Perception Compressor: A Training-Free Prompt Compression Framework in Long Context Scenarios

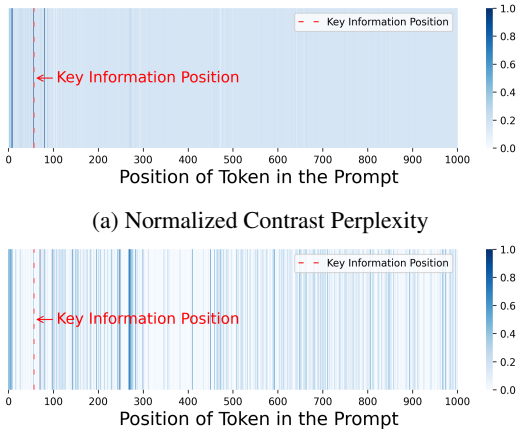
Jiwei Tang[1], Jin Xu[3], Tingwei Lu[1], Zhicheng Zhang[1], Yiming Zhao[4], Lin Hai[1], Hai-Tao Zheng[1,2]*,
[1]Shenzhen International Graduate School, Tsinghua University
[2]Pengcheng Laboratory
[3]Ant Group
[4]School of Artificial Intelligence, Sun Yat-sen University
tangjw24@mails.tsinghua.edu.cn

Abstract

Large language models (LLMs) demonstrate exceptional capabilities in various scenarios. However, they suffer from much redundant information and are sensitive to the position of key information in long context scenarios. To address these challenges, we present *Perception Compressor*, a training-free prompt compression framework. It includes a perception retriever that leverages guiding questions and instruction to retrieve the most relevant demonstrations, a dual-slope ratio allocator to dynamically allocate compression ratios and open-book ratios, and a semi-guided iterative compression that retains key information at the token level while removing tokens that distract the LLM. We conduct extensive experiments on long context benchmarks, *i.e.*, NaturalQuestions, LongBench, and MuSiQue. Experiment results show that *Perception Compressor* outperforms existing methods by a large margin, achieving state-of-the-art performance. [1]

1 Introduction

Large language models (LLMs) (*e.g.*, ChatGPT), known for their powerful generation and reasoning capabilities, have been widely applied in various scenarios. Commonly used methods like Chain-of-Thought (CoT) (Wei et al., 2022), In-Context Learning (ICL) (Dong et al., 2022), Retrieval Augment Generation (RAG) (Lewis et al., 2020), effectively enhance the performance of LLMs by providing domain knowledge. However, these methods generally yield long prompts, even over thousands of tokens, *i.e.*, long context. There are two challenges when using LLMs in long context scenarios: (1) Long prompts inherently contain much redundant information (Shannon, 1951; Shi et al., 2023), and may exceed the window size of LLMs. (2)



(a) Normalized Contrast Perplexity

(b) Normalized Perplexity

Figure 1: Perplexity v.s. Contrast Perplexity. Only a very small number of tokens related to key information have a high contrast perplexity, while the contrast perplexity of other tokens is nearly the same. However, the perplexity of different tokens varies significantly.

LLM is sensitive to the position of key information (relevant to the input question), *i.e.*, the *lost in the middle* challenge presented in Liu et al. (2024).

Some previous research (Nijkamp et al., 2023; Han et al., 2023; Peng et al., 2023; Sun et al., 2023; Ding et al., 2023) extends the window size. However, while these methods increase the input token length, they struggle to overcome the notable decline in performance (Ge et al., 2023). A prompt can be divided into different components (*e.g.* instruction, demonstrations, and question). To address challenge one, other researchers focus on prompt compression methods. Among them, SelectiveContext (Li et al., 2023), LLMLingua (Jiang et al., 2023a), and LLMLingua2 (Pan et al., 2024) are task-agnostic methods, that ignore the input question during the compression process, potentially losing key information and failing to address challenge two in long context scenarios.

To enhance the density of key information and address challenge two, Jiang et al. (2024) intro-

*Corresponding author: zheng.haitao@sz.tsinghua.edu.cn
[1]Our code can be available at https://github.com/Twilightaaa/PerceptionCompressor.

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 4093–4108

duce LongLLMLingua. LLM can perceive the position of key information tokens (KITs) in prompt through its contrast perplexity (Jiang et al., 2024), *i.e.*, tokens with high contrast perplexity contain key information (see Figure 1a). LongLLMLingua first reorders and retrieves the demonstrations based on their relevance to the input question, then performs token-level compression based on contrast perplexity, removing tokens with lower contrast perplexity.

However, this method has two limitations: (1) In the retrieval stage, this method not only ignores that the input question can be complex and difficult to understand, making it challenging to retrieve the most relevant demonstrations in high-noise long context scenarios but also neglects the impact of the instruction contained in the original prompt. The instruction contains all the guidance to generate answers that can significantly impact LLM's performance. (2) In the token-level compression stage, LongLLMLingua retains tokens with high contrast perplexity. While this preserves KITs with high contrast perplexity, it neglects the process of non-key information tokens (NITs). The number of NITs is significantly greater than that of KITs, and the perplexity of NITs is nearly the same (see Figure 1a). This implies that once all KITs are retained, the selection of a large number of NITs is almost random. Irrelevant content can distract LLMs (Shi et al., 2023). Randomly selecting NITs with vastly different perplexities (see Figure 1b) can lead to the retention of distracting content, leading to much noise in the compressed prompt.

Inspired by these observations, we present *Perception Compressor*, a training-free prompt compression framework, to address the challenges and limitations mentioned above. Specifically, we introduce a perception retriever that leverages instruction and guiding questions to retrieve demonstrations. Socratic Method is a philosophical inquiry technique that promotes deep understanding through questioning. This method is named after the ancient Greek philosopher Socrates, who was renowned for his dialogic teaching style. The essence of the Socratic Method lies in guiding students or participants to discover knowledge, clarify their thoughts, identify assumptions, and ultimately achieve a deeper level of understanding through a series of guiding questions (Benson, 2011). We hope to activate the "Scratic Thinking" of LLMs through guiding questions, enabling them to identify underlying assumptions and prerequisites step-by-step, thus guiding them to focus on key information in high-noise long context scenarios. Next, we present a dual-slope ratio allocator to allocate all compression ratios and open-book ratios in the compression process. Knowledge with high perplexity is more uncertain for LLMs, hence tokens with high perplexity in NITs are more likely to distract the LLMs. Therefore, we utilize semi-guided iterative compression based on the open-book ratios to remove high perplexity tokens in NITs while retaining KITs.

Our main contributions are four-fold: (1)We introduce a perception retriever, which achieves the highest recall@1 in the NaturalQuestions recall experiment (see Table 1). And we use Bayes' Theorem to prove the rationality of perception retriever. (2)We present a dual-slope ratio allocator, which dynamically balances the compression ratios and the open-book ratios based on the varying relevance of each demonstration to the input question. (3)We propose semi-guided iterative compression to perform token-level compression, which removes NITs that distracts LLM while retaining KITs. (4)We conduct extensive experiments and comprehensive analysis on benchmarks in long context scenarios, *i.e.*, NaturalQuestions, LongBench, and MuSiQue. The experiment results demonstrate the superiority and effectiveness of *Perception Compressor*.

2 Related Work

2.1 Prompt Compression Methods

As the input context gets longer, methods for compressing prompts receive widespread attention. Recently, prompt compression methods can mainly be divided into two categories: (1) Generating soft prompts. These methods (Li et al., 2024; Cao et al., 2024; Rau et al., 2024; Chuang et al., 2024; Chevalier et al., 2023; Zhang et al., 2024) mainly focus on learning soft tokens to reduce the length of input tokens, which requires further parameter fine-tuning of LLM, long training time, and high computational cost. Moreover, LLMs trained by these methods are generally domain-specific, with relatively poor generalization. (2) Explicitly compress prompt. Recently, some methods based on information theory have emerged, such as Selective Context (Li et al., 2023), LLMLingua (Jiang et al., 2023a). To use bidirectional context, Pan et al. (2024) introduce LLMLingua2. However, these methods ignore the relevance of the content

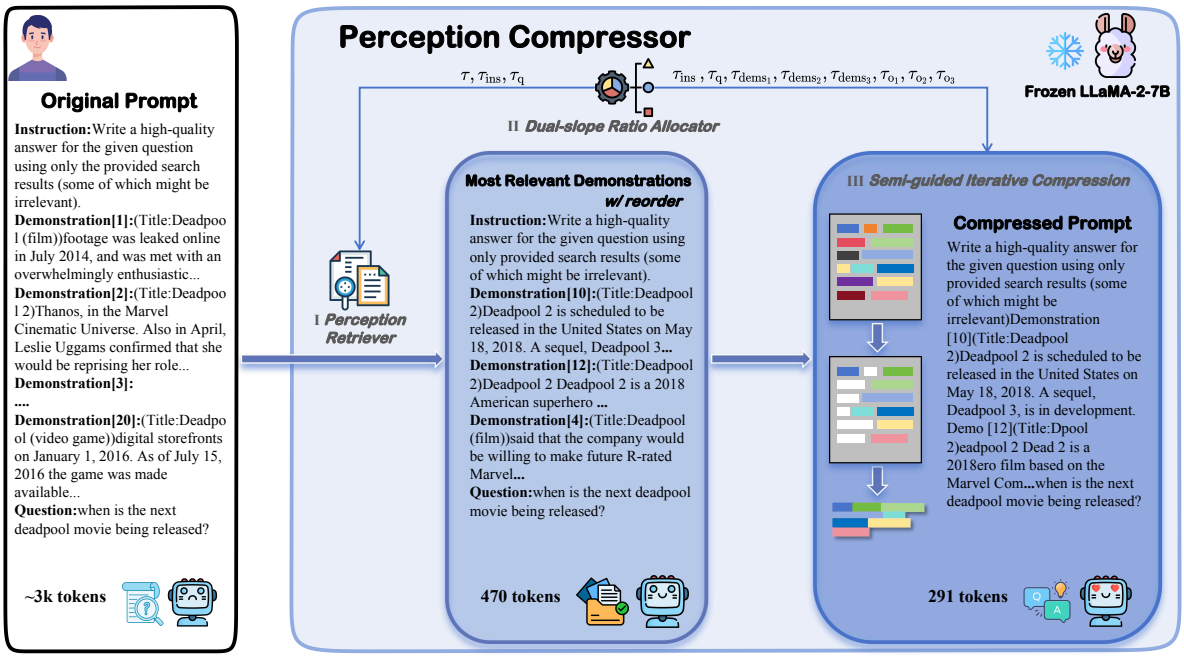


Figure 2: Framework of *Perception Compressor*. The original prompt can be divided into instruction, demonstrations, and question. *Perception Compressor* first uses the perception retriever to retrieve the most relevant demonstrations and reorders them from most to least relevant to the input question. Then, it performs a semi-guided iterative compression to obtain the final compressed prompt. The entire process is controlled by the compression ratios and open-book ratios allocated by the dual-slope ratio allocator.

to the input question. To increase the density of key information, Jiang et al. (2024) propose LongLLMLingua, which firstly reorders the demonstrations based on their relevance to the input question, keeping the most relevant demonstrations. Then use contrast perplexity ITPC to perform token-level compression. Another common method is to generate a summary of the given context, which requires further training of the LLM (Xu et al., 2023; Wang et al., 2023).

2.2 Extending Window Size

Recently, a series of studies have been proposed for extending context window size. This extension has been pursued through various avenues. One strategy is staged pre-training (Nijkamp et al., 2023), where the context window size gradually increases over successive pre-training stages. Some studies involve interpolating position embeddings (Han et al., 2023; Peng et al., 2023), aiming to adapt these embeddings to accommodate longer context. Furthermore, researchers have delved into adjusting attention mechanisms to handle expansive context windows more efficiently. This includes exploring linear or sparse attention mechanisms (Ding et al., 2023; Sun et al., 2023). Additionally, there is a line of study that revolves around leveraging external memory modules for storing context (Bertsch et al., 2023; Tworkowski et al., 2024). These modules alleviate the burden on the LLM by offloading context information to dedicated memory units.

However, while these methods offer promising avenues for extending the context window of LLMs, they generally lead to relatively inferior performance.

2.3 Retrieval Methods

The retrieval methods can be divided into two categories: (1) Sparse retrieval. Sparse retrieval retrieves the most similar items from an index based on query keywords or feature vectors, such as BM25. (2) Dense retrieval. Dense retrieval methods (Xiao et al., 2023; Reimers and Gurevych, 2019; Jiang et al., 2023b; Günther et al., 2023) involve training retrieval models on large corpora, generating dense vector representations for queries and demonstrations through model inference, and computing similarities based on these vector representations.

3 Task Formulation

Given a original prompt with augmented context $\mathbf{x} = (\mathbf{x}^{\text{ins}}, \mathbf{x}^{\text{dems}_1}, ..., \mathbf{x}^{\text{dems}_N}, \mathbf{x}^{\text{q}})$, which con-

Methods	Recall																			
	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10	@11	@12	@13	@14	@15	@16	@17	@18	@19	@20
LLMLingua	4.4	7.1	9.3	12.2	14.8	17.6	20.6	23.0	25.9	29.3	32.6	36.1	39.5	43.2	47.4	52.7	58.4	66.3	77.6	100.0
BM25	8.0	13.9	19.6	24.6	29.4	34.2	38.5	42.7	46.7	52.1	56.4	60.9	64.9	69.2	73.5	77.7	82.6	87.3	92.7	100.0
Bge	35.6	49.5	57.8	63.8	69.0	72.9	76.4	79.4	82.1	84.4	86.3	88.2	89.6	91.1	92.4	93.8	95.3	96.7	98.2	100.0
Gzip	50.1	55.7	59.7	63.0	65.8	68.5	70.6	72.8	74.7	77.2	79.8	81.7	84.1	85.9	87.6	89.3	91.3	93.3	95.6	100.0
Jina	52.2	66.0	74.0	78.6	82.2	85.4	87.6	89.2	91.0	92.2	93.3	94.5	95.4	96.5	97.0	97.6	98.4	99.0	99.4	100.0
OpenAI Embedding	52.9	67.0	75.9	81.5	85.7	88.3	90.6	92.1	93.2	94.1	95.4	96.3	97.1	97.8	98.2	98.5	98.8	99.2	99.6	100.0
SentenceBert	54.7	66.6	73.0	77.5	81.3	84.3	86.5	88.1	89.7	91.5	92.8	93.9	94.9	95.5	96.2	96.8	97.5	98.3	98.9	100.0
BgeLLMembedder	59.9	73.3	80.3	85.1	87.8	89.9	91.5	92.8	93.6	94.6	95.7	96.5	97.2	97.6	98.1	98.4	98.7	99.0	99.6	100.0
BgeReranker	62.9	74.5	80.0	83.4	85.8	88.2	89.8	91.5	92.8	93.9	94.9	95.9	96.5	97.5	97.9	98.2	98.8	99.3	99.5	100.0
LongLLMLingua r_k	67.1	78.1	82.9	86.0	88.7	90.5	91.6	93.0	94.2	95.4	96.3	97.1	97.6	**98.2**	**98.8**	99.0	**99.4**	**99.7**	**99.9**	100.0
Perception retriever	**72.3**	**81.7**	**86.1**	**88.3**	**90.2**	**91.7**	**92.9**	**94.2**	**95.0**	**95.8**	**96.4**	**97.2**	**97.6**	98.1	98.5	**99.0**	99.2	99.6	99.8	**100.0**

Table 1: Comparison of recall on NaturalQuestions (20 documents) (Liu et al., 2024). We use the recall rate of ground truth document as the evaluation metric.

sists of the instruction $\mathbf{x}^{\text{ins}}$, N demonstrations $\{\mathbf{x}^{\text{dems}_i}\}_{i=1}^{N}$, and the input question $\mathbf{x}^{\text{q}}$. A compression ratio $1/\tau$, $\tau \in [0,1]$. The objective of prompt compression can be formulated as:

$$\min_{\widetilde{x},\tau} d\left(p\left(\widetilde{\mathbf{y}} \mid \widetilde{\mathbf{x}}\right), p\left(\mathbf{y} \mid \mathbf{x}\right)\right) \tag{1}$$

where $d(\cdot,\cdot)$ is a function measuring the distance between two distributions (*e.g.*, KL divergence); $\widetilde{y}$ represents the LLM-generated results derived by compressed prompt $\widetilde{x}$ and y denotes the ground-truth answer. The distribution of $\widetilde{y}$ is expected to be as similar to y as possible.

4 Method

In this section, we elaborate on our proposed prompt compression framework, *i.e.*, *Perception Compressor*. The overall framework and entire process are shown in Figure 2.

4.1 Perception Retriever

Generate Guiding Questions We first employ an LLM (*e.g.*, ChatGPT) to generate guiding questions based on the input question. Specifically, given a question, we prompt the LLM to generate a set of guiding questions.

$$LLM(q_0) = \{q_1, q_2, \ldots, q_n\} \tag{2}$$

where we use prompt[2] to guide LLM in generating guiding questions; q_0 is the input question; $\{q_1, q_2, \ldots, q_n\}$ is the generated n guiding questions.

Calculate Semantic Similarity We use SentenceBert to calculate the semantic similarity between q_0 and $q_0, ..., q_n$.

$$e_i = F(q_i) \tag{3}$$

where $F(\cdot)$ is semantic feature extractor, *i.e.*, SentenceBert; e_i, where $i \in \{0, 1, \ldots, n\}$, is the semantic feature vector corresponding to question q_i. Then we can define the semantic similarity w_i between q_0 and q_i as:

$$w_i = \frac{e_0 \cdot e_i}{\|e_0\| \cdot \|e_i\|} \tag{4}$$

Perception Perplexity We first calculate the condition perplexity $r_{k,j}$ for each demonstration $\mathbf{x}^{\text{dems}_k}, k \in \{1, \ldots, N\}$, where N is the number of all demonstrations.

$$r_{k,j} = \sum_{i=1}^{L_{ins}+L_{q_j}+L_r} g\left(\mathbf{x}_i^{\text{con}_j}\right) \log p\left(\mathbf{x}_i^{\text{con}_j} \mid \mathbf{x}^{\text{dems}_k}\right) \tag{5}$$

where $g(\cdot)$ represents the probability distribution of the ground truth; $\mathbf{x}^{\text{con}_j}$ is the concatenation of $\mathbf{x}^{\text{ins}}$, $\mathbf{x}^{\text{q}_j}$ and $\mathbf{x}^{\text{r}}$; $\mathbf{x}_i^{\text{con}_j}$ denotes the i-th token in $\mathbf{x}^{\text{con}_j}$; $\mathbf{x}^{\text{dems}_k}$ and $\mathbf{x}^{\text{r}}$ refer the k-th demonstration and regularization constraint[3], respectively.

Then, we can obtain perception perplexity r_k of demonstration $\mathbf{x}^{\text{dems}_k}$.

$$r_k = \sum_{j=0}^{n} w_j \cdot r_{k,j} \tag{6}$$

Through the derivation in Appendix A, we can obtain:

$$\{r_k\}_i \propto \log \prod_{j=0}^{n} p\left(\mathbf{x}^{\text{dems}_k} \mid \mathbf{x}_i^{\text{con}_j}\right)^{w_j} \tag{7}$$

where $\{r_k\}_i$ is the impact of the i-th token in $\mathbf{x}^{\text{con}_j}$ on the perception perplexity.

Observing Equation (7), we can derive two insights: (1)w_j is a scaling factor. The more relevant q_j is to the q_0, and the greater its impact on $\{r_k\}_i$. (2)$\{r_k\}_i$ is proportional to the log-likelihood function (Fisher, 1922) of the conditional probability

[2]Specifically, "*Please provide **n** most helpful guiding questions to address the original question: {original question}*"

[3]For fair comparison, We use the same regularization constraint in Jiang et al. (2024) to strengthen the connection between $\mathbf{x}^{\text{dems}_k}$ and $\mathbf{x}^{\text{q}}$.

given by $\mathbf{x}_i^{\text{con}_j}$, where j ranges from 0 to n. This indicates that $\{r_k\}_i$ reflects the relevance of the current demonstration $\mathbf{x}^{\text{dems}_k}$ to the entire set of $\mathbf{x}_i^{\text{con}_j}$. The larger $\{r_k\}_i$ is, the stronger the relevance.

Reorder the Demonstrations We reorder the demonstrations based on perception perplexity, from high to low. We retain $\mathbf{x}^{\text{dems}_k}$ with higher r_k as retrieval results until the total token length meets the compression ratio constraint.

4.2 Dual-slope Ratio Allocator

Different components of the prompt (*e.g.* instruction, demonstrations, question) have varying degrees of redundancy. Obviously, the demonstrations have the highest degree of redundancy, so higher compression ratios should be allocated to them.

We need to predefine τ, τ_{ins} and τ_{q} and set the coarse-grained control coefficient μ to determine the number of retrieved demonstrations. Then, we can get the basic compression ratio for each demonstration.

$$\tau_{\text{dems}} = \frac{L'_{\text{dems}} - (\mu\tau L - \tau_{\text{ins}} L_{\text{ins}} - \tau_{\text{q}} L_{\text{q}})}{L'_{\text{dems}}} \quad (8)$$

where L is the token length of the original prompt; L'_{docs} denotes the token length of all retained demonstrations; L_{q} and L_{ins} are the token length of the input question and instruction, respectively.

To assign a lower compression ratio to demonstrations that are more relevant to the input question, we introduce the first slope k_1.

$$\tau_{\text{dems}_k} = max(min(1, \tau_{\text{dems}} + (1 - \frac{2\text{rank}(r_k)}{N_d}) \cdot k_1), 0) \quad (9)$$

where $k_1 > 0$; $rank(\cdot)$ is the ranking index of perception perplexity r_k (*e.g.*, 0, 1);N_d represents the number of demonstrations retained by the Perception retriever.

τ_o is the open-book ratio, which is the budget for considering the contrast perplexity. When the compression ratio of a demonstration is high, we should prioritize the preservation of key information with high contrast perplexity. Therefore, we introduce k_2.

$$\tau_{\text{o}_k} = max(min(1, \tau_{\text{o}} - (1 - \frac{2\text{rank}(r_k)}{N_d}) \cdot k_2), 0) \quad (10)$$

where $k_2 > 0$.

4.3 Semi-guided Iterative Compression

Because LLM's context window size is of limited length, and perplexity may model local information instead of catching long-range dependency (Hu et al., 2024), we divide the complete context into several segments $s = \{s_1, s_2, ...s_m\}$ and then compress them sequentially from front to back.

We use conditional probability modeling for compression, which can be formulated as:

$$p(\widetilde{\boldsymbol{s}}_j) \approx \prod_{i=1}^{L_{s,j}+\sum_k^{j-1}\widetilde{L}_{s,k}} p(s_{j,i}|\widetilde{s}_{<j}, s_{j,<i}) \quad (11)$$

where $s_{j,i}$ denotes the i-th token in the j-th segment; $L_{s,j}$ and $\widetilde{L}_{s,j}$ represent the token length of j-th original and compressed segment, respectively.

The conditional contrast perplexity for each segment is:

$$\begin{aligned} Q(s_{j,i}) &= g(s_{j,i}) \log p(s_{j,i} \mid q_0, \widetilde{s}_{<j}, s_{j,<i}) \\ &\quad - g(s_{j,i}) \log p(s_{j,i} \mid \widetilde{s}_{<j}, s_{j,<i}) \\ &= g(s_{j,i}) \log \frac{p(s_{j,i} \mid q_0, \widetilde{s}_{<j}, s_{j,<i})}{p(s_{j,i} \mid \widetilde{s}_{<j}, s_{j,<i})} \end{aligned} \quad (12)$$

The conditional perplexity can be formulated as:

$$P(s_{j,i}) = g(s_{j,i}) \log p(s_{j,i} \mid \widetilde{s}_{<j}, s_{j,<i}) \quad (13)$$

When the contrast perplexity $Q(s_j)$ and perplexity $P(s_j)$ for each token in segment are obtained, the thresholds γ_j^Q and γ_j^P are dynamically calculated based on the retention ratios $\tau_{\boldsymbol{s}_j} * \tau_o$ and $\tau_{\boldsymbol{s}_j} * (1 - \tau_o)$, respectively.

$$\tau_{\boldsymbol{s}_j} = \begin{cases} \tau_{\text{ins}}, & \text{if } \boldsymbol{s}_j \in \boldsymbol{x}^{\text{ins}}, \\ \tau_{\text{dems}}, & \text{if } \boldsymbol{s}_j \in \boldsymbol{x}^{\text{dems}}, \\ \tau_{\text{q}}, & \text{if } \boldsymbol{s}_j \in \boldsymbol{x}^{\text{q}}. \end{cases} \quad (14)$$

First, we retain KITs through Equation (15).

$$\widetilde{s}_j^K = \{s_{j,i} \mid Q(s_{j,i}) \geq \gamma_j^Q\} \quad (15)$$

where $\widetilde{s}_j^K$ is the retained KITs in $\widetilde{s}_j$.

Then, we remove irrelevant supplementary knowledge that affects the performance of LLMs in NITs via Equation (16).

$$\widetilde{s}_j^N = \{Q(s_{j,i}) < \gamma_j^Q \text{ and } P(s_{j,i}) \leq \gamma_j^P\} \quad (16)$$

where $\widetilde{s}_j^N$ is the retained NITs in $\widetilde{s}_j$.

Therefore, we can get $\widetilde{s}_j$:

$$\widetilde{s}_j = \widetilde{s}_j^K \cup \widetilde{s}_j^N \quad (17)$$

The compressed prompt $\widetilde{s}$ is formed by sequentially concatenating each compressed segment.

$$\widetilde{s} = \widetilde{s}_1 \odot \widetilde{s}_2 \odot \ldots \odot \widetilde{s}_m \quad (18)$$

Methods	LongChat-7B-v1.5-32k 1st	5th	10th	15th	20th	LLaMA-3-8B-Instruct 1st	5th	10th	15th	20th	Length Tokens	$1/\tau$
2x constraint												
Retrieval-based Methods												
OpenAI	62.0	61.7	61.4	61.2	60.8	73.0	73.5	73.0	74.1	73.6	1,408	2.1x
SentenceBert	61.1	60.3	60.5	59.9	59.8	72.8	73.4	72.8	73.0	72.9	1,410	2.1x
BgeReranker	61.9	61.1	60.7	61.3	59.6	74.5	74.1	74.7	73.8	72.6	1,405	2.1x
BgeLLMembedder	62.5	61.8	62.2	61.6	60.8	74.0	73.7	74.2	74.0	74.0	1,407	2.1x
Compression-based Methods												
SelectiveContext	47.2	41.8	40.5	39.2	41.3	64.1	58.3	57.2	56.6	55.9	1,725	1.7x
LLMLingua	39.7	37.4	35.8	34.7	36.0	48.1	46.7	46.0	44.3	44.4	1,535	1.9x
LLMLingua2	54.8	46.4	44.0	42.5	44.3	70.8	62.7	62.1	61.5	61.8	1,475	2.0x
LongLLMLingua	62.3	61.7	62.4	62.4	61.8	77.0	76.0	74.8	75.2	74.9	1,444	2.1x
Perception Compressor	**64.7**	**64.1**	**63.5**	**63.2**	**64.6**	**79.5**	**78.3**	**78.6**	**79.1**	**79.1**	1,373	2.1x
4x constraint												
Retrieval-based Methods												
OpenAI	60.5	61.4	60.5	60.8	60.7	71.1	70.8	72.7	72.4	72.1	664	4.4x
SentenceBert	60.0	59.6	60.1	59.5	58.9	72.0	72.1	71.7	71.8	72.5	665	4.4x
BgeReranker	62.9	62.4	62.7	61.6	59.9	74.7	75.0	74.8	74.8	72.3	664	4.4x
BgeLLMembedder	62.5	57.7	61.6	61.2	61.2	73.5	72.9	73.5	73.1	74.3	664	4.4x
Compression-based Methods												
SelectiveContext	34.0	31.1	30.3	29.0	31.0	47.6	46.7	44.8	44.5	43.2	828	3.6x
LLMLingua	30.4	29.5	29.2	28.5	28.9	38.2	38.0	37.1	35.8	35.6	764	3.9x
LLMLingua2	42.9	35.7	33.9	31.3	33.5	59.3	53.6	52.5	50.1	48.9	741	4.0x
LongLLMLingua	63.4	63.4	62.9	62.4	63.0	76.5	75.0	74.8	74.8	75.0	736	4.0x
Perception Compressor	**66.3**	**66.0**	**66.0**	**66.1**	**66.2**	**79.6**	**79.5**	**80.7**	**80.2**	**80.0**	697	4.2x
Original Prompt	63.1	56.6	53.1	53.1	56.1	76.6	67.5	65.8	67.4	65.6	2,949	-
Zero-shot			31.0					46.7			10	294x

Table 2: Performance of different methods under different target compression constraints on NaturalQuestions. 1st, 5th, 10th, 15th, and 20th refer to the positions of the document containing ground truth among the all 20 documents.

5 Experiments

In this section, We seek to answer the following research questions: (1)How does the Perception Compressor address the two challenges in long context scenarios, *i.e.*, sensitivity to the position of key information, and excessively long input sequences? (2)How does Perception Compressor compare to baseline methods in long context benchmarks? (3)What is the impact of each component within the Perception Compressor on the overall method?

We also conduct some supplementary experiments, including parameter sensitivity analysis (see Appendix B), latency and memory usage (see Appendix C), evaluation on black-box large models (see Appendix D), the impact of compressor model size (see Appendix I), and case study (see Appendix G).

5.1 Datasets & Evaluation Metric

We evaluate our method on three long context benchmarks. To study the impact of the position of key information, we utilize NaturalQuestions (Liu et al., 2024). For the multi-hop question answering (QA) task, we test on MusicQue (Trivedi et al., 2022). Furthermore, to comprehensively assess the performance of compressed prompts across various long context scenarios, we use the English portion of LongBench (Bai et al., 2023). More details are provided in Appendix F.

5.2 Implementation Details

In this paper, We use LLaMA-2-7B to compress prompts. We validate the effectiveness of compressed prompts with LongChat-7B-v1.5-32k and LLaMA-3-8B-Instruct. We implement our method based on Pytorch 2.2.2 and HuggingFace Transformers. For hyperparameters, we set the coarse-grained control coefficient to 1.1, the open-book coefficient τ_o to 0.2, and τ_{ins} and τ_{q} for instruction and input question to 0.95 and 0.9, respectively. Slopes k_1 and k_2 are set to 0.4 and 0.1. More details are provided in the Appendix E.

5.3 Baseline Methods

We consider the following baseline methods.

- *Retrieval-based Methods*. Retrieval-based methods include several SOTA methods: BM25, Sentence Bert (Reimers and Gurevych, 2019), Jina (Günther et al., 2023), Gzip (Jiang et al., 2023b), OpenAI Embeding[4], Bge, BgeReranker and Bge LLMembedder (Xiao et al., 2023). In the recall experiment (Table 1), we compare our method with all these methods, but in the compression experiments (Table 2, 3 and 4), we select the top four methods with high recall rates for comparison,

[4]https://platform.openai.com/docs/guides/embeddings

Methods	LongBench							Length	
	SingleDoc	MultiDoc	Summ.	Code	Synth.	FewShot	Avg	Tokens	$1/\tau$
3,000 tokens constraint									
Retrieval-based Methods									
SentenceBert	24.0	23.0	26.0	37.7	32.9	64.1	34.6	3,001	3.4x
BgeReranker	24.4	25.1	25.9	38.4	34.7	64.8	35.6	3,001	3.4x
BgeLLMembedder	24.3	25.3	25.7	37.3	27.5	65.1	34.2	3,001	3.4x
Compression-based Methods									
SelectiveContext	21.1	22.7	24.8	44.8	11.9	49.2	29.1	3,065	3.4x
LLMLingua	22.3	16.2	25.0	49.0	12.6	65.7	32.1	3,004	3.4x
LLMLingua2	24.4	24.8	25.7	41.8	25.2	48.7	31.8	3,166	3.2x
LongLLMLingua	23.6	27.8	25.6	43.1	52.3	65.9	39.7	3,051	3.4x
Perception Compressor	**27.6**	**28.5**	**26.1**	**50.0**	**54.2**	**66.5**	**42.2**	2,840	3.6x
2,000 tokens constraint									
Retrieval-based Methods									
SentenceBert	25.6	25.5	26.0	39.5	30.0	64.0	35.1	2,244	4.6x
BgeReranker	24.7	25.4	25.4	39.6	32.9	63.2	35.2	2,247	4.6x
BgeLLMembedder	23.1	26.2	25.7	39.4	22.2	63.3	33.3	2,245	4.6x
Compression-based Methods									
SelectiveContext	19.8	22.2	23.1	42.8	8.2	47.2	27.2	2,273	4.5x
LLMLingua	20.4	14.4	24.2	47.0	9.2	65.1	30.8	2,251	4.6x
LLMLingua2	25.2	25.8	25.3	40.5	18.3	46.1	30.2	2,329	4.4x
LongLLMLingua	24.1	25.8	25.2	44.3	38.7	65.3	37.2	2,103	4.9x
Perception Compressor	**26.6**	**26.5**	**26.2**	**54.2**	**51.7**	**65.9**	**41.9**	1,896	5.4x
Original Prompt	28.6	21.5	16.0	55.6	37.0	65.6	37.4	10,276	-

Table 3: Performance of different methods under different tokens constraints on LongBench (Bai et al., 2023).

Methods	F1	Tokens	$1/\tau$
Retrieval-based Methods			
SentenceBert	26.8	1,368	1.9x
BgeReranker	25.9	1,371	1.9x
BgeLLMembedder	26.9	1,371	1.9x
Compression-based Methods			
SelectiveContext	15.8	1,553	1.7x
LLMLingua	15.2	1,343	1.9x
LLMLingua2	21.3	1,298	2.0x
LongLLMLingua	23.5	1,364	1.9x
Perception Compressor	**29.2**	1,287	2.0x
Original Prompt	25.3	2,571	-

Table 4: Performance of different methods on MuSiCQue (Trivedi et al., 2022) with 2x constraint using LLaMA-3-8B-Instruct.

i.e., SentenceBert, BgeLLMembedder, BgeReranker, and OpenAI Embedding. We discard sentences or tokens with low association with the input question until the compression constraint is met. Due to economic constraints, OpenAI Embedding is only compared in the experiments on NaturalQuestions.

- *Compression-based Methods*. Compression-based methods also include four SOTA methods explicitly compressing prompts: SelectiveContext (Li et al., 2023), LLMLingua (Jiang et al., 2023a), LLMLingua2[5] (Pan et al., 2024), and LongLLMLingua (Jiang et al., 2024). As for the LLM for compressing context, SelectiveContext uses the default GPT-2, while LLMLingua and LongLLMLingua use LLaMA-2-7B.

[5]Specifically, we compare with LLMLingua2-large in all experiments.

5.4 Results & Discussions

We analyze experiment results and illustrate our findings in the context of answering our research questions.

1. How does Perception Compressor address the two challenges in long context scenarios, *i.e.*, sensitivity to the position of key information, and excessively long input sequences?

In long context scenarios, LLMs achieve the highest performance when key information appears at the beginning of the prompt, but if the key information appears in the middle, LLM's performance drops significantly, *i.e.*, *lost in the middle* (Liu et al., 2024). To address this challenge, we introduce Perception retriever, which first calculates a perception perplexity r_k for each demonstration, representing the relevance of each demonstration to the input question, and then reorders the demonstrations from r_k in descending order, *i.e.*, the demonstration more relevant to the input question is placed at the front, thus addressing the *lost in the middle* challenge. Perception Compressor has the highest recall@1 in the recall experiment, which proves that it has superior re-ranking performance compared to LongLLMLingua.

As shown in Table 2, Table 3, and Table 4, Perception Compressor achieves state-of-the-art (SOTA) performance while drastically reducing the

input length. This effectively addresses the challenge of excessively long input sequences with too much redundant information.

2. How does Perception Compressor compare to baseline methods in long context benchmarks?

Table 1, Table 2, Table 3, and Table 4 present the results of our method compared to baseline methods on NaturalQuestions, LongBench, and MuSiQue. Here are some findings: (1) Perception Compressor outperforms all baseline methods by a large margin across all compression constraint ($1/\tau$) settings and tasks, using nearly the fewest input tokens. This highlights the robustness and superiority of the Perception Compressor. For example, at the 10th position under a 4x compression ratio, Perception Compressor improves by 5.9% compared to the second-best method, LongLLMLingua, using fewer input tokens. (2) Retrieval-based methods, constrained by relatively low recall, perform well across all datasets but fail to achieve the highest performance. (3) Our Perception retriever surpasses all comparison methods at recall@1-13, with larger gaps as the number of retrieved demonstrations decreases. Notably, at recall@1, it exceeds LongLLMLingua by 5.2%, showcasing the high upper bound of our method under high compression ratios. (4) Methods that do not consider the input question, such as LLMLingua, SelectiveContext, and LLMLingua2, perform poorly in long context scenarios, clearly inferior to retrieval-based methods. LLMLingua and SelectiveContext even perform worse than zero-shot settings on NaturalQuestions. (5) Even for multi-hop questions, the Perception Compressor remains effective. This indicates that when key information is dispersed, the Perception Compressor can still accurately sense the position of different key information and achieve the best performance.

3. How do the individual components of Perception Compressor influence its success?

To evaluate the contributions of different components in Perception Compressor, we conduct five variants of it for an ablation study (see Table 5). (1)*Our w/o Perception Retriever* refers to first reordering demonstrations based on their average perplexity from high to low, then retaining demonstrations with higher average perplexity, which may provide more supplemental knowledge to the LLM (Jiang et al., 2023a).(2)*Our w/o Dual-slope Ratio Allocator* disregards the open-book coefficient τ_o, slopes k_1 and k_2, directly computing τ_{dems} based on pre-defined τ, τ_{q}, and τ_{ins} using Equation (8). (3)*Our w/o Semi-guided Iterative Compression* only retains tokens with higher perplexity during the iterative compression process. (4)*Our w LongLLMLingua* r_k denotes using LongLLMLingua r_k instead of the Perception retriever r_k during the retrieval stage. (5)*Our w contrast perplexity ITPC* indicates replacing the semi-guided iterative compression with contrast perplexity ITPC, which is the token-level compression algorithm in LongLLMLingua.

As shown in Table 5, removing any component of the Perception Compressor leads to a significant drop in performance, which well validates the effectiveness of each component. The removal of the perception retriever leads to the most significant performance drop, which may owe to the loss of key information and the *lost in the middle* challenge. We also conduct a comprehensive comparison with LongLLMLingua, and the use of LongLLMLingua r_k and Contrast ITPC both result in inferior performance, which demonstrates that our method is superior to LongLLMLingua.

	1st	5th	10th	15th	20th	Tokens
Perception Compressor	**79.5**	**78.3**	**78.6**	**79.1**	**79.1**	1,373
- w/o Perception Retriever	45.3	45.9	45.6	46.4	46.1	1,385
- w/o Semi-guided Iterative Compression	78.3	77.3	77.5	78.0	78.2	1,405
- w/o Dual-slope Ratio Allocator	76.2	76.1	76.4	76.2	76.3	1,456
- w LongLLMLingua r_k	77.1	77.3	76.9	77.0	77.4	1,377
- w contrast perplexity ITPC	78.0	77.2	77.6	78.0	78.0	1,382

Table 5: Ablation study on NaturalQuestions under 2x constraint using LLaMA-3-8B-Instruct.

6 Conclusion

In this paper, we present the Perception Compressor, a training-free prompt compression framework. It consists of three components: a perception retriever, a dual-slope ratio allocator, and a semi-guided iterative compression. Our method compresses prompts via demonstrations reordering, compression ratios and open-book ratios allocation, preservation KITs, and removal of high perplexity NITs. We conduct extensive experiments on LongBench, NaturalQuestions, and MuSiQue. The experiment results demonstrate that Perception Compressor achieves SOTA performance across all tasks, effectively solving two challenges, *i.e.*, *lost in the middle*, and excessively long input sequences. Our method has the potential to enhance LLM performance while substantially compressing long context.

Limitations

There are also some limitations in our method. (1) As an explicit prompt compression method, Perception Compressor has a limited upper bound on compression ratio, and obviously cannot achieve the same level of context compression into one token as in Cheng et al. (2024). (2) If the demonstrations in the context are not well-defined, they need to be divided based on separators or semantics, as there is no fixed algorithm for dividing demonstrations.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (Grant No. 62276154), the Research Center for Computer Network (Shenzhen) Ministry of Education, the Natural Science Foundation of Guangdong Province (Grants No. 2023A1515012914 and 44030024103310080177O), the Basic Research Fund of Shenzhen City (Grants No. JCYJ20210324120012033, JCYJ20240813112009013, and GJHZ20240218113603006), and the Major Key Project of PCL (Grants No. PCL2022A05 and PCL2023A09).

References

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

Thomas Bayes. 1958. An essay towards solving a problem in the doctrine of chances. *Biometrika*, 45(3-4):296–315.

Hugh H Benson. 2011. Socratic method. *The Cambridge companion to socrates*, pages 179–200.

Amanda Bertsch, Uri Alon, Graham Neubig, and MatthewR. Gormley. 2023. Unlimiformer: Long-range transformers with unlimited length input.

Zhiwei Cao, Qian Cao, Yu Lu, Ningxin Peng, Luyang Huang, Shanbo Cheng, and Jinsong Su. 2024. Retaining key information under high compression ratios: Query-guided compressor for llms. *ArXiv*, abs/2406.02376.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. xrag: Extreme context compression for retrieval-augmented generation with one token. *ArXiv*, abs/2405.13792.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *ArXiv*, abs/2305.14788.

Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. 2024. Learning to compress prompt in natural language formats. *arXiv preprint arXiv:2402.18700*.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Ronald A Fisher. 1922. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 222(594-604):309–368.

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*.

Henry Gilbert, Michael Sandborn, Douglas C Schmidt, Jesse Spencer-Smith, and Jules White. 2023. Semantic compression with large language models. In *2023 Tenth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 1–8. IEEE.

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. 2023. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents. *Preprint*, arXiv:2310.19923.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*.

Yutong Hu, Quzhe Huang, Mingxu Tao, Chen Zhang, and Yansong Feng. 2024. Can perplexity reflect large language model's ability in long text understanding? *Preprint*, arXiv:2405.06105.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Towards unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2(3).

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023a. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Confer-*

ence on Empirical Methods in Natural Language Processing, pages 13358–13376, Singapore. Association for Computational Linguistics.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand. Association for Computational Linguistics.

Zhiying Jiang, Matthew Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, and Jimmy Lin. 2023b. "low-resource" text classification: A parameter-free classification method with compressors. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6810–6828, Toronto, Canada. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, page 453–466.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Filippo 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. *arXiv: Computation and Language,arXiv: Computation and Language*.

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*.

Zongqian Li, Yixuan Su, and Nigel Collier. 2024. 500xcompressor: Generalized prompt compression for large language models. *arXiv preprint arXiv:2408.03094*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Erik Nijkamp, Tian Xie, Hiroaki Hayashi, Bo Pang, Congying Xia, Chen Xing, Jesse Vig, Semih Yavuz, Philippe Laban, Ben Krause, et al. 2023. Xgen-7b technical report. *arXiv preprint arXiv:2309.03450*.

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. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 963–981, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

David Rau, Shuai Wang, Hervé Déjean, and Stéphane Clinchant. 2024. Context embeddings for efficient answer generation in rag. *arXiv preprint arXiv:2407.09252*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Claude E Shannon. 1951. Prediction and entropy of printed english. *Bell system technical journal*, 30(1):50–64.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2024. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36.

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. 2023. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*.

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.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2024. Compressing lengthy context with ultragist. *arXiv preprint arXiv:2405.16635*.

A Theoretical Derivation of Perception Perplexity r_k

The theoretical derivation of our method primarily relies on Bayes' theorem (Bayes, 1958).

A.1 Bayes's Theorem

Bayes' Theorem is a mathematical formula used to update the probability of a hypothesis based on new evidence. It is expressed as:

$$p(b|a) = \frac{p(a|b) \cdot p(b)}{p(a)} \tag{19}$$

Where $p(b|a)$ is the posterior probability of the hypothesis b given the evidence a; $p(a|b)$ is the likelihood of the evidence given that the hypothesis is true; $p(b)$ is the prior probability of the hypothesis before considering the evidence; $p(a)$ is the total probability of the evidence across all possible hypotheses.

A.2 Theoretical Derivation

$$r_{k,j} = \sum_{i=1}^{L_{ins}+L_{q_j}+L_r} g\left(\mathbf{x}_i^{\text{con}_j}\right) \log p\left(\mathbf{x}_i^{\text{con}_j} \mid \mathbf{x}^{\text{dems}_k}\right) \tag{20}$$

where $g(\cdot)$ represents the probability distribution of the ground truth, *i.e.*, $g\left(\mathbf{x}_i^{\text{con}_j}\right)$ is constant; $\mathbf{x}^{\text{con}_j}$ is the concatenation of $\mathbf{x}^{\text{ins}}$, $\mathbf{x}^{\text{q}_j}$ and $\mathbf{x}^{\text{r}}$; $\mathbf{x}_i^{\text{con}_j}$ denotes the i-th token in $\mathbf{x}^{\text{con}_j}$; $\mathbf{x}^{\text{dems}_k}$ and $\mathbf{x}^{\text{r}}$ refer the k-th demonstration and regularization constraint[6], respectively.

According to Bayes' Theorem, we can get:

$$p\left(\mathbf{x}^{\text{dems}_k} \mid \mathbf{x}_i^{\text{con}_j}\right) = p\left(\mathbf{x}^{\text{dems}_k}\right) \frac{p\left(\mathbf{x}_i^{\text{con}_j} \mid \mathbf{x}^{\text{dems}_k}\right)}{p\left(\mathbf{x}_i^{\text{con}_j}\right)} \tag{21}$$

where $p\left(\mathbf{x}^{\text{dems}_k}\right)$ and $p\left(\mathbf{x}_i^{\text{con}_j}\right)$ are constants.

We can derive from Equation (20) and Equation (21) that:

$$\{r_{k,j}\}_i \propto \log p\left(\mathbf{x}^{\text{dems}_k} \mid \mathbf{x}_i^{\text{con}_j}\right) \tag{22}$$

where $\{r_k\}_i$ is the impact of the i-th token in $\mathbf{x}^{\text{con}_j}$ on the perception perplexity.

$$r_k = \sum_{j=0}^{n} w_j \cdot r_{k,j} \tag{23}$$

According to Equation (23) and Equation (22), we can derive:

[6]For fair comparison, We use the same regularization constraint as in Jiang et al. (2024).

$$\{r_k\}_i = \sum_{j=0}^{n} w_j \{r_{k,j}\}_i$$
$$\propto \sum_{j=0}^{n} w_j \log p\left(\mathbf{x}^{\text{dems}_k} \mid \mathbf{x}_i^{\text{con}_j}\right)$$

So, we can get:

$$\{r_k\}_i \propto \log \prod_{j=0}^{n} p\left(\mathbf{x}^{\text{dems}_k} \mid \mathbf{x}_i^{\text{con}_j}\right)^{w_j} \tag{24}$$

B Parameter Sensitivity Analysis

τ_o, k_1, k_2 are the three main hyperparameters of the Perception Compressor. To explore the performance of the Perception Compressor under different combinations of τ_o, k_1, k_2, we conduct a parameter sensitivity analysis. Specifically, we study the impact of each parameter on performance individually, allowing the value of the parameter under study to range from 0.2 to 0.8 in increments of 0.2 (As mentioned in Appendix E, the initial parameter combination is $k_1 = 0.4, k_2 = 0.1, \tau_o = 0.2$).

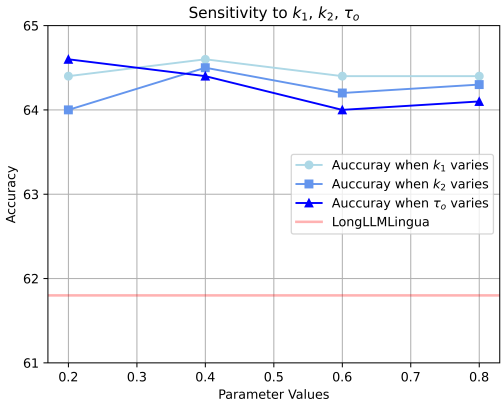


Figure 3: Parameter Sensitivity Analysis on the 20th position of NatureQuestions under 2x constraint. We use LongChat-7B-v1.5-32k as the response model.

The experimental results are shown in Figure 3, from which we can draw two findings: (1) The performance of the Perception Compressor is highly stable under different parameter combinations, with fluctuations in Accuracy of less than 1%, indicating that the Perception Compressor does not require exclusive tuning. (2) Perception Compressor significantly outperforms the previous state-of-the-art method, LongLLMLingua, under all parameter combinations, demonstrating its superiority.

C Latency and Memory Usage

We conduct experiments on Latency and Memory Usage on NatureQuestions using two NVIDIA A800 GPUs. The LLM used for testing Infer Latency is LLaMA-3-8B-Instruct, and other implementation details can be found in Sec. 5.2 and Appendix E.

From Table 6, we can draw three findings: (1)Under different compression constraints, the Compression Latency and Compression Memory Usage of Perception Compressor are slightly higher than those of LongLLMLingua. During the compression process, Perception Compressor used additional guiding questions in the retrieval phase, resulting in slightly higher Compression Latency and Compression Memory compared to LongLLMLingua, with Compression Latency increasing by approximately 1.2 seconds and Compression Memory Usage increasing by about 1 GB. Despite the minor cost in time and space, the performance improvement is significant, demonstrating the effectiveness of our method. (2)The Inference Latency of prompts compressed by LongLLMLingua and Perception Compressor is lower than the Latency of directly inputting the Original Prompt, but the reduction in Latency is minimal. This may be because the token length of the Original Prompt is not long to begin with, so the size of the large language model (LLM) itself is the main factor affecting the Inference Latency at this time. (3)The Infer Memory Usage of compressed prompts is significantly reduced. The Infer Memory Usage of compressed prompts is reduced from approximately 19 GB to about 17 GB.

D Evaluation on Black-box Large Language Models

We conduct further evaluation using the latest black-box large model GPT-4o-mini on NatureQuestions (see Table 1).

Our findings reveal that Perception Compressor consistently outperforms LongLLMLingua under various compression constraints, with the gap widening as the compression constraint increases. This observation strongly validates the superiority and effectiveness of Perception Compressor. This may be attributed to Perception Compressor's highest recall@1 (see Table 7), which enhances the likelihood of ground truth documents being ranked at the beginning of prompts and the probability of retaining key information under high compression

Methods	Memory Usage		Latency		Length	
	Compression Memory Usage	Infer Memory Usage	Compression Latency	Infer Latency	Tokens	$1/\tau$
2x constraint						
LongLLMLingua	17,154.9	17,627.0	1.8	2.3	1,357	2.2x
Perception Compressor	18,383.0	17,634.2	3.0	2.3	1,346	2.2x
4x constraint						
LongLLMLingua	15,545.0	16,967.3	1.0	2.2	697	4.3x
Perception Compressor	17,031.0	16,996.2	2.2	2.2	690	4.3x
Original Prompt	-	19,625.8	-	2.5	2,949	-

Table 6: The comparison of Latency (s) and Memory Usage (MB) between Perception Compressor and LongLLMLingua on NatureQuestions. Each value in the table is the average of experimental results from five positions (i.e., 1st, 5th, 10th, 15th, and 20th).

rates.

Methods	1st	5th	10th	15th	20th	Tokens
2x constraint						
LongLLMLingua	76.9	77.4	77.6	77.4	77.6	1,444
Perception Compressor	**78.2**	**77.6**	**78.6**	**78.2**	**78.5**	**1,373**
4x constraint						
LongLLMLingua	77.8	76.3	76.6	76.5	76.9	736
Perception Compressor	**78.9**	**79.2**	**79.1**	**78.5**	**79.4**	**697**

Table 7: Performance Comparison of LongLLMLingua and Perception Compressor on NatureQuestions under 2x constraint. The response model is GPT-4o-mini.

E Implementation Details

We use LLaMA-3-8B-Instruct tokenizer to count all the tokens. We generate three guiding questions for each input question and set the segment size to 200 tokens. For retrieval-based methods, we use the current strongest baseline model, and the relation between methods and versions is shown in Table 8. We conduct all experiments on 8 NVIDIA GeForce RTX 3090 GPUs.

Methods	Corresponding Version
OpenAI Embedding	text-embedding-3-large
SentenceBert	all-mpnet-base-v2
Bge	bge-large-en-v1.5
BgeReranker	bge-reranker-large
BgeLLMembedder	llm-embedder
Jina	jina-embeddings-v2-base-en
Gzip	-

Table 8: The relation between methods and versions. There is only one version of Gzip.

F Dataset Details

NaturalQuestions Multi-document QA NaturalQuestions Multi-document QA dataset is constructed by Liu et al. (2024) based on the NaturalQuestions dataset (Kwiatkowski et al., 2019), containing 2,655 questions. Each sample in the dataset includes a question and k related documents. These k related documents are obtained from historical queries issued to the Google search engine and human-annotated answers extracted from Wikipedia using the Contriever retrieval system (Izacard et al., 2021), with only one of them containing the correct answer document. In our experiments, we used the version with 20 documents, where the dataset includes five different true document position settings in the prompt: 1st, 5th, 10th, 15th, and 20th. Following Liu et al. (2024), we use Accuracy as the evaluation metric. The average prompt tokens length in this benchmark is 2,949.

LongBench LongBench (Bai et al., 2023) is a long context benchmark, and its English portion includes 16 tasks and 3,750 questions. These tasks can be divided into six categories, comprehensively covering various long context application scenarios, *i.e.*, multi-document question answering, single-document question answering, few-shot learning, code completion, synthetic tasks, and summarization. In our experiment, we use the English portion of this dataset and the evaluation script provided with it. The average prompt token length in this benchmark is 10,276.

MuSiQue MuSiQue (Trivedi et al., 2022) dataset is a multi-hop question-answering dataset consisting of 39,876 training samples, 4,834 validation samples, and 4,918 test samples. Among them, the validation set is composed of 2,411 answerable questions and 2,507 unanswerable questions. Completing this task requires large language model (LLM) to conduct multiple inferences based on

Input Question:
where did the titanic sink at what ocean?

Prompt:
Please provide three most helpful guiding questions to address the original question: where did the titanic sink at what ocean?

GPT-3.5-Turbo's Response:
1. Can you recall any historical events related to a famous ship sinking?
2. What do you know about the Titanic and its tragic fate?
3. Can you identify any significant bodies of water where maritime disasters have occurred?

Figure 4: Generate guiding questions for the input question "where did the titanic sink at what ocean?".

several documents, necessitating the capability for global information processing. In our experiments, we use the answerable questions from the validation set. Following Trivedi et al. (2022), we report standard F1 as the evaluation metric. The average prompt token length in this benchmark is 2,571.

G Cases Study

Large language models (LLMs) effectively comprehend the semantic information in the compressed prompts, even if it might be challenging for humans (Gilbert et al., 2023; Jiang et al., 2023a).

Observing the compressed prompts (see Figure 5 and Figure 6), we can find that the key information is placed at the beginning and remains intact. As shown in Figure 5, the Document [19] containing the ground truth, which should appear at the end of the prompt, is reordered to the beginning of the prompt, and the key information March 8, 2018 remains intact. As shown in Figure 6, the function `instructionNames`, which is very similar to the function `fieldNames`, is preserved under a high compression ratio and is reordered to the beginning. The key information `for(InlinedInstruction inst : insts) {` also remains intact. This indicates that Perception Compressor not only removes redundant information and significantly reduces the length of the input sequences but also solves the *lost in the middle* challenge raised in Liu et al. (2024), demonstrating its superiority and effectiveness. Therefore, the LLM generates correct responses to the seemingly corrupted compressed prompts.

H Generate Guiding Questions

In our experiment, we utilize GPT-3.5-Turbo to generate guiding questions for the input question (see Figure 4). Generating guiding questions related to the input question can help clarify thoughts and gradually break down the input question. Guiding questions often consider the input question from different angles, uncovering potential factors to help identify underlying assumptions and prerequisites. it is promising to propagate insights of solving guiding questions to inspire solving the input question.

I Impact of Model Size

The parameter size of LLMs affects their performance (Kaplan et al., 2020). To explore the impact of parameter scale on performance, we use GPT-2, a small LLM with only 137M parameters, to compress prompt instead of LLaMA-2-7B. As shown in Table 9, we can draw the following conclusions: (1) The prompt compression effect of GPT-2 is obviously not as good as that of LLaMA-2-7B, which may be due to the larger parameter size of the LLM having stronger capabilities. (2) Compared with other methods, the performance of Perception Compressor using GPT-2 is only lower than LongLLMLingua using LLaMA-2-7B and Original Prompt in the 1st position, while it achieves the highest performance in other settings. This demonstrates the superiority and effectiveness of our method.

Methods	1st	5th	10th	15th	20th	Tokens
Perception Compressor						
- w LLaMA-2-7B	**79.5**	**78.3**	**78.6**	**79.1**	**79.1**	1,373
- w GPT2	75.7	76.0	75.4	75.7	76.2	1,342
OpenAI	73.0	73.5	73.0	74.1	73.6	1,408
SentenceBert	72.8	73.4	72.8	73.0	72.9	1,410
BgeReranker	74.5	74.1	74.7	73.8	72.6	1,405
BgeLLMembedder	74.0	73.7	74.2	74.0	74.0	1,407
LongLLMLingua	77.0	76.0	74.8	75.2	74.9	1,444
Original Prompt	76.6	67.5	65.8	67.4	65.6	2,949

Table 9: Impact of model size on NaturalQuestions under 2x constraint using LLaMA-3-8B-Instruct.

Original Prompt:
Write a high-quality answer for the given question using only the provided search results (some of which might be irrelevant).
Document [0](Title:Jessica Jones (season 3))was ordered in April 2018, a month after the second season was released. Filming for the season began by the end of that June, with Ritter making her directorial debut during the season. The season is scheduled to be released in 2019. Star Krysten Ritter directs an episode for the season, marking her directorial debut. On April 12, 2018, a month after the release of the second season, Netflix ordered a third season of "Jessica Jones". With the season order came confirmation that the returning starring cast would include Krysten Ritter as Jessica Jones, Rachael Taylor as Patricia "Trish" Walker,
(omitted some tokens here)
Question: when is season 2 of jessica jones being released
Answer:

2866 tokens

Compressed Prompt:
Write high-quality answer for given question using only the provided search results (some of which might be irrelevant). 19:Jessica Jones (season 2))The season is scheduled to be released on March 8, 2018.
Document [1](Title:Jessica Jones (season 1))shortas. second season of "Jessica Jones" was ordered on January 1, 2016. <includeonlyinclude> October 201, Marvel and announced that Marvel Television and ABC Studios wouldflix with live action series aroundaredevil, Jessica Jones, Iron Fist, and Luke Cage, leading up to a miniseries based on the Defenders.issa Rosenberg was brought on to showrun theica Jones series, beured as a "-over" from original she had developed in 2010 for ABC. In December 2014, the title was revealed to be "Marvel's A.K.A. Jessica Jones", [1](Title:Jessica Jones (season 1)) 22, 2017, in Region 2 and Region on December , 2016, and in Region 4 on December 7, 2016. Asflix does notalberership numbers for of their original series,phony Group data the on sample size of 1,000 theirones viewinging's. to Symphony, December 2015, of "Jessica Jones"aged 4. million viewers 35-day viewing. The was by Wurtzel, Nal research [Title:essica Jones (season))Jica Jones (2 second season of the American web television seriesJessica Jones on the Marvel Comics of the same name, follows she on case surrounding encounter with Kilgrave in the Marvel Cinematic Universe (MCU), sharing continuity with the films other television series of the franchise. season by Television association with ABC Studios andall Productions, with Melissa Rosenberg as showrunner. Krysten Ritter as,ael Taylor,rie-Anne Moss, Eka Darville [9](Title:Jessica Jones (series))before? And answer."essica Jones available streaming serviceflix inories is available inra HD4K dynamic range (HDR). The first season inDR its initial release post-productionuxe The season released opposed toizedrage binge-watching for Netflix original series. Disney Consumer Products a line ofater to more adult audience the'sgier tone. Paul Gitter, [](Title:Jessica Jones (season)) in April 2018, a after the season was released. Filming the began end of directorial debut the. season scheduled to be released in 2019 Starrysten Ritter episode,ingial debut 2, 2018, month of season, Netflix third seasonJessica Jonesation include Krysten Ritterica Jones,ael Taylor asriciaTrish" Walker, [3](Title:Jessica Jones ( ))season,andostoo say season201 Netflixs a other "aredevils 2015 However, Netflix confirmed season 2015 announ November 20. The high range its initial release-productionuxe In 201,intic is based on, [](Title:Jessica Jones (season 1))"etherness mostatable trauma the amount of to othering A trailer released on November , 2015ene Leeds ofThe Street Journal" thatilered empowerment to theating' itsere . Region1u-ray Region
Question: when is season 2 of jessica jones being released
Answer:

702 tokens

LLaMA-3-8B-Instruct's Response:
The second season of "Jessica Jones" is scheduled to be released on March 8, 2018.

LongChat-7B-v1.5-32k Response:
The season is scheduled to be released on March 8, 2018.

Ground Truth:
March 8, 2018

Figure 5: Case study on NaturalQuestions (20 documents) (Liu et al., 2024) under 4x constraint.

Original Prompt:

```
Please complete the code given below.

/*******************************************************************************
 * Copyright (c) 2009 - 2015 IBM Corporation.
 * All rights reserved. This program and the accompanying materials
 * are made available under the terms of the Eclipse Public License v1.0
 * which accompanies this distribution, and is available at
 * http://www.eclipse.org/legal/epl-v10.html
 *
 * Contributors:
 *     IBM Corporation - initial API and implementation
```

(omitted some tokens here)

```
final Map<IField,String> field2Name = new LinkedHashMap<IField, String>();

Next line of code:
```

4236 tokens

Compressed Prompt:

```
Please complete the code given below.

public static Map<Instruction, String> instructionNames(Set<InlinedInstruction> insts) {
final Map<Node,String>Names = nodeNames(s.levantMethods(insts));
final Map<String, List<InlinedInstruction>> name2Inst = new LinkedHashMap<String, List<InlinedInstruction>>();
final Map<InlinedInstruction, String> inst2Name = new LinkedHashMap<InlinedInstruction, String>();

for(InlinedInstruction inst : insts) {
final String mget(inst.cNode());
 Stringix
 = inst.Index();
if (idx._VALUE) {
in";
 else if (idx.MAX_VALUE) {
infix = "end";
} else {
 String =.Class().getSimpleNamereplaceAll(" "replaceAll("Instruction", "");
fix = cname+idx;
final name = m + + infix ""; m; ListlinedInstruction>2Inst.get(name);  (namednull) {
named = new<InlinedInstruction3 2Inst.put(name,
}
for(Entry, List<linedInstruction>> entry2Inst.entrySet()) {
finallinedInstruction> =.getValue();  (namedsize()1) {
2Name.putnamed.(.getKey()); linedInstruction :) {
final
assert.empty Site it.callStack().(); b.append.itr.next()));
itrNext {
_" +.r.

public static Map<IField, String> fieldNames(Set<IField> fields) {
final Map<String, List<IField>> name2Field = new LinkedHashMap<String, List<IField>>();
final Map<IField,String> field2Name = new LinkedHashMap<IField, String>();

Next line of code:
```

389 tokens

LLaMA-3-8B-Instruct's Response:

```
for(IField field : fields) {
```

Ground Truth:

```
for(IField field : fields) {
```

Figure 6: Case study on lcc task in LongBench (Bai et al., 2023) under 500 tokens constraint.