Learning to Compress Prompt in Natural Language Formats

Yu-Neng Chuang* Rice University ynchuang@rice.edu

Tianwei Xing Samsung Research America t.xing@samsung.com

Chia-Yuan Chang Texas A&M University cychang@tamu.edu

Zirui Liu Rice University zl105@rice.edu

Xun Chen Samsung Research America xun.chen@samsung.com

Xia Hu Rice University xia.hu@rice.edu

Abstract

Large language models (LLMs) are excel at processing multiple natural language processing tasks, but their abilities are constrained by inferior performance with long context, slow inference speed, and the high cost of computing the results. Deploying LLMs with precise and informative context helps users process large-scale datasets more effectively and cost-efficiently. Existing works rely on compressing long prompt contexts into soft prompts. However, soft prompt compression encounters limitations in transferability across different LLMs, especially API-based LLMs. To this end, this work aims to compress lengthy prompts in the form of natural language with LLM transferability. This poses two challenges: (i) Natural Language (NL) prompts are incompatible with back-propagation, and (ii) NL prompts lack flexibility in imposing length constraints. In this work, we propose a Natural Language Prompt Encapsulation (Nano-Capsulator) framework compressing original prompts into NL formatted *Capsule Prompt* while maintaining the prompt utility and transferability. Specifically, to tackle the first challenge, the Nano-Capsulator is optimized by a reward function that interacts with the proposed semantics preserving loss. To address the second question, Nano-Capsulator is optimized by a reward function featuring length constraints. Experimental results demonstrate that the *Capsule Prompt* can reduce **81.4%** of the original length, decrease inference latency up to $4.5\times$, and save **80.1%** of budget overheads while providing transferability across diverse LLMs and different datasets.

1 Introduction

Large Language Models (LLMs) have demonstrated substantial proficiency across a variety of natural language processing tasks. Despite their significant potential and broad adoption, LLMs are fundamentally limited by the long context length input, which impairs their capability to understand lengthy documents and affects their efficiency during inference (Touvron et al., 2023; Brown et al., 2020; Yang et al., 2023; Jin et al., 2024). As the demand for processing millions of tokens increases, it is progressively crucial to deploy LLMs that are adept at comprehending extended lengths while minimizing budgetary requirements.

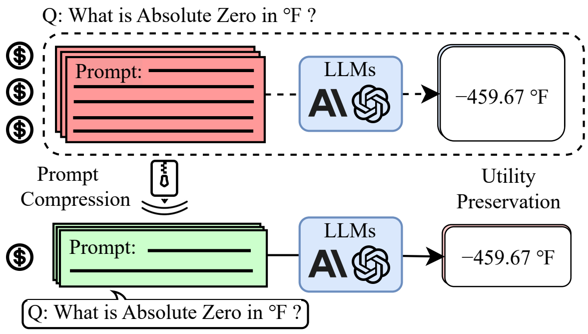

Figure 1: An example of successful prompt compression with NL formats. The compressed NL-formatted prompt (green) aims to obtain a shorter length and maintain transferability and utility of the long prompt (red).

To help LLMs better process long context knowledge, recent advancements have focused on compressing long prompt contexts into concise soft prompts. This approach effectively transforms the original extensive prompt into a manageable series of short-length soft prompt tokens. Generally, compression-oriented soft prompts are learned with the guarantee of semantics through self-information (Chevalier et al., 2023), instruction finetuning (Ge et al., 2023; Ren et al., 2023), and the performance alignment via knowledge distillation (Wingate et al., 2022; Mu et al., 2023). However, with the rapid evolution and the growth of API-based accessibility of LLMs, soft prompts pose significant limitations in terms of transferability across different LLMs, implying that well-trained soft prompts can only be effectively adapted to the specific LLMs for which they were designed.

*Work done as an intern at Samsung Research America.

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 7756–7767

This situation creates a critical need to achieve both transferability and utility effectively. A natural question is raised: ***Can we compress lengthy prompts in a natural language format, yet still preserve utility and ensure transferability among various LLMs?***

Compressing extended prompts into a shorter, natural language (NL) format continues to be a challenging and unresolved issue. As depicted in Figure 1, effective prompt compression entails preserving essential semantic information in a constrained length with successful performance preservation. However, unlike soft prompts, which can be directly optimized with a fixed length, compressing prompts into shorter NL prompts is challenging for several reasons: *(i) NL prompts are incompatible with back-propagation*, as the gradient cannot backward to a discrete raw text; *(ii) NL prompts lack flexibility on imposing strict length constraints*, where overly stringent limitations on generation length may lead to performance degradation. Thus, it is nontrivial to compress lengthy prompts into shorter NL ones.

To tackle the aforementioned problems, we propose a Natural Language Prompt Encapsulation (Nano-Capsulator) framework to effectively compress original prompts into a *Capsule Prompt* with the aid of a rewarding technique. Our proposed Nano-Capsulator aims to encapsulate long prompts into shorter ones under specific generation length constraints, maintaining performance through an explicit semantic preservation objective with reward scores. Specifically, we compress our prompt by employing a semantics-preserving summarization, and then monitor the optimization process using reward scores that reflect the remaining information relevant to the downstream task. Notably, shorter *Capsule Prompts*, characterized by their concise NL formatting, preserve transferability and utility across diverse LLMs. *Capsule Prompt* enables two advantages: the preservation of prompt transferability and utility across different LLMs, and the reduction of inference time and budget overheads. Additionally, Nano-Capsulator can be directly applied to unseen datasets without any further training, provided these new datasets encompass downstream tasks with similar domains.

To assess the effectiveness of Nano-Capsulator, we conduct compression experiments with two different prompt types: few-shot demonstration chain-of-thoughts (CoT) and passage prompts of reading comprehension (i.e., content passages). *Capsule Prompt* exhibits strong transferability across different LLMs and similar but unseen downstream datasets. This enables effective adaptation without retraining the prompt compressor.

Our main contributions are concluded as follows: *First*, we introduce and formalize Nano-Capsulator framework, which can effectively generate high-quality *Capsule Prompt* with prominent transferability across multiple LLMs and unseen datasets with similar downstream tasks. *Second*, we effectively reduce the original prompts to **81.4%** of their initial length and transform them into NL-formatted *Capsule Prompt*, which retains the prompt's transferability and utility across various LLMs. Our compression mechanism significantly decreases up to $4.5\times$ of the inference latency and saves **80.1%** of the budget overheads for input sequences. *Third*, experimental results demonstrate that *Capsule Prompt* can efficiently perform across diverse LLMs, which is applicable to both few-shot demonstration CoT and input contextual prompts.

2 Related Work

2.1 Soft Prompt Compression

In the realm of prompt compression for LLMs, most of the existing work aims to compress the prompt into soft prompts. Soft prompts are trainable vectors that are optimized in conjunction with a designated LLM, which embeds the original content information of the long hard prompts into low-dimensional vectors.

The first line of work (Wingate et al., 2022) leverages the knowledge distillation object to extract the information from hard prompts to soft prompts. The compressed soft prompts are expected to capture high-level concepts and preserve the fluency from the original hard prompts. The second line of work (Chevalier et al., 2023) employs the summarization capabilities of LLMs to condense lengthy and complex prompts into soft prompts. The process involves dividing the input prompts into multiple segments and sequentially compressing the information from the original prompt into smaller segments of soft prompts, where they assemble from these individual fragments of soft prompts to form the final soft prompts. Another work, Gist Token (Mu et al., 2023), condenses instruction prompts into customized prefix soft prompts by training a virtual soft prompt predictor.

Nevertheless, the transferability of soft prompt-based compression across diverse LLMs is constrained, necessitating the re-training of soft prompts with each change in the specified LLMs. This means that the soft prompts generated are specifically tailored to work only with that particular LLM, which falls short in maintaining transferability across different LLMs, especially applied on API-based LLMs, such as Claude2 (Anthropic, 2023) and PaLM (Chowdhery et al., 2023).

2.2 Context Distillation for Compression

Besides directly compressing hard prompts into soft prompt vectors, recent advancement (Li et al., 2023; Jiang et al., 2023) involves computing the self-information scores or perplexity of the given input context prompt to shorten the prompt length. This process includes filtering out words with lower scores from the input prompt, resulting in a more concise input during the inference stage. The primary distinction between our work and these recent studies is that they operate prompt compression without considering any information from downstream tasks, resulting in inferior performance while directly applying to downstream tasks or transferring between similar but unseen downstream datasets.

3 Long Prompt Encapsulation

We systematically introduce the Nano-Capsulator framework in this section. Figure 2 illustrates the overall pipeline of Nano-Capsulator. In particular, our pipeline initially compresses text into NL-formatted *Capsule Prompt* and concurrently optimizes their utility using the proposed rewarding method. The design of the NL-formatted compression aims to maintain prompt utility and preserve transferability among different LLMs.

3.1 Prompt Encapsulation

The primary aim of Nano-Capsulator is to preserve the inherent utility of the original pre-compressed text and ensure the compressed prompt closely reaches the designated length constraint. Specifically, the prompt encapsulation mechanism includes two components to effectively generate *Capsule Prompt*: (1) NL-formatted prompt compression and (2) prompt utility preservation. The learning of Nano-Capsulator involves integrating two components and optimizing them concurrently, thereby assuring the compressed prompts are sufficient to preserve their inherent utility.

Algorithm 1 Algorithm of Nano-Capsulator

Input: Original Long Prompt K, Compression Instructions $\mathbf{T}_{\text{Rep}}$ and $\mathbf{T}_{\text{Summ}}$, and pre-trained frozen LLMs $\mathcal{G}^*(\cdot)$, Sampled set of downstream task questions Q.
Output: NL-formatted Capsule Prompts C.

```
Initialize F(· | θ_C) and G*(·) with pre-trained weights
while not convergence do
    Generate C from F(K | T_Rep, T_Summ, θ_C)
    Randomly sample a set of questions Q
    Receive reward scores from R_cap(G*(·), Q, C, K)
    F(· | θ_C) ← minimizing with L_Nano(·)
end while
```

3.1.1 NL-formatted Prompt Compression

We adopt an unsupervised training approach featuring semantic preservation loss, motivating the model to compress contexts while retaining similar semantic content. In this work, we shorten the long prompts by summarizing their context and applying our proposed semantics loss $\mathcal{L}_{\text{Comp}}$ to ensure maximal preservation of semantic meaning. Here, semantics refers to the logical thinking process from the few-shot demonstration CoT and the beneficial content from context passages.

Given the original prompt $K = \{k_1, \cdots k_n\}$ consisted of n tokens to the *Capsule Prompt* $C = \{c_1, \cdots c_m\}$ with m tokens, where $n \gg m$. Our semantics loss aims to ensure the maximal semantics preservation by measuring the similarity between the hidden state embedding of C and K of Nano-Capsulator $\mathcal{F}(\cdot \mid \theta_C)$. To obtain the hidden state embedding of K, we instruct $\mathcal{F}(\cdot \mid \theta_C)$ to replicate the input prompt K, which aids in better preserving and embedding the semantic meanings of K. Specifically, d-dimensional hidden state embedding of K and C can be generated by $\boldsymbol{e}_K \sim P_{\mathcal{F}}(K \mid \theta_C, \mathbf{T}_{\text{Rep}})$ and $\boldsymbol{e}_C \sim P_{\mathcal{F}}(C \mid \theta_C; \mathbf{T}_{\text{Summ}})$, where $\mathbf{T}_{\text{Rep}}$ and $\mathbf{T}_{\text{Summ}}$ denote a replicating instruction and a summarizing instruction, respectively. With the aid of $\mathbf{T}_{\text{Rep}}$, we compel $\mathcal{F}(\cdot \mid \theta_C)$ to replicate K under the model parameter θ_C, ensuring that $\boldsymbol{e}_K \in \mathbb{R}^d$ accurately represents the embedding of K. Following the criterion, $\mathcal{F}(\cdot \mid \theta_C)$ essentially minimizes the semantics loss as follows:

$$\mathcal{L}_{\text{Comp}} = \mathbb{E}_C \left[D_{\text{dist}}(\boldsymbol{e}_K \,||\, \boldsymbol{e}_C) \right] \tag{1}$$

where $D_{\text{dist}}(\cdot \,||\, \cdot)$ can be any suitable distance measurement in metric space. In this work, we leverage mean square error as the distance function to measure the similarity between $\boldsymbol{e}_K$ and $\boldsymbol{e}_C$.

3.1.2 Prompt Utility Preservation

To impose a constraint on the generated length while preserving utility, we establish a reward func-

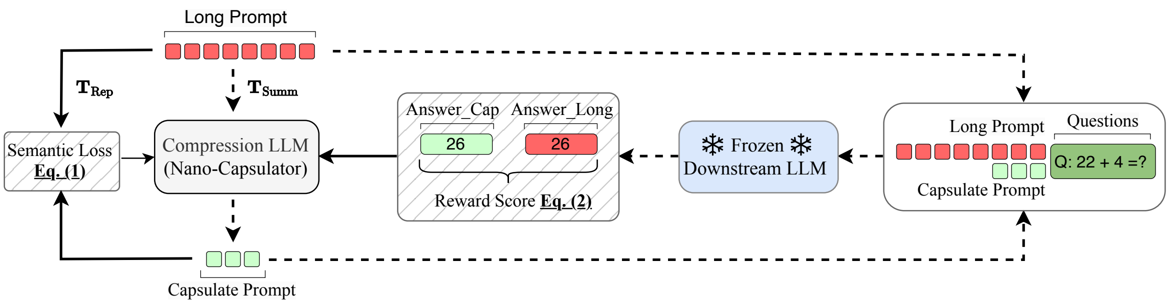

Figure 2: The illustration of Nano-Capsulator training framework. Nano-Capsulator compress the long prompt with the action of semantic (Equation 1) and utility preservation (Equation 2). Questions are sampled from the training set to develop the reward scores for utility preservation.

tion $\mathcal{R}_{\text{cap}}(\cdot)$ featuring a strict cut-off mechanism $\Phi(\cdot)$ for restricting the generated length of *Capsule Prompt*. The high-level idea of the reward function is to calculate the score changes of the downstream task question based on leveraging the original prompt K and the *Capsule Prompt* C. Notably, the reward function employs a truncation strategy to limit the C to a predetermined length before proceeding to compute the scores using the reward function. In this manner, the *Capsule Prompt* that surpasses the specified length threshold could be assigned a lower reward score as a result of the cut-off mechanism.

Formally, given K and C along with arbitrary pre-trained frozen LLMs $\mathcal{G}^*(\cdot)$ and a sampled set of downstream task questions Q, the reward function $\mathcal{R}_{\text{cap}}(\cdot)$ can be defined as follows:

$$\mathcal{R}_{\text{cap}} = \mathbb{E}_Q\big[\, \mathbf{I}\{\, \mathcal{G}\big(\Phi(C_i) \oplus Q_i\big) \mid\mid \mathcal{G}\big(K_i \oplus Q_i\big)\ \}\,\big] \tag{2}$$

where $\mathbf{I}(\cdot \mid\mid \cdot)$ denotes an arbitrary reward metric for yielding the reward score., and $\oplus$ represents concatenation of prompts and questions. In this study, we calculate the reward scores using the mean square error between the hidden state embedding from $\mathcal{G}^*(\cdot)$. It's noteworthy that $\mathbf{I}(\cdot \mid\mid \cdot)$ can be replaced by other metrics, such as accuracy and GPT4Eval Scores (Liu et al., 2023), facilitating its potential application to API-based LLMs.

3.1.3 Compression with Reward

Upon receiving the reward scores from $\mathcal{R}_{\text{cap}}$ as per Equation 2, we synchronize these scores with the semantic loss $\mathcal{L}_{\text{Comp}}$ to maintain utility. Formally, the ultimate objective function of Nano-Capsulator can be indicated as:

$$\mathcal{L}_{\text{Nano}} = \mathcal{L}_{\text{Comp}}(\cdot|\theta_C) * \mathcal{R}_{\text{cap}}(\cdot|\theta^*) \tag{3}$$

where θ^* denotes the frozen model parameters of $\mathcal{G}^*(\cdot)$ and θ_C is the trainable parameters of Nano-Capsulator. The fundamental principle of $\mathcal{L}_{\text{Nano}}(\cdot)$ is to impose penalties when shorter versions of *Capsule Prompt* exhibit inferior performance. This implies that if a *Capsule Prompt* receives a low reward score from Equation 2, its semantic loss will be composed by a high penalty value, resulting in a substantial semantic loss value as punishment during the training phase of Nano-Capsulator.

3.2 Algorithm of Nano-Capsulator

The framework of Nano-Capsulator is detailed in Algorithm 1. Nano-Capsulator adheres to Equation 1 for the preservation of semantic meaning and integrates the rewarding function from Equation 2 to maintain the utility of compressed NL-formatted prompts. The two elements are then aligned, as depicted in Equation 3, and optimized simultaneously with the goal of obtaining compressed NL-formatted prompts of high utility. In the inference phase, Nano-Capsulator is solely required to produce the compressed version of *Capsule Prompt* from the provided long input prompt.

4 Experiments

In this section, we conduct experiments to evaluate the performance of Nano-Capsulator, aiming to answer the following three research questions:

- **RQ1:** How does Nano-Capsulator perform in terms of the efficacy and transferability among different LLMs and datasets?
- **RQ2:** How do the two components of Nano-Capsulator contribute to the compression performance for utility preservation?
- **RQ3:** What are the inference latency and impact factors of *Capsule Prompt*?

4.1 Dataset

We conduct compression experiments with two different prompt types: few-shot CoT and passage prompts of reading comprehension. The details of the datasets are provided as follows:

Few-shot CoT Dataset. We choose two reasoning datasets to evaluate the proposed framework.

- **CommonsenseQA (Talmor et al., 2019):** The CommonsenseQA (CSQA) dataset is a publicly accessible collection of multiple choice questions with 1221 samples for the commonsense reasoning task. CSQA presents questions characterized by intricate semantics, typically demanding reasoning grounded in pre-existing knowledge.
- **GSM8K (Cobbe et al., 2021):** The GSM8K is a dataset containing 1319 samples of graduate school math questions. Each question is collected from the Math World Problem Repository (Roy and Roth, 2015) with a numerical answer.

Reading Comprehension Dataset.

- **MultiRC (Khashabi et al., 2018; DeYoung et al.):** MultiRC (Multi-Sentence Reading Comprehension) comprises a collection of brief paragraphs paired with multi-sentence questions, where the answers can be derived from the paragraph's content. The dataset obtains 24029 samples for training, 3214 samples for validating, and 4848 samples for testing.
- **TriviaQA LongBench (Joshi et al., 2017):** TriviaQA LongBench (TriviaQA-Long) is a reading comprehension dataset featuring 300 question-answer-evidence triples collected from the Longbench dataset. It includes question-answer pairs created by trivia enthusiasts, along with independently sourced evidence documents, offering robust supervision for responding to the questions.

4.2 Experiment Settings

In this part, we introduce the experimental settings and metrics for evaluating Nano-Capsulator. Two distinct types of transferability evaluations are taken into account. *To verify the model transferability*, the compression models are trained on one downstream LLM, and tested on different downstream LLMs. The evaluation is performed on the same dataset, with a division of 70% allocated for training and validation and 30% designated as the testing set. *To assess data transferability*, we train the compression models using one seen dataset and then test them on unseen datasets that the models have not previously encountered with the same downstream tasks. The considered compression settings and implementation details are shown as follows. Two types of prompt compression tasks are focused on.

Few-shot CoT Compression Task. For the few-shot CoT compression task, we randomly compile seven examples from the CSQA dataset and eight from the GSM8K dataset following (Wei et al., 2022), all selected from their respective training sets, to construct the few-shot CoT. During the training phase, a total of 1,000 CoT samples are then gathered to serve as the training data for Nano-Capsulator. During the inference stage, we aim to compress the manual few-shot CoT proposed in (Wei et al., 2022), where the demonstrations in manual CoT are eliminated from any training set. The primary evaluation metric used in both CSQA and GSM8K datasets is accuracy, implying that the model scores only when it provides answers that exactly match the expected responses.

Reading Comprehension Compression Task. For the reading compression task, we aim to compress the reading paragraphs from the question-answer triplets. Due to the limitation of GPU memory, we eliminate the paragraphs that exceed 2k tokens in TriviaQA from the LongBench dataset, resulting in an average length of 900 tokens, while MultiRC remains to utilize the all paragraphs in the dataset. Throughout the training phase, we select 2,000 question-answer-paragraph triplets in the MultiRC dataset to serve as training data and leverage all training data in the TriviaQA-Long dataset. Our framework is evaluated on the entire set of testing data, using accuracy as the metric of assessment.

Implementation Details. In primary experiments, we utilize Vicuna-7B (Chiang et al., 2023) as the initial compression model $\mathcal{F}(\cdot \mid \theta_C)$ in Nano-Capsulator. The pre-trained LLMs $\mathcal{G}^*(\cdot)$ is given as Vicuna-7B with frozen weights. We train Nano-Capsulator using Vicuna-7B and then assess the generated *Capsule Prompt* with various LLMs other than Vicuna-7B, in order to evaluate its transferability. To reduce memory consumption during training, we utilize LoRA[1] and train the Nano-Capsulator using two NVIDIA A40 GPUs of 48GB memory. We employ the Adam optimizer for the fine-tuning process, with a learning rate set at 5e-6

[1]PEFT: https://github.com/huggingface/peft

	CSQA			GSM8K			MultiRC		TriviaQA-Long	
	Manual	Zero-shot	Ours	Manual	Zero-shot	Ours	Original	Ours	Original	Ours
Vicuna-13B	60.4	44.6	58.8	34.4	25.3	31.9	57.3	57.1	86.0	88.8
PaLM	73.7	67.5	75.5	62.8	56.8	59.5	72.7	72.2	78.9	78.8
Claude2	76.6	69.4	74.6	85.6	52.7	84.9	59.4	58.2	95.0	92.3
Length (# of Token)	831	–	154	751	–	231	378.39	95.66	915.7	422.6
Compress Rate (%)	–	–	81.4%	–	–	69.3%	–	74.71%	–	53.84%

Table 1: Evaluation of Nano-Capsulator among different LLMs. The results show that Nano-Capsulator compress up to 81.4% of the original long prompt and save up to 80.1% of the expense on requesting for LLM API calls.

	Claude2 (Anthropic, 2023)		
Cost($)	Original	*Capsule Prompt*	Saved
CSQA	15.03	3.30	-77.9%
GSM8K	5.22	1.88	-63.9%
MultiRC	45.91	13.01	-71.6%
TrivaQA-Long	2.14	0.42	-80.1%

Table 2: API cost comparison of *Capsule Prompt* and original prompt, where *Capsule Prompt* save up to 80.1% of the original cost.

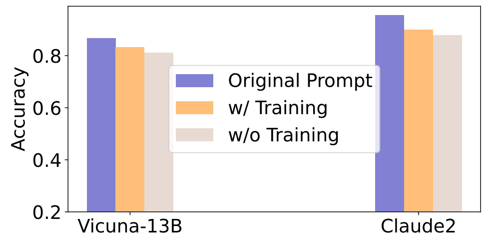

Figure 3: Evaluation of transferability on Nano-Capsulator across unseen datasets.

under the gradient clipping of 0.8, depending on the datasets. The instructions that leverage for prompt encapsulation $\mathbf{T}_{\text{Rep}}$ and $\mathbf{T}_{\text{Summ}}$ are listed in Table 6 from Appendix F.

4.3 Main Results (RQ1)

Model Transferability. To assess the effectiveness and transferability, we compress the original input prompts into the *Capsule Prompt* by Nano-Capsulator. We then evaluate the transferability and utility of these compressed prompts across three different LLMs not included in the pre-training of Nano-Capsulator: Vicuna-13B (Chiang et al., 2023), PaLM (Chowdhery et al., 2023), and Claude2 (Anthropic, 2023). The main findings are presented in Table 1. In the table, "Manual" refers to the manually created few-shot CoT proposed by (Wei et al., 2022), "Zero-shot" denotes the zero-shot CoT followed (Kojima et al., 2022), and "Original" indicates the original paragraphs used in the reading comprehension tasks.

In the primary experiment, we establish a compression constraint limiting to a maximum of 150 tokens for the CSQA and MultiRC datasets; and a maximum of 350 and 500 tokens for the GSM8K and TriviaQA-Long dataset, where 150 tokens are not sufficient for preserving the logic of GSM8K and TriviaQA-Long dataset. We observe that the Nano-Capsulator obtains up to 81.4% of the compression rate and saves up to 80.1% of the Claude2 API cost compared to the original input prompts, as displayed in Table 2. The cost of PaLM API can be found in Appendix C. For utility preservation, Nano-Capsulator retains the original performance on the CSQA, GSM8k, and TriviaQA-Long datasets mostly among three LLMs. Remarkably, Nano-Capsulator maintains almost identical performance to that achieved with non-compressed prompts in MultiRC datasets. The significant compression rate can advantageously impact the LLMs by allowing for a higher tolerance in batch inference, accompanied by reduced latency and cost.

Dataset Transferability. We previously assessed the effectiveness of Nano-Capsulator within the same datasets, where the testing set was derived from the same training domain. In this section, we investigate the transferability of Nano-Capsulator across unseen datasets (i.e., not in training data) with the same downstream tasks. We train Nano-Capsulator on the MultiRC dataset (seen dataset) and test on BoolQ (Clark et al., 2019) (unseen dataset) without any further training, where BoolQ is also a reading comprehension dataset, under Vicuna-13B and Claude2. The results are demonstrated in Figure 3. We see a competitive performance with only a slight accuracy drop compared to the training version of *Capsule Prompt*. While *Capsule Prompt* yields better performance with training, the results indicate that Nano-Capsulator possesses a great property of data transferability.

4.4 Contributions of Utility Preservation (RQ2)

In this section, we explore the effectiveness of components from Nano-Capsulator. Specifically,

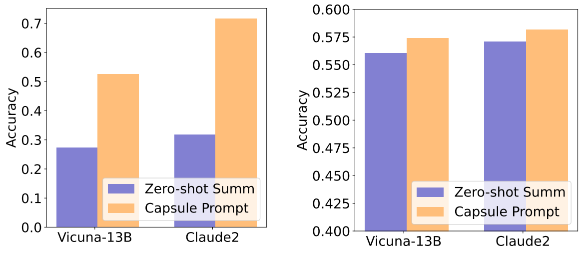

Figure 4: Comparison results of *Capsule Prompt* and Zero-shot Summarization on GSM8K dataset (left) and MultiRC dataset (right).

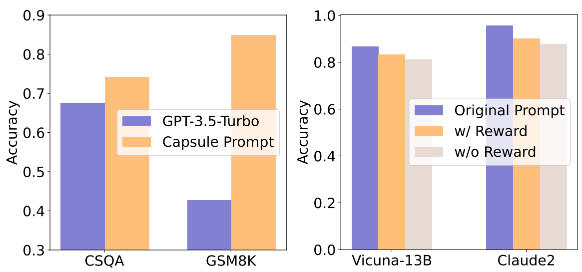

Figure 5: Ablation studies of comparison with *Capsule Prompt* and GPT-35-Turbo Summarization on CSQA dataset and GSM8K dataset (left); and of the contribution of Reward Function from Equation 2 (right).

we conduct ablation studies from two perspectives. *First*, we evaluate the efficacy of semantic preservation. We compare Nano-Capsulator with in-context zero-shot summarization generated by Vicuna-7B, as Vicuna-7B is the initial model weight of Nano-Capsulator for prompt compression. The results are demonstrated in Figure 4 with the comprehensive comparison of three LLMs, including Vicuna-13B and Claude2. We observe that *Capsule Prompt* yielded by Nano-Capsulator outperforms in all scenarios, which means that our Nano-Capsulator can significantly preserve more semantic information and preserve prompt utility. Additionally, we assess the performance by directly employing GPT-3.5-Turbo to summarize the provided prompts. Figure 5 (left) illustrates that Nano-Capsulator maintains a higher level of prompt utility compared to GPT-3.5-Turbo, resulting in enhanced performance on both the CSQA and GSM8K tasks.

Secondly, we carry out ablation studies to establish the effectiveness of the reward function in Nano-Capsulator. These studies are conducted on Vicuna-13B and Claude2 using the TriviaQA-Long dataset. As shown in Figure 5 (right), the results indicate a degradation in performance for downstream tasks when the reward function is not utilized. Note that "w/ reward" means Nano-Capsulator trained with the reward function, while

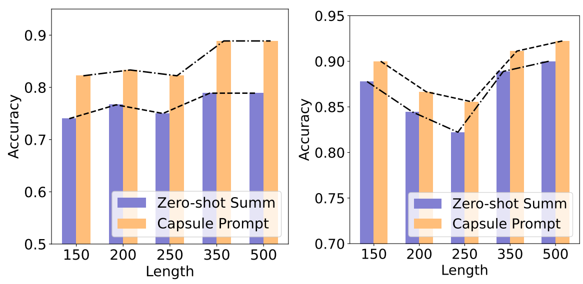

Figure 6: Impact of prompt length on Vicuna-13B (left) and Claude2 (right) on TriviaQA dataset.

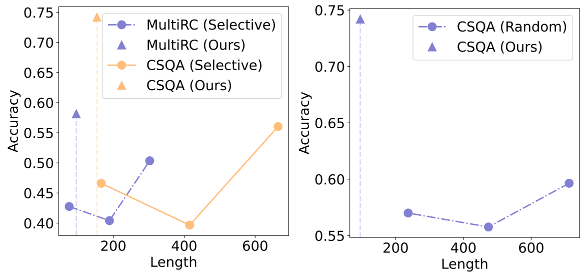

Figure 7: The comparison results of *Capsule Prompt* and text dropping methods, including Selective Context (left) and random demonstration elimination (right).

"w/o reward" denotes a Nano-Capsulator trained without the reward function. We further present case studies of *Capsule Prompt* on GSM8K to showcase the logic preservation, as illustrated in Figure 10 from Appendix E. These studies clearly demonstrate that *Capsule Prompt* retains more semantic meanings by preserving complete logical structures, suggesting that the utility of prompts is better maintained.

4.5 Exploration of Impact Factors (RQ3)

In this section, we explore our proposed compression mechanism in greater detail, examining various factors that influence its performance.

Impact of Capsule Prompt Length. In the main experiment, the length constraint is fixed to 150 or 300 tokens for the prompt compression. We further explore how the compression rate affects the preservation of utility. The results are displayed in Figure 6, obtained from experiments conducted using the TriviaQA dataset with Vicuna-13B and Claude2. We observe that the length of *Capsule Prompt* can impact its utility on different LLMs. While longer prompts might capture more useful information for downstream tasks, they can also introduce certain noise or misinformation to the LLMs. This can result in suboptimal performance when specific LLMs interact with the compressed

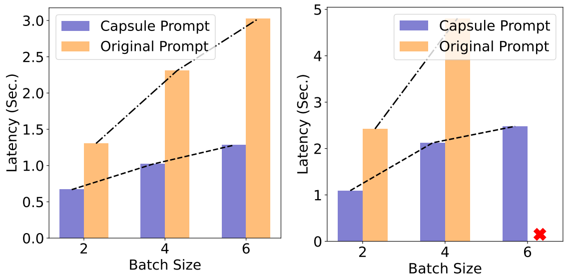

Figure 8: Inference Latency of Vicuna-13B on CSQA dataset (left) and TrivaQA-Long dataset (right), where ✖ indicates out of memory.

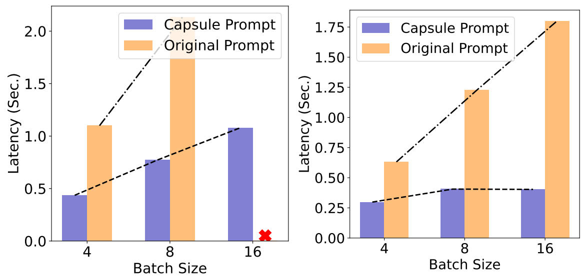

Figure 9: Inference Latency of OPT-2.7B on CSQA dataset (left) and TrivaQA-Long dataset (right), where ✖ indicates out of memory.

prompts. The situation can be observed from the results of *Capsule Prompt* and Zero-shot Summ Prompt, where the length of 150 outperforms other length settings under Claude2 and the length of 200 outperforms other length settings Vicuna-13B. We notice that the desired length settings of Nano-Capsulator can be observed from the performance of the Zero-shot Summ Prompt, as they share similar performance trends.

Impact of Discrete Text Elimination. In addition to compressing prompts using Nano-Capsulator, we acknowledge that prompt length can also be reduced by employing methods like random dropping or rule-based selection. To this end, we have conducted studies comparing the performance of straightforward text dropping with our proposed framework. We consider two baseline methods under Claude2: a naive random demonstration elimination on the CSQA dataset and Selective Context as described in (Li et al., 2023), in which Selective Context eliminated the word according to the self-information values, on both the CSQA and MultiRC datasets. The outcomes of these comparisons are showcased in Figure 7. We observe that *Capsule Prompt* outperforms the other two baselines. Particularly, *Capsule Prompt* achieves better performance when the length is similar to the baselines. This again demonstrates the effectiveness of *Capsule Prompt* in preserving the utility.

4.6 Latency of Nano-Capsulator (RQ3)

The configuration of the computational infrastructure is given in Appendix A. We conducted the efficiency experiments on two publicly available LLMs: OPT-2.7B (Zhang et al., 2022) and Vicuna-13B (Chiang et al., 2023), under different batch size settings with the generated length of 200 tokens. Due to the limited GPU memory, we set Vicuna-13B as bfloat16 to accommodate a single GPU, while OPT-2.7B remains its inherent version. As demonstrated in Figure 8 and Figure 9, we observe that Nano-Capsulator framework achieves much lower computational latency compared to the case while inputting the original prompt on both LLMs. Specifically, *Capsule Prompt* obtains mostly of its original performance while reducing $2.1\times \sim 4.5\times$ of execution latency. As depicted in Figure 9, *Capsule Prompt* is capable of being accommodated within OPT-2.7B under a larger batch size, whereas the use of the original longer prompt leads to an out-of-memory issue (i.e., when batch size = 16). Additionally, we notice that *Capsule Prompt* achieves considerable benefits in speeding up the inference process as the batch size increases. Under OPT-2.7B, *Capsule Prompt* accelerates the process by up to $4.5\times$, and under Vicuna-13B, it achieves a speed increase of $4.1\times$ compared to the original input prompt. This indicates that *Capsule Prompt* allows for a larger batch size while reducing the time required for the inference process.

5 Conclusion

Our work introduces Natural Language Prompt Encapsulation (Nano-Capsulator), a framework for effectively compressing long prompts for LMs while preserving essential information. Nano-Capsulator alleviates the context length limitations of LLMs, enhancing processing efficiency and cost-effectiveness. Our results show that Nano-Capsulator reduces prompt lengths by 81.4%, decreases inference latency by up to 4.5×, and cuts budget overheads by 80.1% with almost identical accuracy and relevance. This demonstrates its significant potential for improving LLM efficiency across various applications that utilize long input documents. Future research will focus on refining Nano-Capsulator for broader domain applications and exploring its usage in data-intensive fields.

References

Anthropic. 2023. Claude.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. Eraser: A benchmark to evaluate rationalized nlp models.

Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. Llm maybe longlm: Self-extend llm context window without tuning. *arXiv preprint arXiv:2401.01325*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, page arXiv:1705.03551.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment, may 2023. *arXiv preprint arXiv:2303.16634*, 6.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*.

Siyu Ren, Qi Jia, and Kenny Q Zhu. 2023. Context compression for auto-regressive transformers with sentinel tokens. *arXiv preprint arXiv:2310.08152*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. 2022. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. *arXiv preprint arXiv:2210.03162*.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *arXiv preprint arXiv:2304.13712*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Appendix

A Computation Infrastructure

For a fair comparison of testing algorithmic throughput, the experiments are conducted based on the following physical computing infrastructure in Table 3.

Device Attribute	Spec
Computing infrastructure	GPU
GPU model	Nvidia-A40
GPU number	1
GPU Memory	46068 MB

Table 3: Computing infrastructure for the experiments.

B Additional Experiments of Comparison to Soft Prompt Baselines

To evaluate our proposed framework against existing soft prompt methods, we conduct experiments with AutoCompressors (Chevalier et al., 2023) on the GSM8K dataset, as shown in Table 4. Our Capsule Prompt is utilized for predictions using the Llama-2-7B model, which is identical to the pre-trained model used by AutoCompressor. As we can see, AutoCompressor does not preserve essential information in the compressed soft prompts, leading to a considerable performance drop in the GSM8K task.

GSM8K	AutoCompressors	Ours
Accuracy	3.79	19.7

Table 4: Computing infrastructure for the experiments.

	PaLM (Chowdhery et al., 2023)		
Cost($)	Original	*Capsule Prompt*	Saved
CSQA	0.156	0.034	-77.9%
GSM8K	0.054	0.019	-63.9%
MultiRC	0.478	0.135	-71.6%
TrivaQA-Long	0.022	0.004	-80.1%

Table 5: API cost comparison of *Capsule Prompt* and original prompt on PaLM, where *Capsule Prompt* save up to 80.1% of the original cost.

C API Cost of PaLM

We here provide the API cost of PaLM during the evaluation of Nano-Capsulator. We can observe that *Capsule Prompt* generated by Nano-Capsulator save up to 80.1% of its original cost on PaLM. The results further underscore the excellent cost-efficiency attributes of *Capsule Prompt*.

D Training Costs of Nano-Capsulator

In this section, we discuss the training time the memory cost of Nano-Capsulator in this section. All datasets are trained using the initial weights of Vicuna-7B. Training time and memory requirements are different with the volume and types of training data. In our work, the training time for the few-shot CoT compression task is approximately 8 hours, and for the reading comprehension compression task, it is about 4 hours. Once the Nano-Capsulator completes the training process, we can directly derive the compressed hard prompts through a single forward pass of Nano-Capsulator.

E The Case Studies of *Capsule Prompt*

We demonstrate the case study of *Capsule Prompt* on the GSM8K and MultiRC dataset. The results, depicted in Figure 10 and Figure 11, show that Nano-Capsulator can better obtain the semantic meanings of math logic from the original CoT prompt. We observe that in the absence of Nano-Capsulator, the Zero-shot Summarization approach struggles to compress long prompts effectively.

F Instruction Usage in Inference LLMs

We provide a list of the instruction utilized in training our Nano-Capsulator framework in Table 6, including $\mathbf{T}_{\text{Rep}}$ replicating instruction and summarizing instruction $\mathbf{T}_{\text{Summ}}$.

Original Prompt

- **Q:** There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
  A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6. The answer is 6.
- **Q:** If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
  A:There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5. The answer is 5.
- **Q:** Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
  A: Originally, Leah had 32 chocolates. Her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 - 35 = 39. The answer is 39.
- **Q:** Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
  A: Jason started with 20 lollipops. Then he had 12 after giving some to Denny. So he gave Denny 20 - 12 = 8. The answer is 8.
- **Q:** Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
  A: Shawn started with 5 toys. If he got 2 toys each from his mom and dad, then that is 4 more toys. 5 + 4 = 9. The answer is 9.
- **Q:** There were nine computers in the server room. Five more computers were installed each day, from monday to thursday. How many computers are now in the server room?
  A: There were originally 9 computers. For each of 4 days, 5 more computers were added. So 5 * 4 = 20 computers were added. 9 + 20 is 29. The answer is 29."
- **Q:** Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many golf balls did he have at the end of wednesday?
  A: Michael started with 58 golf balls. After losing 23 on tuesday, he had 58 - 23 = 35. After losing 2 more, he had 35 - 2 = 33 golf balls. The answer is 33.
- **Q:** Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
  A: Olivia had 23 dollars. 5 bagels for 3 dollars each will be 5 x 3 = 15 dollars. So she has 23 - 15 dollars left. 23 - 15 is 8. The answer is 8.

Capsule Prompt

- There were 15 trees in the grove, and grove workers planted 6 more trees, so there are now 21 trees in the grove.
- In the parking lot, there were 3 cars, and 2 more cars arrived, so there are now 5 cars in the parking lot.
- Leah and her sister had 32 and 42 chocolates, respectively, and they ate 35 chocolates, so they have 39 chocolates left in total.
- Jason had 20 lollipops and gave 8 lollipops to Denny, so he now has 12 lollipops.
- Shawn had 5 toys and got 2 toys each from his mom and dad, so he now has 9 toys.
- There were 9 computers in the server room, and 20 more computers were installed, so there are now 29 computers in the server room.
- Michael had 58 golf balls and lost 23 on tuesday and 2 more on wednesday, so he has 33 golf balls left.
- Olivia had $23 and bought 5 bagels for $3 each, so she has $8 left.

Zero-shot Summarization Prompt

- Grove workers planted 6 trees in the grove.
- There are now 5 cars in the parking lot.
- Leah and her sister have 39 pieces of chocolates left in total.
- Jason gave Denny 8 lollipops.Shawn has 9 toys now.
- There are 29 computers in the server room.
- Michael had 33 golf balls at the end of wednesday.
- Olivia has 8 dollars left.

Figure 10: A case study on GSM8K dataset. The results are the *Capsule Prompt* and in-context summarization prompt generated Vicuna-7B, following the settings in RQ2.

Compression Tasks	$\mathbf{T}_{\text{Rep}}$ (Replicating Instruction)	$\mathbf{T}_{\text{Summ}}$ (Summarizing Instruction)
Few-shot CoT	Repeat the following main input.	Please summarize each question-answer pair in one sentence within less than {word count} words. Make sure not to repeat the input question-answer pair.
Reading Comprehension	Repeat the following main input.	Please summarize the passage within less than {word count} words. Make sure not to repeat the passage.

Table 6: Instructions used in training Nano-Capsulator.

Original Prompt

- The day was just breaking , as I left the tower ; though it was still too dark in the house to be able to see without a light , and I took one of the study candles with me on my ' round . By the time I had finished the ground floor , the daylight was creeping in , wanly , through the barred windows . My search had shown me nothing fresh . Everything appeared to be in order , and I was on the point of extinguishing my candle , when the thought suggested itself to me to have another glance ' round the cellars . I had not , if I remember rightly , been into them since my hasty search on the evening of the attack . For , perhaps , the half of a minute, I hesitated . I would have been very willing to forego the task -- as , indeed , I am inclined to think any man well might -- for of all the great , awe - inspiring rooms in this house , the cellars are the hugest and weirdest . Great , gloomy caverns of places , unlit by any ray of daylight . Yet , I would not shirk the work . I felt that to do so would smack of sheer cowardice . Besides , as I reassured myself , the cellars were really the most unlikely places in which to come across anything dangerous ; considering that they can be entered , only through a heavy oaken door , the key of which , I carry always on my person.

Capsule Prompt

- The narrator left the tower and took a candle with them to search the ground floor. The daylight was creeping in through the barred windows by the time the narrator finished searching the ground floor. The narrator hesitated to search the cellars but decided to do so because they did not want to be seen as cowardly and because the cellars were the most unlikely place to find anything dangerous.

Zero-shot Summarization Prompt

- The narrator left the tower and took a candle with them, searching the ground floor of the house. The daylight was creeping in through the barred windows, but the search showed nothing new. The narrator hesitated before deciding to search the cellars, which were large and eerie, but ultimately decided to face their fear and complete the task.

Figure 11: A case study on MultiRC dataset. The results are the *Capsule Prompt* and in-context summarization prompt generated Vicuna-7B, following the settings in RQ2.