcal{S}_i|} |\mathrm{S}_{i,j}|$ **then**
17: **Break**
18: **end if**
19: Add $\mathrm{S}_{i,m}$ to $\hat{\mathcal{S}}_i$
20: **end for**
21: $\hat{\mathrm{C}}_i = \texttt{concatenate}(\hat{\mathcal{S}}_i)$
22: Add $\hat{\mathrm{C}}_i$ to $\hat{\mathcal{C}}_{\text{sent}}$
23: **end for**
24: $\hat{\mathrm{C}} = \texttt{concatenate}(\hat{\mathcal{C}}_{\text{sent}})$

otherwise $\hat{\mathcal{C}}_{\text{chunk}}$ is used if sentence-level compression is not applied. When concatenating $\hat{\mathcal{S}}_i$ to $\hat{\mathcal{C}}_{sent}$ (line 21), we restore them to their original order, except for $\hat{\mathcal{C}}_{sent}$ in Natural Questions due to lost-in-the-middle problem (Liu et al., 2023).

Step 1. Chunk-level Compression. We first determine the hierarchical ratio between two-level compression using a hyperparameter ρ. Given the length of the original prompt be $|\mathrm{P_C}|$ and the number of target tokens T, the number of removing tokens is defined as $E_{\text{comp}} = |\mathrm{P_C}| - T$. We set the number of chunk- and sentence-level compression tokens as $E_{\text{chunk}} = \rho \cdot E_{\text{comp}}$ and $E_{\text{sent}} = (1 - \rho) \cdot E_{\text{comp}}$, respectively. That is, if ρ is 1, R2C only performs chunk-level compression, and if ρ is 0, R2C only performs sentence-level compression. R2C constructs a compressed chunk list $\hat{\mathcal{C}}_{\text{chunk}}$ by adding chunks until the number of removing tokens E_{chunk} is exceeded by the remaining tokens.

Step 2. Sentence-level Compression. R2C adaptively sets the number of compressed sentences for each chunk based on chunk-level importance. It allows R2C to reserve more information in more salient chunks, reflecting the global context. Specifically, given the chunk-level importance $\hat{c}_i$ for the i-th chunk in the compressed chunk list $\hat{\mathcal{C}}_{\text{chunk}}$, we define the number of sentence-level compressed tokens $E_{\text{sent},i}$.

$$E_{\text{sent},i} = \frac{(1/\hat{c}_i)^{\gamma}}{\sum_{\hat{c}_i \in \hat{c}} (1/\hat{c}_i)^{\gamma}} \times E_{\text{sent}}, \tag{6}$$

γ is a coefficient that controls the impact of inverted chunk-level scores. Note that if γ is 0, the sentence compression is applied uniformly to all chunks.

4 Experiments Setup

4.1 Datasets

We validate the performance of R2C on two datasets. (i) **In-domain**: We utilize Natural Questions (NQ) (Kwiatkowski et al., 2019), which are widely adopted in Open-domain Question Answering (ODQA) tasks. We retrieve 20 candidate passages for each question using DPR (Karpukhin et al., 2020; Izacard and Grave, 2021a). (ii) **Out-of-domain**: To evaluate the generalizability of compressed prompts, we use the LongBench (Bai et al., 2023)[2]. We include five types of tasks: single-document QA (SingleDoc), multi-document QA (MultiDoc), summarization (Summ.), few-shot learning (FewShot), and code completion (Code). Note that we only evaluate the English datasets and omit the synthetic tasks to validate the model's ability in real-world scenarios. (See appendix A.1 for further details.)

4.2 Evaluation Metrics and Prompts

For Natural Questions, we use Span Exact Match (Span EM) and prompts following Liu et al. (2023) to evaluate whether the generated text contains the answer. For LongBench, we follow metrics and prompts of each dataset provided by the official benchmark (Refer to appendix A.2 and A.3).

4.3 Baselines

We compared R2C with the following models. (i) Two retrieval-based models: BM25 (Robertson and Walker, 1994) and DPR (Karpukhin et al., 2020), performing only chunk-level compression. DPR (Karpukhin et al., 2020) is trained with knowledge distillation on the NQ dataset (Izacard and Grave, 2021a). For the BM25 results in the LongBench dataset, we follow the experimental setup from LongLLMLingua (Jiang et al., 2023b). The key difference is that we apply BM25 at the chunk-level instead of the sentence-level.

[2] https://github.com/THUDM/LongBench

(ii) Five compression-based models: Selective-Context (Li et al., 2023), LLMLingua (Jiang et al., 2023a), LongLLMLingua (Jiang et al., 2023b), LLMLingua-2 (Pan et al., 2024) and RECOMP (Xu et al., 2024). Unlike other baseline methods, RECOMP compresses the context at the sentence-level by selecting sentences based on the similarity between the question and sentence embeddings. As a result, RECOMP is unsuitable for tasks where a question is absent, such as summarization, and is reported only on QA tasks.

4.4 Implementation Details

We trained the Fusion-in-Decoder model (Izacard and Grave, 2021b) on the Natural Questions train set, utilizing 20 passages for each question. For target LLMs, we used Llama2-7b-chat-hf [3] (LLaMA2-7B, Touvron et al., 2023) and GPT-3.5-turbo-1106 [4] (GPT-3.5). We randomly sampled 20% of the NQ dev set to tune the compression hyper-parameters in R2C. We set the hierarchical ratio ρ and the chunk-level score coefficient γ as 0.8 and 1, respectively. All token counts are based on the ChatGPT tokenizer. We regard a demonstration, paragraph, or dialogue as a chunk unit if given. For longer chunks, we split them based on line breaks and set the maximum token length for each chunk as 128. We reproduced baselines based on the official codes and reported with the results from the original papers.

5 Results and Analysis

5.1 Main Results

Question Answering Task. Table 1 reports the accuracy of FiD and two target LLMs, namely GPT-3.5 and LLaMA2-7B, with different compression methods on the Natural Questions test set. (i) The proposed method achieves the best performance compared to state-of-the-art compression methods. Specifically, it shows improvements of 5.6% and 11.1% over the most effective baseline, RECOMP (Xu et al., 2024), when GPT-3.5 and LLaMA2-7B are used as target LLMs, respectively. Notably, R2C outperforms the original prompt when used with LLaMA2, and with GPT-3.5, we achieve comparable performance with six times fewer tokens. (ii) DPR (Karpukhin et al., 2020), RECOMP (Xu et al., 2024), and R2C, which are trained on question-answer datasets, *i.e.*, in-domain evaluation, significantly outperform other methods. While the other two methods are optimized to identify significant passages or sentences, R2C yields better performance. This suggests that learning to answer questions directly contributes to capturing important information in context, and highlights the benefits of using cross-attention scores for compression after QA training. (iii) Despite the relatively modest size of FiD (223M) in comparison to LLaMA-2 (7B), it performs well on the QA task, with only a 4.5% difference in accuracy. This indirectly suggests that FiD is effective at capturing important information within multiple documents.

Target LLM	Compression	NQ test (Span EM)	# tokens
FiD	-	50.5	-
GPT-3.5	Original	66.7	3,018
	BM25	49.4	534
	DPR	<u>63.0</u>	501
	Selective-Context	44.4	501
	LLMLingua	41.9	478
	LLMLingua-2	52.1	510
	LongLLMLingua	55.6	489
	RECOMP	<u>63.0</u>	500
	R2C (Ours)	**66.5**	482
LLaMA2 -7B	Original	52.8	3,018
	BM25	41.8	534
	DPR	<u>54.3</u>	501
	Selective-Context	38.1	501
	LLMLingua	32.7	478
	LLMLingua-2	42.5	510
	LongLLMLingua	49.0	489
	RECOMP	53.7	500
	R2C (Ours)	**59.7**	482

Table 1: Accuracy of FiD and LLaMA2-7B in NQ test with 6x compressed prompt (*i.e.*, $T = 500$). For context C of prompt P_C, we used 20 passages retrieved by DPR (2020; 2021a). The best and the second-best performance using the compressed prompt are marked in **bold** and <u>underline</u>, respectively.

Out-of-domain Evaluation. Table 2 shows an accuracy of R2C and other compression methods with two target LLMs on out-of-domain datasets in LongBench (Bai et al., 2023)[5]. Note that RECOMP selects sentences based on their similarity to the question embedding, so it was only evaluated on the QA task. We summarize our key observations as follows:

(i) R2C yields the highest average performance

[3] https://huggingface.co/meta-llama

[4] https://chatgpt.com/

[5] While the official results without compression (*i.e.*, Original and Original*) are presented, they are not fully evaluated. This is because the middle part has been truncated if samples exceed the maximum sequence length.

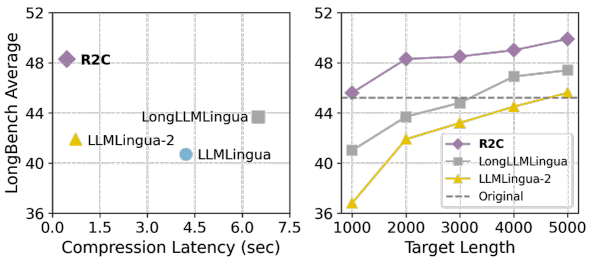
Target LLM	Compression	SingleDoc	MultiDoc	Summ.	FewShot	Code	Average	# tokens
GPT-3.5	Original	43.2	46.1	25.2	69.2	64.4	49.6	9,881
	Orignal*	39.7	38.7	26.5	67.0	54.2	45.2	9,881
	BM25	34.9	41.0	23.3	68.1	49.6	43.4	1,949
	LLMLingua	30.4	31.4	20.9	66.0	55.0	40.7	1,830
	Selective-Context	29.8	35.4	22.1	52.4	45.0	36.9	2,009
	LLMLingua-2	36.2	40.9	23.2	61.5	47.6	41.9	2,023
	LLMLingua-2*	29.8	33.1	25.3	66.4	**58.9**	42.7	≈1,898
	LongLLMLingua	37.0	44.9	22.0	65.1	49.4	43.7	1,743
	LongLLMLingua*	39.0	42.2	**27.4**	**69.3**	56.6	46.9	≈1,753
	RECOMP	40.1	48.1	-	-	-	-	-
	R2C (ours)	**43.5**	**48.7**	24.9	66.9	57.6	**48.3**	1,976
LLaMA2-7b	Original	25.6	22.4	24.6	62.9	55.2	38.1	9,881
	Original*	24.9	22.6	24.7	60.0	48.1	36.1	9,881
	BM25	25.4	25.4	25.0	61.5	44.0	36.3	1,949
	LLMLingua	19.4	17.3	22.3	61.0	51.0	34.2	1,830
	Selective-Context	21.0	19.7	23.5	46.1	34.0	28.9	2,009
	LLMLingua-2	22.6	23.8	23.8	56.0	43.2	33.9	2,023
	LongLLMLingua	26.9	29.6	23.4	61.3	43.9	37.0	1,743
	RECOMP	27.4	**31.3**	-	-	-	-	-
	R2C (ours)	**28.5**	29.8	**25.3**	**64.4**	**54.0**	**40.4**	1,976

Table 2: Performance of two LLMs (GPT-3.5 and LLaMA2-7B) in LongBench benchmark with 5x compressed prompt (*i.e.*, $T = 2,000$). # tokens denotes the average number of tokens across all datasets based on the ChatGPT tokenizer. Among the compression methods, the best performance is marked in **bold** and the second-best is underlined. * denotes the result from (Bai et al., 2023; Jiang et al., 2023b; Pan et al., 2024).

for both target LLMs with improvements of 3.0% in GPT-3.5 and 9.2% in LLaMA2-7B over the best competitive baseline. It also outperforms the original prompt in two QA tasks, SingleDoc and MultiDoc, indicating the effectiveness of removing the ambiguity of lengthy prompts using R2C. (ii) The question-answering task proves to be an effective alternative for compressor training. R2C shows superior performance on all tasks, including QA tasks. We also report the performance of each model as reported in their respective papers. Inconsistencies in the results are likely due to differences in the version of GPT-3.5. Focusing on the results without an asterisk (-), which we reproduced using the same version, GPT-3.5-turbo-1106, R2C achieved the highest performance among compression-based methods. Additionally, we observed that R2C performed well on the code completion task, likely because generating code from multiple files is analogous to generating answers from multiple documents. (iii) We observe different trends depending on the target LLM for the FewShot task. Specifically, on GPT-3.5, chunk-level filtering with BM25 outperforms R2C, achieving 68.1 compared to 66.9. However, on LLaMA2-7b, R2C significantly surpasses BM25, with scores of 64.5 and 61.5, respectively. It suggests that GPT-3.5 can understand the task even when the demonstration is not directly relevant, highlighting the importance of using the intact demonstration. R2C, by compressing hierarchically, may disrupt the demonstration. We conjecture that different prompts may require different levels of granularity for effective compression.

5.2 In-depth Analysis

Compression Efficiency. Figure 3-(a) presents the efficiency of R2C and baseline methods on LongBench. We measure the latency using a single NVIDIA A6000 GPU with a batch size of 1. The y-axis represents the average accuracy of the tasks included in LongBench, and the x-axis represents the average compression latency per prompt. Results are based on $T = 2000$ and GPT-3.5 as the target LLM. Notably, R2C dramatically improves efficiency while also enhancing accuracy, achieving a latency improvement of 1.6 times over the most efficient existing method, *i.e.*, LLMLingua-2, and 14.5 times over the most effective method, *i.e.*, LongLLMLingua. This improvement in efficiency is largely due to the smaller model size used for compression. Specifically, R2C is based on T5-base (Raffel et al., 2020), which has 223M parameters, while LLMLingua-2 adopts XLM-RoBERTa-large (Conneau et al., 2020) with 355M parame-



(a) (b)

Figure 3: Performance of GPT-3.5 with various compression methods in LongBench. (a): compression effectiveness-efficiency comparison. (b): effectiveness over varying compression ratios (2x–10x).

Compression	Comp. latency	API latency	E2E latency	E2E latency in %
-	0s	1.52s	1.52s	100.0%
R2C (5x)	0.45s	0.88s	1.33s	87.5%
R2C (10x)	0.44s	0.68s	1.11s	74.0%

Table 3: End-to-end efficiency of R2C on LongBench dataset. Comp. latency indicates the latency for compressing prompts. E2E denotes the latency from the prompt compression to the black-box API (GPT-3.5). Note that latency is in seconds.

ters. LongLLMLingua and LLMLingua use a large LLM (Touvron et al., 2023) with 7B parameters, resulting in significant computational overhead. Additionally, although R2C uses a generative model, it efficiently captures important information by utilizing only the cross-attention scores from the first token generation. The results demonstrate that R2C enhances the quality of compressed prompts and optimizes the compression efficiency.

End-to-end Efficiency. Table 3 illustrates the end-to-end latency of R2C depending on the different number of target tokens. We use 3,350 samples in LongBench datasets and set the maximum decoding token of the API to 200 for all datasets following (Jiang et al., 2023b). Although token pruning can be used in API-based models and reduce API costs, the benefits can be offset if the compression process takes too long. Notably, the overall end-to-end inference time is accelerated with the proposed method. As shown in Figure 3-(a), the compression is efficiently performed by R2C. Considering the latency of both compression and the API, R2C can infer the answer with 74.0% latency compared to the original prompt.

Varying Target Length. Figure 3-(b) shows the average performance on LongBench according to the length of compressed prompts. We adjusted

Variants	NQ dev	# tokens
R2C	**58.44**	483
R2C w/ chunk only	57.01	485
R2C w/ sentence only	57.58	480
R2C w/ tokens only	51.86	478
T5-base initialize	44.88	483
w/ decoder last layer only	56.73	483
unit aggregation: max	58.39	483
unit aggregation: sum	57.58	482

Table 4: Ablation study of R2C on the NQ dev dataset. The results are based on LLaMA2-7B with 6x compressed prompts.

the compression ratio from 2x to 10x (*i.e.*, 5K–1K tokens) for LongBench, which averages about 10K tokens originally. At all compression ratios, R2C delivers higher performance compared to the original prompt, indicating its ability to eliminate noisy information. However, LongLLMLingua exhibits lower performance at high compression ratios. Unlike R2C, LongLLMLingua employs entropy-based metrics, struggling with retaining important information at high compression ratios. Since the two QA tasks show different tendencies compared to the other tasks, we also report the full results for each task of R2C with different lengths in Table 5 in the appendix. The results show that the QA tasks, including SingleDoc and MultiDoc, achieve their best performance when the target length is in the range of 2000 to 3000 tokens, while the performance of the other tasks continues to improve as the target length increases. These results show that R2C not only outperforms other models due to its strong performance on QA tasks, but also excels at compression across a variety of tasks, resulting in superior generalization capabilities. Furthermore, this suggests that the performance of the target LLM can be further improved by first denoising the input contexts and then feeding them into the target model.

5.3 Ablation Study

Comparison with variants of R2C. Table 4 analyzes the effectiveness of various strategies utilized in R2C. (i) R2C with token-level compression drastically degrades the performance of R2C. This indicates that since it focuses only on prominent tokens while neglecting semantic consistency, it easily confuses the target LLM, as shown in Section 3.3. (ii) R2C without the QA task training, *i.e.*, T5-base (Raffel et al., 2020) initialization, shows a

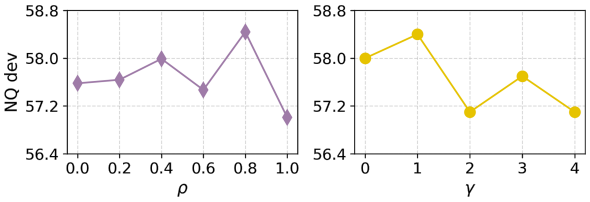


Figure 4: Performance of LLaMA2-7B with R2C on the NQ dev dataset adjusting (a) the hierarchical ratio ρ and (b) importance coefficient γ.

significant performance drop of 23.2%. It implies that training the compressor with the QA task is appropriate as it naturally learns to discern salient parts in lengthy inputs. (iii) While the last decoder layer can focus on generating the final answer tokens, using cross-attention scores from all layers contributes to a performance gain. We attribute this to the tendency of the last decoder layer to focus excessively only on certain parts. (iv) Lastly, R2C aggregates unit importance by averaging token-level scores within a chunk or sentence. The results imply that average pooling is slightly more effective while using maximum token importance is valid. We adopt averaging as it captures crucial information across the entire unit and can provide a more balanced and comprehensive representation rather than focusing solely on the most prominent tokens.

Effectiveness of Hierarchical Compression. Figure 4 illustrates the impact of hierarchical compression parameters (*i.e.*, ρ and γ) on the NQ dev set. Figure 4-(a) illustrates that the balance of chunk and sentence-level compression by $\rho = 0.8$ yields the best performance, indicating that it can preserve essential information more effectively by considering both coarse- and fine-grained levels. Figure 4-(b) depicts the effect of γ in eq (6). Higher γ removes more tokens from less important chunks. While $\gamma = 0$ treats all chunks equally, using chunk-level importance in sentence-level compression improves performance by tailoring compression to the importance of each chunk. However, high γ values reduce performance, suggesting that aggressive sentence-level compression on low-scored chunks harms overall compression quality.

6 Conclusion

In this work, we explored the capabilities of multi-document readers for prompt compression and successfully bridged two different lines of research. We propose a novel prompt compression method, namely *Reading To Compressing* (R2C), leveraging the Fusion-in-Decoder. R2C effectively captures global context and identifies salient information across multiple segments. Furthermore, by training the compressor using question-answering datasets, the most influential tokens are identified without using noisy pseudo-labels for compressed prompts. Experimental results demonstrate that R2C outperforms existing prompt compression methods by preserving semantic integrity and even surpasses uncompressed prompts by removing ambiguity.

7 Limitations

Task Generalization. While training QA captures important context across various benchmarks, it remains challenging to generalize the capability to all tasks. Prompt compression requires understanding both instructions and the entire prompt. Recent work (Ye et al., 2023) validates the FiD structure in in-context learning and suggests excluding redundant questions. Given FiD's efficiency in capturing global context, further research, including diverse training strategies, remains to be explored.

Dynamic Compression Ratios and Granularity. R2C sets a target number of tokens for prompt compression, providing intuitive usability. However, each prompt has an optimal compression ratio, and using fewer tokens can remove noise and enhance performance in some cases. It is necessary to adjust the length of prompts dynamically based on each prompt. Additionally, the appropriate compression granularity, *e.g.*, chunk, sentence, phrase, or token, should be adjustable. Even when compressing the same number of tokens, determining the appropriate granularity for each prompt and compressing accordingly can ensure that the most relevant information is retained, potentially improving performance across various tasks.

Ethics Statement

This work fully respects ACL's ethical guidelines. We have utilized scientific resources available for research under liberal licenses, and our use of these tools is consistent with their intended applications.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II190421, RS-2022-II220680, RS-2022-II221045)

References

Muhammad Asif Ali, Zhengping Li, Shu Yang, Keyuan Cheng, Yang Cao, Tianhao Huang, Lijie Hu, Lu Yu, and Di Wang. 2024. PROMPT-SAW: leveraging relation-aware graphs for textual prompt compression. *CoRR*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. xrag: Extreme context compression for retrieval-augmented generation with one token. *CoRR*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *EMNLP*, pages 3829–3846.

Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. 2024. Learning to compress prompt in natural language formats. *CoRR*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *ACL*, pages 8440–8451.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey for in-context learning. *CoRR*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR*.

Xijie Huang, Li Lyna Zhang, Kwang-Ting Cheng, and Mao Yang. 2023. Boosting LLM reasoning: Push the limits of few-shot learning with reinforced in-context pruning. *CoRR*.

Gautier Izacard and Edouard Grave. 2021a. Distilling knowledge from reader to retriever for question answering. In *ICLR*.

Gautier Izacard and Edouard Grave. 2021b. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, pages 874–880.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023a. Llmlingua: Compressing prompts for accelerated inference of large language models. In *EMNLP*, pages 13358–13376.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *CoRR*.

Hoyoun Jung and Kyung-Joong Kim. 2023. Discrete prompt compression with reinforcement learning. *CoRR*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*, pages 6769–6781.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, pages 452–466.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, pages 3045–3059.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *EMNLP*, pages 6342–6353.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *CoRR*.

Jesse Mu, Xiang Li, and Noah D. Goodman. 2023. Learning to compress prompts with gist tokens. In *NeurIPS*.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *CoRR*.

Guanghui Qin and Benjamin Van Durme. 2023. Nugget: Neural agglomerative embeddings of text. In *ICML*, pages 28337–28350.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew M. Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, James Molloy, Jilin Chen, Michael Isard, Paul Barham, Tom Hennigan, Ross McIlroy, Melvin Johnson, Johan Schalkwyk, Eli Collins, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Clemens Meyer, Gregory Thornton, Zhen Yang, Henryk Michalewski, Zaheer Abbas, Nathan Schucher, Ankesh Anand, Richard Ives, James Keeling, Karel Lenc, Salem Haykal, Siamak Shakeri, Pranav Shyam, Aakanksha Chowdhery, Roman Ring, Stephen Spencer, Eren Sezener, and et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, pages 232–241.

Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. 2020. Intellicode compose: code generation using transformer. In *ESEC/FSE*, pages 1433–1443.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*.

Wiebke Wagner. 2010. Steven bird, ewan klein and edward loper: Natural language processing with python, analyzing text with the natural language toolkit - o'reilly media, beijing, 2009, ISBN 978-0-596-51649-9. *Lang. Resour. Evaluation*, 44:421–424.

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md. Rizwan Parvez, and Graham Neubig. 2023. Learning to filter context for retrieval-augmented generation. *CoRR*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. RECOMP: improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

Qinyuan Ye, Iz Beltagy, Matthew E. Peters, Xiang Ren, and Hannaneh Hajishirzi. 2023. Fid-icl: A fusion-in-decoder approach for efficient in-context learning. In *ACL*, pages 8158–8185.

A Additional Experimental Setup

A.1 Dataset

Natural Questions (Kwiatkowski et al., 2019). The dataset contains Wikipedia articles and questions corresponding to Google search queries. We split the dataset into train, dev, and test sets following Izacard and Grave (2021b). We further sample 1,757 dev samples for experiments, as shown in Table 6.

LongBench (Bai et al., 2023). The dataset is a multi-task benchmark for long context comprising 21 datasets across 6 task categories in both English and Chinese. The task types cover essential long-text application scenarios including single-document QA, multi-document QA, summarization, few-shot learning, code completion, and synthetic tasks. We examine the effectiveness of the proposed method only on *English real-world datasets*, excluding Chinese datasets and synthetic tasks, as listed in Table 6.

A.2 Evaluation Metrics

Natural Questions. We adopt a Span Exact Match (Span EM) for an evaluation metric, aligned with previous works (Lester et al., 2021; Liu et al., 2023; Jiang et al., 2023b). Span EM measures whether the answer is part of the generated answer of the target LLM, effectively reflecting the performance of LLMs on QA datasets rather than mere Exact Match (EM).

LongBench. The metric of each dataset is shown in Table 6. We follow the official metrics which are consistent with the original work. For NarrativeQA, Qasper, MultiFieldQA, HotpotQA, 2WikiMultihopQA, MuSiQue, and TriviaQA, we adopt the F1 score as a metric. We use the Rouge-L score (Lin, 2004) for GovReport, QMSum, MultiNews, and SAMSum, which are widely adopted in summarization tasks. For the TREC dataset, the classification accuracy is measured. Lastly, Edit Sim (Svyatkovskiy et al., 2020) (Levenshetein distance) is used for LCC and RepoBench-P, which is popularly used in code generation evaluation.

A.3 Evaluation Prompts

Table 7 describes the evaluation prompts used for generating answers with target LLMs. We follow the prompt of Liu et al. (2023) and Bai et al. (2023) for Natural Questions and LongBench, respectively.

B Case Study

Figure 5, 6, 7, 9, 8, and 10 showcase case studies of how R2C effectively compresses lengthy prompts. We set the target length T to 500 and 1,000 tokens for the Natural Questions and LongBench datasets, respectively. As demonstrated in the case studies, R2C successfully captures and retains the essential information from long inputs. By employing chunk- and sentence-level compression, R2C preserves the semantic integrity of the prompts, enabling the target LLM to better understand and respond to the prompts. These examples highlight the ability of R2C to efficiently compress prompts without sacrificing crucial details necessary for generating correct answers.

Natural Questions and the SingleDoc task require finding the specific part of a document relevant to the question from multiple documents or a very long source document. In Figure 5, R2C gives a high score to "*Linda Davis*" out of 20 documents and finds the part that says she sang "*Does He Love You*" from the document. Also, in a long document of 9,845 tokens (Figure 6), R2C gives a high score to the sentence that he worked in "*3D printing and software development*" in "*Tennessee*".

The MultiDoc task requires referencing multiple parts to generate accurate responses, making it difficult to compress due to the distribution of essential information. In Figure 7, R2C captures the global context and retains documents that contain key information, such as references to "*Night of Dark Shadows*" and "*alcohol*".

The Summarization task requires generating a short summary from a long original text. Figure 8 shows an example from the GovReport dataset, generating a one-page summary from a 3,510 token long report. We can see that R2C preserves important information, such as the "*eligibility*" and "*coverage*" of the PSOB.

The FewShot Task provides several relevant demonstrations to the current question and asks the LLM to generate an answer by referring to them. Figure 9 illustrates the task of summarizing the dialogue in which "*Jones and Angelina*" make an "*appointment*". In this example, R2C gives high scores to conversations that remind "*Audrey and Tom*" of a previous "*appointment*."

The Code Completion task predicts the next line of the currently given code. To solve this task, it is necessary to reference multiple code files relevant to the current code. In Figure 10, the task

Target LLM	Compression	T	SingleDoc	MultiDoc	Summ.	FewShot	Code	Avg.	# toks
GPT-3.5	Original	-	43.2	46.1	**25.2**	69.2	**64.4**	49.6	9,881
	R2C	1,000	41.8	45.4	24.1	64.9	51.9	45.6	1,048
		2,000	43.5	**48.7**	24.9	66.9	57.6	48.3	1,976
		3,000	43.7	46.9	24.9	67.1	59.8	48.5	2,832
		4,000	43.6	47.5	25.0	68.3	60.9	49.0	3,603
		5,000	**44.4**	47.8	25.1	**69.9**	62.5	**49.9**	4,305

Table 5: Performance of the GPT-3.5 on the LongBench benchmark with R2C, varying the target tokens T. # tokens denotes the average number of tokens across all datasets based on the ChatGPT tokenizer. Since some prompts with lengths shorter than the target tokens are not compressed, the average number of tokens may be less than the target tokens T. The best performance is marked in **bold**.

is predicting the next line of "`owningAccount = getEucalyptusAccount() ;`", and we can observe that R2C gives a high sentence-level score to "`owningAccount = getEucalyptusAccount();`" and "`adminUser = getEucalyptusAdmin();`" among the existing code.

Dataset	Source	Avg len	Metric	# samples
Question Answering				
Natural Questions (NQ)	Wikipedia	3,018	Span EM	79,168/1,757/3,610
Single-document QA (SingleDoc)				
NarrativeQA	Literature, Film	29,872	F1	200
Qasper	Science	5,090	F1	200
MultiFieldQA-en	Multi-field	6,988	F1	150
Multi-document QA (MultiDoc)				
HotpotQA	Wikipedia	12,867	F1	200
2WikiMultihopQA	Wikipedia	7,188	F1	200
MuSiQue	Wikipedia	15,650	F1	200
Summarization (Summ.)				
GovReport	Government report	10,276	Rouge-L	200
QMSum	Meeting	13,917	Rouge-L	200
MultiNews	News	2,642	Rouge-L	200
Few-shot Learning (FewShot)				
TREC	Web	6,785	Accuracy	200
TriviaQA	Wikipedia	11,799	F1	200
SAMSum	Dialogue	9,173	Rouge-L	200
Code Completion (Code)				
LCC	Github	3,179	Edit Sim	500
RepoBench-P	Github	10,826	Edit Sim	500

Table 6: Detailed statistics of Natural Questions (Kwiatkowski et al., 2019) and LongBench (Bai et al., 2023). For NQ, we split data into train, dev, and test set following Izacard and Grave (2021b). For LongBench, we only include English datasets and omit a synthetic task for the evaluation in real-world settings. 'Source' denotes the original source of the context. 'Avg len' is the average token length of each dataset computed by the GPT-3.5 tokenizer. Note that we exclude the system message when counting the Avg len for the NQ dataset.

Dataset		Prompts
Natural Questions		<s>[INST] «SYS» You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Please ensure that your responses are socially unbiased and positive in nature. If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information. «/SYS» Write a high-quality answer for the given question using only the provided search results (some of which might be irrelevant). `{compressed_context}` Question: `{question}` Answer: [/INST]
LongBench	NarrativeQA (SingleDoc)	You are given a story, which can be either a novel or a movie script, and a question. Answer the question as concisely as you can, using a single phrase if possible. Do not provide any explanation. Story: `{compressed_context}` Now, answer the question based on the story as concisely as you can, using a single phrase if possible. Do not provide any explanation. Question: `{question}` Answer:
	Qasper (SingleDoc)	You are given a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write "unanswerable". If the question is a yes/no question, answer "yes", "no", or "unanswerable". Do not provide any explanation. Article: `{compressed_context}` Answer the question based on the above article as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write "unanswerable". If the question is a yes/no question, answer "yes", "no", or "unanswerable". Do not provide any explanation. Question: `{question}` Answer:
	MultiFieldQA-en (SingleDoc)	Read the following text and answer briefly. `{compressed_context}` Now, answer the following question based on the above text, only give me the answer and do not output any other words. Question: `{question}` Answer:
	HotpotQA (MultiDoc)	Answer the question based on the given passages. Only give me the answer and do not output any other words. The following are given passages. `{compressed_context}` Answer the question based on the given passages. Only give me the answer and do not output any other words. Question: `{question}` Answer:
	2WikiMultihopQA (MultiDoc)	Answer the question based on the given passages. Only give me the answer and do not output any other words. The following are given passages. `{compressed_context}` Answer the question based on the given passages. Only give me the answer and do not output any other words. Question: `{question}` Answer:
	MuSiQue (MultiDoc)	Answer the question based on the given passages. Only give me the answer and do not output any other words. The following are given passages. `{compressed_context}` Answer the question based on the given passages. Only give me the answer and do not output any other words. Question: `{question}` Answer:
	GovReport (Summ.)	You are given a report by a government agency. Write a one-page summary of the report. Report: `{compressed_context}` Now, write a one-page summary of the report. Summary:
	QMSum (Summ.)	You are given a meeting transcript and a query containing a question or instruction. Answer the query in one or more sentences. Transcript: `{compressed_context}` Now, answer the query based on the above meeting transcript in one or more sentences. Query: `{question}` Answer:
	MultiNews (Summ.)	You are given several news passages. Write a one-page summary of all news. News: `{compressed_context}` Now, write a one-page summary of all the news. Summary:

	TREC (FewShot)	Please determine the type of the question below. Here are some examples of questions.
		`{compressed_context}`
		`{question}`

	TriviaQA (FewShot)	Answer the question based on the given passage. Only give me the answer and do not output any other words. The following are some examples.
		`{compressed_context}`
		`{question}`

	SAMSum (FewShot)	Summarize the dialogue into a few short sentences. The following are some examples.
		`{compressed_context}`
		`{question}`

	LCC (Code)	Please complete the code given below. `{compressed_context}` Next line of code:
	RepoBench-P (Code)	Please complete the code given below. `{compressed_context}` `{question}` Next line of code:

Table 7: Prompts used for the target LLM on Natural Questions (Kwiatkowski et al., 2019) and LongBench (Bai et al., 2023), following Liu et al. (2023) and Bai et al. (2023). `{compressed_context}` and `{question}` denote the compressed context and question, respectively.

Case study of Natural Questions

R2C Compressed Prompt with $T = 500$ (481 tokens, Original length: 3,038 tokens)

<s>[INST] «SYS»
You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Please ensure that your responses are socially unbiased and positive in nature. If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.
«/SYS»
Write a high-quality answer for the given question using only the provided search results (some of which might be irrelevant).

Document [2](Title: Linda Davis) Linda Davis Linda Kaye Davis (born November 26, 1962) is an American country music singer. Before beginning a career as a solo artist, she had three minor country singles in the charts as one half of the duo Skip & Linda. Her highest chart entry is "Does He Love You", her 1993 duet with Reba McEntire, which reached number one on the "Billboard" country charts and won both singers the Grammy for Best Country Vocal Collaboration.
Document [1](Title: Does He Love You) Does He Love You "Does He Love You" is a song written by Sandy Knox and Billy Stritch, and recorded as a duet by American country music artists Reba McEntire and Linda Davis. It is one of country music's several songs about a love triangle.
Document [5](Title: Linda Davis) Linda Davis Linda Kaye Davis (born November 26, 1962) is an American country music singer. Before beginning a career as a solo artist, she had three minor country singles in the charts as one half of the duo Skip & Linda.
Document [4](Title: Greatest Hits Volume Two (Reba McEntire album)) (then a background singer in McEntire's road band), was the first single and turned out to be a smash. "Does He Love You" is the first of three duets featuring Reba and Linda Davis.
Document [10](Title: Does He Love You) on Patti LaBelle's album, "Flame". It then shows Linda as a professional actress at a movie premiere, then cuts back to Reba smashing the picture with a
Document [11](Title: Shoot for the Moon (album)) Tim DuBois, president of Arista's Nashville division, said that "we rushed it" after Davis won a Grammy Award for her duet wirh Reba McEntire on "Does He Love You".
Document [7](Title: Reba McEntire) Out of the ten tracks were two new singles: the first, "Does He Love You", was a duet with

Question: who sings does he love me with reba
Answer: [/INST]

Prediction (Llama2-7b-chat-hf)
Based on the search results provided, the answer to the question "Who sings 'Does He Love Me' with Reba" is Linda Davis. According to Document [1](Title: Does He Love You), the song "Does He Love You" was written by Sandy Knox and Billy Stritch, and recorded as a duet by American country music artists Reba McEntire and Linda Davis. Therefore, Linda Davis is the singer who collaborates with Reba

Ground truth
Linda Davis

Figure 5: Case study on the Natural Questions development set. The number of tokens is calculated using a ChatGPT tokenizer excluding system messages. The purple colorbox indicates the core information to generate answers.

Case study of SingleDocQA task in LongBench

R2C Compressed Prompt with $T = 1,000$ (1,012 tokens, Original length: 9,845 tokens)
Read the following text and answer briefly.

My Aspergers Child: COMMENTS & QUESTIONS [for Feb., 2017]
I emailed you a while back and you mentioned that I could email when I needed to. But the attorney, even though he was just vaguely familiar with Aspergers, has been very good with Craig. He has the compassion and excellence that is needed here.

⋮

It's been 2 years now and he will not accept his diagnosis. I've tried telling him that it's not a bad thing, that there's been many, many very successful
It would have been so much easier to mention to my adult son that I think (I know he does, but want to ease into the subject)
he has Asperger's when we were living together two years ago. He has since moved to Tennessee working in his field of interest
which is 3-D printing and software development. I am so happy for him that he has found his way into a job that he truly enjoys
even though he's socially isolated.
He's not diagnosed and does not know he has it. How I know is his classic symptoms being sensory issues (fabric feeling like sandpaper)
I wanted to let you know about a research opportunity for children, teens, and young adults with autism.
to manuel labor type jobs (which is not something he enjoys but he did it anyway).
At 19 1/2 he left to serve a 2 year full-time mission for our church. He completed his mission successfully. (I don't think it was without some struggle, stress and depression, but he was able to pick himself up and move on from those times).

⋮

I hope you'll agree it shows that starting work in the industry takes dedication and skill and that becoming a game designer isn't just a fly-by-night job!

Now, answer the following question based on the above text, only give me the answer and do not output any other words.

Question: What field does Danny work in in Tennessee?
Answer:

Prediction (GPT-3.5-turbo-1106)
3-D printing and software development.

Ground truth
3-D printing and software development.

Figure 6: Case study on the MultiFieldQA_en dataset (SingleDoc) in the LongBench benchmark. The purple colorbox indicates the core information to generate answers.

Case study of MultiDocQA task in LongBench

R2C Compressed Prompt with $T = 1,000$ (986 tokens, Original length: 4,137 tokens)

Answer the question based on the given passages. Only give me the answer and do not output any other words.

The following are given passages.

⋮

Using aseptic packaging equipment, products can be packed in aseptic packaging. Pasteurized or UHT treated products packed into this format can be "shelf-stable", requiring no refrigeration.

Passage 3:

Bagman (disambiguation)

A bagman or bag man is a collector of dirty money for organized crime.

The Bag Man, a 2014 film

Bag Man (podcast), a 2018 podcast about Spiro Agnew's 1973 bribery and corruption scandal

"Bagman" (Better Call Saul), an episode of the television series Better Call Saul

See also

Bağban (disambiguation)

Bag boy (disambiguation)

Bag lady (disambiguation)

(1945 film), a 1945 American film starring Fred Allen

Passage 7: The Bag Man

Purcell, Robert De Niro, and Sticky Fingaz. The film premiered on February 28, 2014, in New York and Los Angeles.

Plot

Brutal gangster Dragna recruits professional killer Jack to pick up a bag and wait for his arrival at a motel. Dragna stresses that Jack is not to open the bag or allow anyone to view its contents under any circumstances. Confused as to why

⋮

Passage 8:

Una prostituta al servizio del pubblico e in regola con le leggi dello stato

Una prostituta al servizio del pubblico e in regola con le leggi dello stato (literally "A prostitute serving the public and complying with the laws of the state", also known as Prostitution Italian Style) is a 1970 Italian comedy-drama film written and directed by Italo Zingarelli.For her performance Giovanna Ralli won the Grolla d'oro for best actress.

⋮

Now, answer the following question based on the above text, only give me the answer and do not output any other words.

Question: Which film came out first, Una Prostituta Al Servizio Del Pubblico E In Regola Con Le Leggi Dello Stato or The Bag Man?

Answer:

Prediction (GPT-3.5-turbo-1106)

Una Prostituta Al Servizio Del Pubblico E In Regola Con Le Leggi Dello Stato

Ground truth

Una Prostituta Al Servizio Del Pubblico E In Regola Con Le Leggi Dello Stato

Figure 7: Case study on the 2WikiMultihopQA dataset (MultiDoc) in the LongBench benchmark. The purple colorbox indicates the core information to generate answers.

Case study of Summarization task in LongBench

R2C Compressed Prompt with $T = 1,000$ (1,007 tokens, Original length: 3,510 tokens)
You are given a report by a government agency. Write a one-page summary of the report.
Report:
$\cdots$ There is no minimum amount of time a person must have served to be eligible for benefits. To be eligible for PSOB benefits as a law enforcement officer, firefighter, or chaplain, a person must have served in a "public agency" in an official capacity, with or without compensation. For the purposes of PSOB eligibility, a public agency is defined as the federal government and any department, agency, OB eligibility, the definition of firefighter includes both professional firefighters and persons serving as an "officially recognized or designated member of a legally organized volunteer fire department." A chaplain is eligible for PSOB benefits $\cdots$ To qualify for coverage under the PSOB program, a public safety officer's disability or death must have been the result of a personal injury. evidence that the injury was the result of the individual's status as a public safety officer, that injury is covered by the PSOB program. The lump-sum PSOB death and disability benefit for FY2019 is $359,316. The benefit amount is adjusted annually to reflect changes in the cost of living using the annual percentage change in the Consumer Price Index for Urban Consumers (CPI-U) for the one-year period ending in the previous June. If a public safety officer receives a disability benefit and later dies from the same injury, the officer's survivors may not receive a PSOB death benefit. The payable benefit amount is Court struck down the federal Defense of Marriage Act in United States v. Windsor , the legally married spouse of a public safety office may be of the same sex as the officer. or mental disability. PSOB death benefits are paid to eligible survivors in the following order: 1. if the officer is survived by only a spouse, 100% of the death benefits are payable to the spouse; 2. if the officer is survived by a spouse and children, 50% of the death benefits are payable to the spouse and the remaining 50% is distributed equally among the officer's children; 3. if the officer is survived by only children, the death benefits are equally distributed among the officer's children; 4. if the $\cdots$
Now, write a one-page summary of the report.
Summary:

Prediction (GPT-3.5-turbo-1106)
The report provides information on the Public Safety Officers' Benefits (PSOB) program, which provides benefits to public safety officers, including law enforcement officers, firefighters, chaplains, FEMA employees, and emergency medical services members, who are permanently disabled or die as a result of a personal injury sustained in the line of duty. The report outlines the eligibility criteria for PSOB benefits, $\cdots$

Ground truth
The Public Safety Officers' Benefits (PSOB) program provides cash benefits to federal, state, and local law enforcement officers; firefighters; employees of emergency management agencies; and members of emergency medical services agencies who are killed or permanently and totally disabled as the result of personal injuries sustained in the line of duty. The Public Safety Officers' Educational Assistance (PSOEA) program, a component of the PSOB program, provides higher-education assistance to the children and spouses of public safety officers killed or permanently disabled in the line of duty $\cdots$

Figure 8: Case study on the GovReport dataset (Summarization) in the LongBench benchmark. The purple colorbox indicates the core information to generate answers.

Case study of FewShot task in LongBench

R2C Compressed Prompt with $T = 1,000$ (1,034 tokens, Original length: 15,836 tokens)

Summarize the dialogue into a few short sentences. The following are some examples.

⋮

Maisy: im hiding from a strange dinosaur :P wat happened to rach and cameron?
Steph: they split :/ yesterday
Maisy: oh no!
Dialogue: Tom: How are you today?
Tom: Great, I was afraid it could be difficult after last night
Summary: Jenny and Mia are already at the office.
Dialogue: Teresa: where should we meet?
Dialogue: Audrey: Honey, remember that we are going to buy a Christmas tree just after work.
Tom: I know, honey :)
Audrey: And a few gifts for kids
Tom: Do you have any ideas?
Audrey: Maybe some cars for Sam and some puzzles for Lucy?
Tom: It's a good idea.
Audrey: :*
Summary: Audrey and Tom will buy a Christmas tree and gifts for children after work.
Dean: ?
Summary: Tom had a job interview.
It's been planned a long time ago.
Bill: I'm terribly sorry but would it be very incovenient to you if we postponed our meeting? I admit it's a little disappointing. These are some private issues.
Liam: which book?

⋮

Dialogue: Jones: Hey.
Angelina: Hey.
Angelina: Long time. How are you doing?
Jones: I'm fine
Jones: You?
Angelina: I'm cool too.
Jones: You think we can meet today later in the afternoon in town?
Angelina: Definitely.
Jones: Okay. I will call you to confirm where we will meet.
Angelina: Cool
Summary:

Prediction (GPT-3.5-turbo-1106)

Jones and Angelina plan to meet in town later in the afternoon.

Ground truth

Jones and Angelina will meet in town in the afternoon.

Figure 9: Case study on the SAMSum dataset (FewShot) in the LongBench benchmark. The purple colorbox indicates the core information to generate answers.

Case study of Code Completion task in LongBench

R2C Compressed Prompt with $T = 1,000$ (980 tokens, Original length: 9,403 tokens)
Please complete the code given below.

```
    ⋮
              continue;
            } else {
              // Add it to the map
              accountIdAccountMap.put(grantInfo.getUserId(), account);
            }
          } catch (Exception e) { // In case the account is deleted, skip
              the grant
            LOG.warn("Account ID " + grantInfo.getUserId() + " does not not
                exist. Skipping this grant");
            deletedAccountIds.add(grantInfo.getUserId());
            continue;
          }
        }
  * This method transforms a Walrus bucket to an OSG bucket.
            owningAccount = getEucalyptusAccount();
            owningUser = getEucalyptusAdmin();
          } else if (noCanonicalIdAccountIds.contains(walrusBucket.
              getOwnerId())) { // If canonical ID is missing, use eucalyptus
              admin account
            LOG.warn("Account ID " + walrusBucket.getOwnerId() + " does not
                have a canonical ID. Changing the ownership of bucket "
                + walrusBucket.getBucketName() + " to eucalyptus admin
                    account");
             owningAccount = getEucalyptusAccount();
                // This is to avoid insert null IDs into cached sets/maps
This is executed in the {@code ModifyWalrusBuckets} stage</li>
  *
             adminUser = getEucalyptusAdmin();
            } else if (noCanonicalIdAccountIds.contains(walrusObject.
                getOwnerId())) { // If canonical ID is missing, use
                eucalyptus admin account
              LOG.warn("Account ID " + walrusObject.getOwnerId() + " does
                  not have a canonical ID. Changing the ownership of object "
                  + walrusObject.getObjectKey() + " in bucket " +
                      walrusObject.getBucketName() + " to eucalyptus admin
                      account");
              owningAccount = getEucalyptusAccount();
                  owningAccount = getEucalyptusAccount();
```

Next line of code:

Prediction (GPT-3.5-turbo-1106)

```
```
adminUser = getEucalyptusAdmin();
```
```

Ground truth

```
                  adminUser = getEucalyptusAdmin();
```

Figure 10: Case study on the LCC dataset (Code) in the LongBench benchmark. The purple colorbox indicates the core information to generate answers.