EFPC: Towards Efficient and Flexible Prompt Compression

Anonymous ACL submission

Abstract

The emergence of large language models (LLMs) like GPT-4 has revolutionized natural language processing (NLP), enabling diverse, complex tasks. However, extensive token counts lead to high computational and financial burdens. To address this, we propose Efficient and Flexible Prompt Compression (EFPC), a novel method unifying task-aware and task-agnostic compression for a favorable accuracy-efficiency trade-off. EFPC uses GPT-4 to generate compressed prompts and integrates them with original prompts for training. During training and inference, we selectively prepend user instructions and compress prompts based on predicted probabilities. EFPC is highly data-efficient, achieving significant performance with minimal data. Compared to the state-of-the-art method LLMLingua-2, EFPC achieves a 4.8% relative improvement in F1-score with 1% additional data at a 4× compression rate, and an 11.4% gain with 10% additional data on the LongBench single-doc QA benchmark. EFPC's unified framework supports broad applicability and enhances performance across various models, tasks, and domains, offering a practical advancement in NLP.

1 Introduction

The rise of large language models (LLMs) such as GPT-4 has significantly advanced the field of natural language processing, making it possible to tackle a wide range of complex tasks. Various prompting techniques, such as Chain-of-Thought (COT) (Wei et al., 2022), In-context Learning (ICL) (Dong et al., 2023), and Retrieval Augmented Generation (RAG) (Lewis et al., 2020), have been instrumental in maximizing the potential of these models by generating rich and informative prompts. However, these methods often require prompts that are tens of thousands of tokens long, resulting in increased computational and financial overhead as well as diminished information perception abilities of the LLMs (e.g., degraded performance when processing noisy and lengthy contexts (Jiang et al., 2023c)).

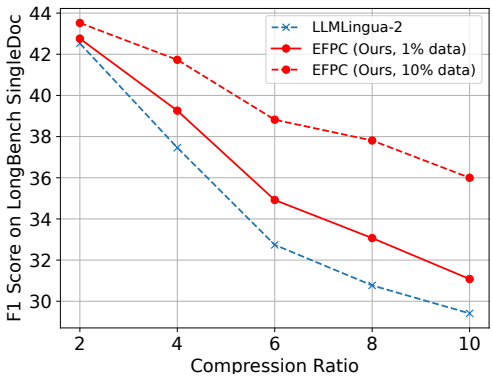

Figure 1: Performance under varying compression rates and training data amounts. Our EFPC method achieves significant improvements with minimal training data and larger gains with higher compression rates.

To address these challenges, prompt compression has emerged as a promising solution. The goal is to shorten prompts without losing essential information, thereby improving efficiency and reducing costs. Existing methods generally fall into two categories: task-aware and task-agnostic. Task-aware methods (Jiang et al., 2023c; Xu et al., 2024; Jung and Kim, 2023; Huang et al., 2023), tailor compressed prompts to specific tasks or queries, achieving enhanced performance on downstream tasks. However, these methods often face challenges in terms of efficiency and generalizability, requiring multiple compressions of the same document depending on the associated queries and typically relying on time-consuming Transformer decoder architectures (Vaswani et al., 2017). Task-agnostic methods (Jiang et al., 2023b; Li et al., 2023), propose to compress prompts by removing tokens or lexical units based on information entropy, irrespective of the downstream task. Re-

cently, LLMLingua-2 (Pan et al., 2024) approachs prompt compression as a token classification task and take the predicted probability of each token being labeled as preserve as the compression metric. While offering better generalizability, these methods suffer from insuperior performanace on tasks such as question answering.

Building on previous foundational works, we propose an Efficient and Flexible Prompt Compression method, named EFPC, which can adapt between task-aware and task-agnostic scenarios based on the specific use case to maintain high efficiency. Specifically, our method leverages GPT-4 to compress prompts while ensuring their ability to perform tasks such as answering user questions. These compressed prompts are then paired with the original ones to create a binary classification training set, and a binary classifier with a Transformer encoder architecture (Devlin et al., 2019) is trained on this set. During training and inference, user instructions are concatenated with the original prompts to predict the retention probability for each word, which is then used to compress the prompts by discarding low-probability words. For task-agnostic compression, the user instruction is set to an empty string.

Moreover, this work addresses two primary challenges in the field: balancing efficiency and accuracy, and mitigating high data costs associated with prompt compression.

- Efficiency and Accuracy Balance. Existing work such as LongLLMLingua (Jiang et al., 2023c) employs decoder-based LLM LLaMA-2-7B (Touvron et al., 2023) for prompt compression, achieving high accuracy at the expense of efficiency, making it less suitable for resource-limited scenarios requiring low latency. In contrast, encoder-based methods like LLMLingua-2 (Pan et al., 2024) offer a more lightweight structure that enhances efficiency but underperforms on tasks like question answering (QA). Our method bridges these gaps by enabling LLMLingua-2 to incorporate both task-aware and task-agnostic capabilities, significantly improving performance in QA and similar tasks.
- Data Efficiency. LLMLingua-2 uses data distillation with GPT-4 to create compressed datasets, incurring high financial and energy costs. Our method enhances data efficiency within the same framework as Pan et al. (2024). As illustrated in Figure 1, our method achieves a 4.8% improvement in F1-score on the LongBench (Bai et al., 2024) single-doc benchmark with only 1% more data at a 4x compression rate. When the additional data is slightly increased to 10%, the relative improvement markedly rises to 11.4%. Our approach boosts performance while minimizing additional data use, cutting API costs and energy consumption.

In summary, our EFPC offers an innovative and efficient solution to prompt compression, contributing to the practical advancements in NLP. The primary contributions of our work are as follows:

- Flexible Compression Framework: EFPC effectively bridges task-aware and task-agnostic approaches through a simple switch in the prompt prefix. This dual capability substantially improves the performance of tasks like document question answering.
- Data Efficiency: Our method is highly data-efficient and achieves significant improvements with minimal additional data, reducing the need for costly API usage and thereby saving both money and energy.
- Training and Inference Efficiency: Unlike traditional decoder-based methods, EFPC employs a lightweight encoder structure, leading to more efficient training and inference processes.
- Generalizability. We validated the effectiveness of our approach across various tasks and domains using multiple benchmarks. Moreover, our method is compatible with different compression models and target LLMs. It is designed for plug-and-play integration, demonstrating excellent generalization capabilities.

2 Related Works

Depending on whether task information is utilized, prompt compression methods can be classified as either task-aware or task-agnostic.

Task-aware compression adapts the context based on the downstream task or current query. For example, LongLLMLingua (Jiang et al., 2023c) uses a question-aware coarse-to-fine approach to estimate token information entropy, adjusting it according to the question. Reinforcement Learning (RL)-based methods (Jung and Kim, 2023; Huang et al., 2023) train models with reward signals from downstream tasks to achieve prompt compression. Soft prompt tuning methods (Wingate et al., 2022; Mu et al., 2023) typically require fine-tuning for specific tasks, while (Xu et al., 2024) trains a sum-

Our Instruction for Compression:
Compress some text to short expressions, and such that you (GPT-4) can reconstruct it as close as possible to the original. Unlike the usual text compression, I need you to comply with the 6 conditions below:
1. You can ONLY remove unimportant words.
2. Do not change the order of words.
3. Do not change the original words.
4. Do not use abbreviations or emojis.
5. Do not add new words or symbols
6. Ensure that the compressed text can answer the given question.

Compress the origin aggressively by removing words only.
Compress the origin as short as you can, while retaining as much information as possible to answer the given question. If you understand, the given question is *{given_question}*, please compress the following text:*{text_to_compress}*
The compressed text is:

Figure 2: Our instruction used for data distillation: sending user instructions and the original text to GPT-4 for compression, and the compressed text is required to complete the user instructions. The highlighted part shows the difference between our method and LLMLingua-2 (Pan et al., 2024) in data collection.

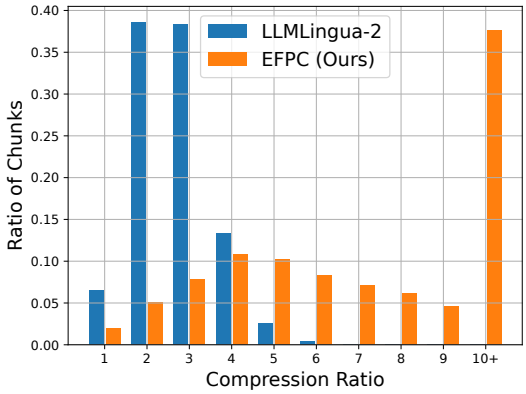

Figure 3: Compression ratio distribution on MeetingBank (Hu et al., 2023). It shows that our collected dataset achieves a much higher compression ratio (5 times more) compared to LLMLingua-2.

marization model to compress context based on the query. These task-aware approaches are often tailored to particular tasks and compression ratios, potentially limiting their applicability in real-world scenarios. Compared to these methods, our method can *flexibly switch between task-aware and task-agnostic modes* by simply setting prefix prompts, and can *dynamically select the compression rate*.

In contrast, task-agnostic methods compress prompts without considering specific tasks, making them versatile for various applications and black-box LLMs. However, generating compressed text that generalizes across different tasks is challenging. Information entropy-based metrics are commonly used to prune redundant information (Li et al., 2023; Jiang et al., 2023b), with a small language model estimating token importance. While these methods do not require training, they may not optimally capture token importance distributions for specific LLMs and often incur high computational costs. Summarization-based methods are also used (Chen et al., 2023; Packer et al., 2023) but often exclude essential details and struggle to generalize effectively. Recently, LLMLingua-2 (Pan et al., 2024) proposes a data distillation procedure to derive knowledge from an LLM to compress prompts without losing crucial information.

3 The Proposed Method

The overall framework of our method is shown in Figure 4. We begin with the data collection process in Sec. 3.1, followed by our training and inference algorithm in Sec. 3.2.

3.1 Instruction-Aware Data Collection

In this paper, we propose an instruction-aware data collection method for prompt compression. As shown in Figure 2, we send the user instruction together with the original text for compression and ask the LLM to generate compressed text which can complete the user task. Following Pan et al. (2024), we segment each long context into chunks, each containing no more than 512 tokens and ending with a period, and instruct GPT-4 to compress each chunk individually.

For fair comparison with LLMLingua-2, we utilize MeetingBank (Hu et al., 2023) as the source dataset. In Figure 3, we present a histogram of the compression ratio distribution for each chunk. The compression ratio is calculated as follows: LEN(text before compression)/LEN(text after compression), where LEN$(\cdot)$ denotes the length function. We compare our results with the dataset released by LLMLingua-2[1], which is task-agnostic. As shown in Figure 3, our instruction-aware data collection method achieves significantly higher compression rates. The average compression ratio of our dataset is 10.9, whereas the LLMLingua-2 dataset has an average ratio of 2.7. This outcome is expected, as fulfilling a specific task often requires only a subset of the original text's information.

For label generation, we first search for the corresponding word in the original prompt for each word in the compressed prompt. If a match is found, the word is assigned a label of 1; otherwise, it is assigned a label of 0.

[1]https://huggingface.co/datasets/microsoft/MeetingBank-LLMCompressed

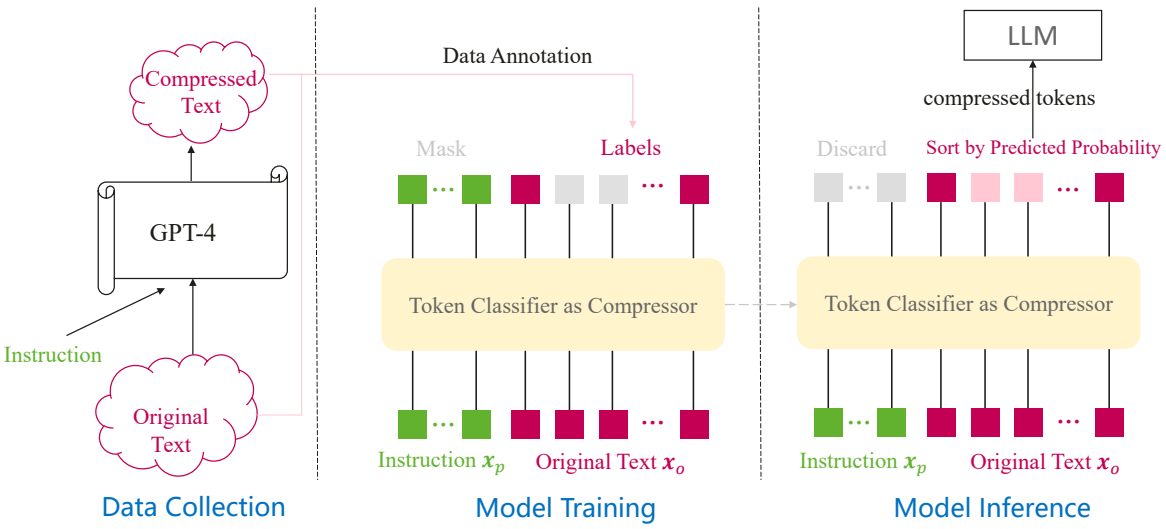

Figure 4: The proposed method. From left to right are the data collection, model training, and inference processes.

3.2 Flexible Token Classification Model

Architecture Following Pan et al. (2024), we adopt a Transformer encoder as the feature encoder f_θ and a linear classification layer on top. In this paper, we propose instruction-aware training and inference based on a token classification model. The input consists of two parts. The first part is an optional instruction (e.g., a question) composed of M words, $\boldsymbol{x}_p = \{x_i\}_{i=1}^{M}$, where $M = 0$ means that there is no extra instruction input. The second part is the original input text consisting of N words, $\boldsymbol{x}_o = \{x_i\}_{i=M+1}^{M+N}$. The final input concatenates $\boldsymbol{x}_p$ before $\boldsymbol{x}_o$ and the whole process is:

$$\boldsymbol{h} = f_\theta(\boldsymbol{x}), \tag{1}$$

$$p(x_i, \Theta) = \text{softmax}(W h_i + b), \tag{2}$$

where $\boldsymbol{x} = [\boldsymbol{x}_p, \boldsymbol{x}_o]$, $\boldsymbol{h} = \{h_i\}_{i=1}^{M+N}$ denotes feature vectors for all words, $p(x_i, \Theta) \in \mathbb{R}^2$ denotes the probability distribution of labels {preserve, discard} for the i-th word x_i, and $\Theta = \{\theta, W, b\}$ represent all the trainable parameters.

Loss Function We formulate the prompt compression task as a binary classification problem, using cross entropy (CE) loss for training. Let $\boldsymbol{y} = \{y_i\}_{i=1}^{M+N} \in \{0, 1\}$ be the labels corresponding to all words in $\boldsymbol{x}$, where $y_i = 0$ means the i-th word is discarded, and $y_i = 1$ means it is preserved.

The labels for the words from $i = M+1$ to $M+N$ are derived directly from the matching results between the original and compressed text. For the instruction part, there are two approaches: one way is to directly set the labels of the prepended part to zero, as it is not intended to be retained in the final output. The corresponding loss function is:

$$L_{\text{drop}}(\Theta) = \frac{1}{M+N} \sum_{i=1}^{M+N} \text{CE}(y_i; p(x_i, \Theta)), \tag{3}$$

where $\{y_i\}_{i=1}^{M} = 0$.

Alternatively, we can ignore this part of the output during training since it is not used when predicting. The corresponding loss function is:

$$L_{\text{mask}}(\Theta) = \frac{1}{N} \sum_{i=M+1}^{M+N} \text{CE}(y_i; p(x_i, \Theta)). \tag{4}$$

For a more intuitive comparison, we also consider the loss function of LLMLingua-2, which is task-agnostic and does not take additional $\boldsymbol{x}_p$ as input ($M = 0$). The loss function is defined as follows:

$$L_{\text{agnostic}}(\Theta) = \frac{1}{N} \sum_{i=1}^{N} \text{CE}(y_i; p(x_i, \Theta)). \tag{5}$$

Subsequent results show that both L_{mask} and L_{drop} outperform L_{agnostic}, proving the effectiveness of our methods. Notably, L_{mask} is superior to L_{drop} because it maintains consistency between tranining and inference, allowing the instruction part to guide the output of the original prompt part $\boldsymbol{x}_o$ without using its prediction probability.

Training Strategy To equip our model with both task-agnostic and task-aware capabilities, we propose two strategies:

Table 1: Data Efficiency of our method. The first row is uncompressed and the second row is LLMLingua-2.

Extra Training Data		Compression	LongBench Single-Doc QA				
Fraction	# Documents	Ratio	narrativeQA	qasper	multifield_en	multifield_zh	**AVG**
-	-	1×	22.87	41.37	52.60	60.37	44.30
0%	0	4.4×	16.01	38.75	43.48	44.16	35.16
1%	50		16.62	39.54	44.59	48.35	37.28
5%	250		14.89	42.42	48.59	50.50	39.10
10%	500		16.09	40.38	50.53	54.63	40.41
100%	5000		17.44	44.57	51.48	55.11	**42.15**

(1) Incremental training. Building on the pre-trained LLMLingua-2 model, we add extra instruction-aware compression data for training, as detailed in Sec. 3.1. Experimental results demonstrate that our method is highly data-efficient and can achieve significant improvements with only a small amount of additional data.

(2) Joint training. We combine task-agnostic data from LLMLingua-2 with our task-aware data. For fair comparisons, we keep the total data amount constant, adjusting only the ratio between the two.

Inference Stage Our method for compressing the initial prompt $\boldsymbol{x}_o = \{x_i\}_{i=M+1}^{M+N}$ to achieve a target compression ratio of $1/\tau$ involves three main steps. Here, τ represents the ratio of the number of words in the compressed prompt $\tilde{\boldsymbol{x}}_o$ relative to the number of words in the original prompt $\boldsymbol{x}_o$. First, we determine the target number of words/tokens to be maintained in the compressed prompt as $\tilde{N} = \tau N$. Next, the token classification model estimates the probability p_i for each word x_i being assigned the label "preserve". Finally, we select the top $\tilde{N}$ words from the original prompt $\boldsymbol{x}_o$ with the highest p_i values, maintaining their original sequence, to construct the compressed prompt $\tilde{\boldsymbol{x}}_o$.

Notably, our model can switch between task-agnostic and task-aware modes during inference by altering the prefix prompt. For instruction-aware tasks, such as question answering, we concatenate the user instruction, $\boldsymbol{x}_p$, with the original text $\boldsymbol{x}_o$ as the input. Conversely, when instruction-agnostic compression is required, we set $\boldsymbol{x}_p$ to null.

4 Experimental Results

We provide implementation details in Sec. 4.1 and evaluate the data efficiency in Sec. 4.2. Experiments on benchmark datasets are discussed in Sec. 4.3. In Sec. 4.4, we conduct ablation studies to analyze the impact of different components integrated into our method. All experiments were conducted using PyTorch on Tesla V100 GPUs.

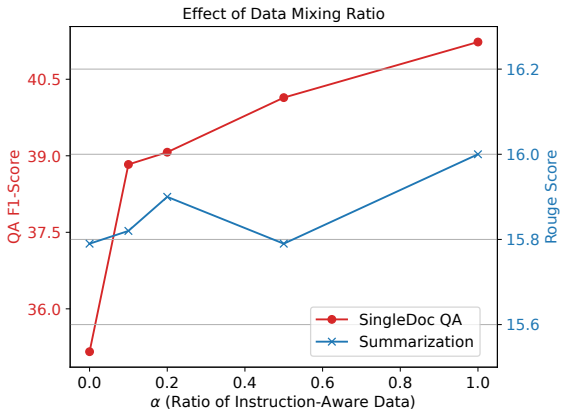

Figure 5: Data Efficiency of our method. We keep the same amount of training data and set target token to 3,000 during compression.

4.1 Implementation Details

Datasets To ensure a fair comparison with LLMLingua-2 (Pan et al., 2024), we construct our text compression dataset using training examples from MeetingBank (Hu et al., 2023) as the training set, as illustrated in Sec. 3.1. The compressed prompts are evaluated on two groups of datasets:

(i) **In-Domain:** We utilize the MeetingBank test set for in-domain evaluation. In addition to the summarization task, we adopt the QA task as outlined in (Pan et al., 2024). For both tasks, we used the same evaluation metrics as (Pan et al., 2024).

(ii) **Out-of-Domain:** For long-context scenarios, we use LongBench (Bai et al., 2024) and ZeroSCROLLS (Shaham et al., 2023), employing the same evaluation metrics as LongLLMLingua (Jiang et al., 2023c).

Training Details Our approach is implemented using Huggingface's Transformers and PyTorch 2.0.1 with CUDA-12.2. We use `multilingual-BERT` (Devlin et al., 2019) as the feature encoder f_θ in our compressor, fine-tuning both models for 10 epochs. We employe the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1e-5 and a batch size of 10. Unless specified

Table 2: In-domain evaluation of different methods on MeetingBank. Results marked with † are sourced from Pan et al. (2024), while those indicated with ◇ are reproduced using LLama-3.1-8b-instruct to ensure fair comparisons.

Backbone	QA	Summary					Length	
	F1 Score	BELU	Rouge1	Rouge2	RougeL	BERTScore	Tokens	$1/\tau$
Selective-Context (Li et al., 2023)†	66.28	10.83	39.21	18.73	27.67	84.48	1,222	2.5×
LLMLingua (Jiang et al., 2023b)†	67.52	8.94	37.98	14.08	26.58	86.42	1,176	2.5×
LLMLingua-2 (Pan et al., 2024)◇	77.26	15.65	41.47	18.94	30.41	86.50	984	3.0×
EFPC (Ours)◇	83.35	16.71	42.29	20.63	31.75	87.49	998	3.0×
Original◇	84.46	17.32	43.19	22.90	33.96	88.24	3,003	1.0×

Table 3: Out-of-domain evaluation on general long-context scenarios. Results with † come from Pan et al. (2024)

Methods	**LongBench**									**ZeroSCROLLS**		
	Single-Doc	Multi-Doc	Summ.	FewShot	Synth.	Code	**AVG**	Tokens	$1/\tau$	**AVG**	Tokens	$1/\tau$
2,000-token constraint												
Task(Question)-Agnostic Compression												
Selective-Context (2023)†	16.2	34.8	24.4	15.7	8.4	49.2	24.8	1,925	5×	19.4	1,865	5×
LLMLingua (2023b)†	22.4	32.1	24.5	61.2	10.4	56.8	34.6	1,950	5×	27.2	1,862	5×
LLMLingua2 (2024)†	29.8	33.1	25.3	66.4	21.3	58.9	39.1	1,954	5×	33.4	1,898	5×
SBERT (2019)†	33.8	35.9	25.9	23.5	18.0	17.8	25.8	1,947	5×	20.5	1,773	6×
OpenAI†	34.3	36.3	24.7	32.4	26.3	24.8	29.8	1,991	5×	20.6	1,784	5×
LongLLMLingua (2023c)†	39.0	42.2	27.4	69.3	53.8	56.6	48.0	1,809	6×	32.5	1,753	6×
EFPC (Ours)	41.7	42.2	25.8	67.3	27.0	57.6	43.6	1,972	5×	32.7	1,877	5×
3,000-token constraint												
Selective-Context (2023)†	23.3	39.2	25.0	23.8	27.5	53.1	32.0	3,328	3×	20.7	3,460	3×
LLMLingua (2023b)†	31.8	37.5	26.2	67.2	8.3	53.2	37.4	3,421	3×	30.7	3,366	3×
LLMLingua-2 (2024)†	35.5	38.7	26.3	69.6	21.4	62.8	42.4	3,392	3×	33.5	3,206	3×
SBERT (2019)†	35.3	37.4	26.7	63.4	51.0	34.5	41.4	3,399	3×	24.0	3,340	3×
OpenAI†	34.5	38.6	26.8	63.4	49.6	37.6	41.7	3,421	3×	22.4	3,362	3×
LongLLMLingua (2023c)†	40.7	46.2	27.2	70.6	53.0	55.2	48.8	3,283	3×	32.8	3,412	3×
EFPC (Ours)	42.9	46.6	26.9	70.4	32.5	59.7	46.5	3,415	3×	33.9	3,327	3×
Original Prompt†	39.7	38.7	26.5	67.0	37.8	54.2	44.0	10,295	-	34.7	9,788	-
Zero-Shot†	15.6	31.3	15.6	40.7	1.6	36.2	23.5	214	48×	10.8	32	306×

otherwise, due to resource constraints, all reported metrics use `multilingual-BERT` as the compressor and `Llama-3.1-8b-instruct` (Dubey et al., 2024) as the target LLM for downstream tasks. For fair comparison with mainstream methods in Table 3, we employ `GPT-3.5-Turbo-0613` as the target LLM for downstream tasks, with greedy decoding at a temperature of 0 to ensure enhanced stability across experiments.

4.2 Data Efficiency of Our Method

Before delving into detailed experimental results, it is essential to highlight our method's data efficiency. As discussed in Sec. 3.2, this is evaluated via two primary approaches: incremental training with small additional amounts of data and joint training with fixed data volume but varying proportions of instruction-aware data. As noted in Section 3.1, our collected data is instruction-aware (see Figure 2), while instruction-agnostic data comes from Pan et al. (2024).

In Table 1, we extend the training of the pretrained LLMLingua-2 model by adding a small amount of extra data. Different dataset ratios are employed to create subsets with a target token limit of 3,000 for compression, achieving an effective compression rate of approximately 4.4. As shown in Table 1, our EFPC exhibits remarkable data efficiency. With just a 1% increment in training data (around 50 additional documents from MeetingBank (Hu et al., 2023)), we see a 6% improvement in the F1-score on the LongBench single-doc QA benchmark. When the additional data ratio increases to 10%, the improvement surges to 14.9%.

In Figure 5, we keep the total training data volume constant while varying the ratio of instruction-aware data α (with the ratio of instruction-agnostic data being $1-\alpha$). As the proportion of our collected data increased, QA tasks consistently improve, while summarization tasks remain stable.

Experimental results in Table 1 and Figure 5 validate our method's effectiveness and data efficiency. Our approach achieves significant performance improvements with minimal additional data, highlighting its potential for efficient data utilization in various applications.

Query: How many underclassmen are on the NBA Draft Early-Entry List? **Answer:** 62.

Original Text: The Official 2006 NBA Draft Early-Entry List. 30 International Players, 62 underclassmen and one lone 5th year high school player make up this year's list, for a total of 93 early-entrants. Extensive commentary and early draft status projections are included.. For comparison, in 2005, 108 players declared (61 NCAA, 12 high school, 35 internationals), up from 94 in 2004, and 73 in 2003.\nThere were no major surprises on the early-entry list, besides a few .....

(a) Original Text

Compress Text: 2006 NBA Draft. 30 International Players 62 5th 93 entrants. 2005 108 61 NCAA 12 35 internationals 94 2004 73 2003. no surprises Holloway. Hrvoje Peric Fesenko. NCAA Joakim Noah Al Horford Corey Brewer Josh McRoberts Brandon Rush Tyler Hansbrough Marco Belinelli Tripkovic Goran Dragic. Akbar Abdul.....

Pred: At least 24 underclassmen are on the NBA Draft Early-Entry List.

(b) LLMLingua-2

Compress Text: 93 early - entrants. 2005 108 players 61 NCAA 12 high school 35 internationals 94 2004 73 2003. \n no major surprises early - entry mid - major Division II NAIA players 5th year high school player Clarence Holloway. Hrvoje Peric Renaldas Seibutis Kyrylo Fesenko. NCAA Joakim Noah Al Horford Corey Brewer Josh McRoberts Brandon Rush Tyler Hansbrough Marco Belinelli Uros Tripkovic.....

Pred: 53

(c) Ours-Task Agnostic

Compress Text: Official 2006 NBA Draft Early - Entry List. 30 International Players, 62 underclassmen and one 5th year high school player ' for 93 early - entrants. draft projections included. in 2005 108 players declared 61 NCAA 12 high school 35 internationals from 94 in 2004 73 in 2003. no major surprises on early - entry list mid – II NAIA players 5th year high school player Clarence Holloway.....

Pred: 62

(d) Ours-Task Aware

Figure 6: Comparison of compressed text by different methods. (a) Original text. (b) LLMLingua-2. (c) Our method without task-aware inference. (d) Our method with task-aware inference. It can be observed that our method is able to retain more information relevant to the problem, thereby enabling the LLM to generate correct answers.

Table 4: Ablation Study of task-aware design. For fair comparison, we use 3000-token constraint here.

Task-Aware Training?	Task-Aware Inference?	LongBench				
		Single-Doc	Multi-Doc	Summ.	Few-Shot	**AVG**
×	×	35.16	31.13	15.79	33.88	28.99
×	✓	36.51	31.98	15.70	34.44	29.66
✓	×	35.64	31.99	15.97	37.01	30.15
✓	✓	**42.15**	**36.59**	**16.12**	**38.99**	**33.46**

4.3 Experiments on Benchmark Datasets

Results on In-Domain Benchmark In Table 2, our proposed method surpasses strong baselines on MeetingBank. Despite the fact that our compressors are much maller than the LLaMa-2-7B (Touvron et al., 2023) used in the baselines, our approach achieves significantly better performance on both the QA and Summary tasks, and nearly matches the original prompt's performance. This demonstrates the effectiveness of our constructed dataset and the benefits of optimizing the compression model using prompt compression knowledge.

Results on Out-of-Domain Benchmark As shown in Table 3, EFPC outperforms LLMLingua-2, which is based on the same lightweight model, by 39.9% on single-doc dataset and 20.8% on multi-doc dataset at a 5× compression ratio. Our method also achieves comparable results to LongLLMLingua (Jiang et al., 2023c), which is based on a much heavier decoder structure. Notably, LongLLMLingua incurs higher overhead in terms of memory, latency, and power consumption during inference, as will be shown in Sec. 4.6.

Moreover, as the compression ratio increases from 3× to 5×, LLMLingua2 exhibits a performance drop of 16.1% on single-doc QA, whereas our method only experiences a decline of 2.8%. This indicates that our approach is more effective at higher compression ratios, showcasing a stronger ability to preserve essential information.

Table 5: Ablation study of loss function. We use 50% extra data with 3000-token constraint here.

Loss Function	LongBench				
	Single-Doc	Multi-Doc	Summ.	Few-Shot	**AVG**
L_{agnostic}	35.16	31.13	15.79	33.88	28.99
L_{drop}	41.40	35.78	15.98	36.14	32.33
L_{mask} (Ours)	**42.15**	**36.59**	**16.12**	**38.99**	**33.46**

4.4 Ablation Study

In this section, we conduct ablation experiments to evaluate the impact of different components in our proposed method (Table 4) and analyze the effect of different loss functions (Table 5).

Task-aware Training and Inference We examine the effects of task-aware training and inference. Task-aware inference involves prepending the task instruction to the input (e.g., a query for QA datasets) during inference. Table 4 shows that without task-aware training or inference (first row), the method degrades to LLMLingua-2. When both are used (last row), it represents our EFPC method. Results indicate that task-aware inference consistently improves accuracy, regardless of whether task-aware training is used. Moreover, the combination of both yields the most significant benefits. For example, using task-aware inference without task-aware training improves relative accuracy by 3.84% and 2.73% for single-doc and multi-doc QA datasets, respectively. When both are employed (our EFPC), relative improvements increase to 10.95% and 10.24%.

Table 6: Evaluation with Mistral-7B as the Target LLM on MeetingBank and LongBench single-doc QA task. We report Rouge1 (Lin, 2004) for summary. †: Results from Pan et al. (2024).

Methods	MeetingBank				LongBench Single-Doc					
	QA	Summ.	Tokens	$1/\tau$	2,000-token cons.	Tokens	$1/\tau$	3,000-token cons.	Tokens	$1/\tau$
Selective-Context (2023)†	58.13	26.84	1,222	2.5×	22.0	2,038	7.1×	26.0	3,075	4.7×
LLMLingua (2023b)†	50.45	23.63	1,176	2.5×	19.5	2,054	7.1×	20.8	3,076	4.7×
LLMLingua-2 (2024)†	75.97	29.93	984	3.0×	25.3	1,949	7.4×	27.9	2,888	5.0×
LLMLingua-2-Large†	76.22	30.18	970	3.0×	26.8	1,967	7.4×	27.3	2,853	5.1×
EFPC (Ours)	80.51	29.72	992	3.0×	29.7	1,994	7.3×	30.2	2,850	5.1×
EFPC-Large (Ours)	**81.63**	**30.43**	985	3.0×	**31.8**	2,002	7.2×	**31.9**	2,913	5.0×
Original Prompt†	66.95	26.26	3,003	-	24.5	14,511	-	24.5	14,511	-

In Figure 6, we compare the compressed texts using different methods (with the 2,000-token constraint for all). Both columns (b) and (c), which do not employ task-aware inference, result in compressed texts that lose critical information required to answer user questions. For instance, column (b) omits "underclassmen". In contrast, our method in column (d) retains all essential information. It is worth noting that the same compression ratio is used in columns (b), (c), and (d). Notably, column (d) demonstrates a variable compression pattern: key regions are compressed less, while non-essential regions are compressed more.

Loss Function In Table 5, we compare different loss functions from Sec. 3.2. Both L_{mask} and L_{drop} outperform L_{agnostic}, which is equivalent to LLMLingua-2 (Pan et al., 2024). Comparing L_{drop} and L_{mask}, we see that ignoring the labels and predictions of the instruction part during training yields better results. This is because it ensures consistency between the training and inference stages.

In summary, the experiments in Table 4 and Figure 6 show that both task-aware training and inference are effective. Combining both (ensuring consistency between training and inference) yields the highest gains. Additionally, Table 5 demonstrates the effectiveness of the proposed L_{mask}.

4.5 Generalizability

In this section, we validate our method's generalizability across different model architectures, including prompt compression model and LLM used for generating responses. Previously, we tested primarily on the bert-base model, but here we extend our evaluation to `xlm-roberta-large` (Conneau et al., 2020) and use `Mistral-7b-v0.1` (Jiang et al., 2023a) as the target LLM.

As shown in Table 6, on the in-domain dataset MeetingBank QA, our method achieve absolute improvements of 4.5% and 5.4% over the best baseline method, LLMLingua-2, when using `multilingual-BERT` and `xlm-roberta-large`, respectively. This trend is consistent in out-of-domain evaluations as well. The results in Table 6 demonstrate that our method is robust across various backbones, tasks, domains and target LLMs.

Table 7: Efficiency comparison on MeetingBank. ◇: We re-evaluate these metrics on Tesla V100 GPU. †: Results from Pan et al. (2024).

$1/\tau$	1×	2×	3×	5×	GPU Mem. (GB)
End2End w/o Compression◇	15.8				
End2End w/ EFPC◇	-	10.0	8.8	6.7	
Selective-Context†	-	15.9	15.6	15.5	26.5
LLMLingua†	-	2.9	2.1	1.5	16.6
LLMLingua-2†	-	0.5	0.4	0.4	2.1
LLMLingua-2◇	-	0.4	0.4	0.4	2.1
EFPC◇	-	0.4	0.4	0.4	2.1

4.6 Latency Comparison

Table 7 presents the latency and GPU memory usage of different methods with various compression rates. Sharing the same lightweight architecture, our method shows nearly the same latency as LLMLingua2. Compared to other compression techniques, EFPC results in significantly lower computational overhead and achieves an end-to-end speedup of 1.6× to 2.4×. Additionally, our approach can reduce GPU memory costs by up to 8×, decreasing the demand for hardware resources.

5 Conclusions

In this paper, we proposed an efficient and flexible prompt compression method EFPC, for improving data, training, and inference efficiency. We identified the challenges faced by existing methods and addressed them accordingly. With only a single model and once training, our method supports both task-aware and task-agnostic modes, allowing flexible switching between both modes as needed. Extensive experiments across various tasks and domains demonstrate that our approach significantly outperforms other baseline methods in terms of performance and compression latency.

Limitations

While our proposed Efficient and Flexible Prompt Compression (EFPC) method demonstrates significant improvements in efficiency and performance, several limitations and areas for future work remain noteworthy.

First, although we emphasize the data efficiency of our method, the training data for our experiments was exclusively sourced from MeetingBank. This was done to ensure a fair comparison with existing methods. While this limitation allowed us to demonstrate notable improvements and generalizability on out-of-domain datasets, we did not explore the potential gains that could be achieved with larger-scale training datasets. Future research could investigate the impact of training with more extensive datasets, which may reveal additional performance enhancements and insights into the scalability of our method.

Second, due to resource constraints, our experiments utilized an 8-billion parameter large language model (LLM). We did not systematically explore how our prompt compression method performs across different sizes of LLMs. Intuitively, larger LLMs might handle prompt compression more effectively, potentially resulting in smaller accuracy losses at higher compression rates. Exploring the sensitivity of our method to various LLM sizes could provide a deeper understanding of its adaptability and robustness, offering valuable guidance for future deployments in diverse computing environments.

Lastly, EFPC, while designed for flexibility and efficiency, may still face challenges when applied to extremely high-compression scenarios or highly specialized tasks that were not covered in our experimental setup. Future research should explore these scenarios to further enhance the robustness and versatility of our approach.

In summary, while EFPC represents a significant step forward in the realm of prompt compression, addressing these limitations could unlock further potential and showcase its applicability across an even broader range of tasks and models. We encourage future work to explore these dimensions to fully realize the capabilities of prompt compression in advancing natural language processing.

References

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. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, page 3119–3137.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzman, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, page 8440–8451.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, pages 4171—-4186.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yebowen Hu, Tim Ganter, Hanieh Deilamsalehy, Franck Dernoncourt, Hassan Foroosh, and Fei Liu. 2023. Meetingbank: A benchmark dataset for meeting summarization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, page 16409–16423.

Xijie Huang, Li Lyna Zhang, Kwang-Ting Cheng, Fan Yang, and Mao Yang. 2023. Fewer is more: Boosting llm reasoning with reinforced context pruning. *arXiv preprint arXiv:2312.08901*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. Llmlingua: Compressing

prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023c. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*.

Hoyoun Jung and Kyung-Joong Kim. 2023. Discrete prompt compression with reinforcement learning. *arXiv preprint arXiv:2308.08578*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *The International Conference on Learning Representations*, pages 1–15.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, pages 9459–9474.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, page 6342–6353.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, page 74–81.

Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, pages 19327–19352.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

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. *arXiv preprint arXiv:2403.12968*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bertnetworks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, page 3982–3992.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. Zeroscrolls: A zero-shot benchmark for long text understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, page 7977–7989.

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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

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

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. 2022. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, page 5621–5634.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, pages 1–14.