ATACompressor: Adaptive Task-Aware Compression for Efficient Long-Context Processing in LLM

Anonymous ACL submission

Abstract

Long-context inputs in large language models (LLMs) often suffer from the "lost in the middle" problem, where critical information becomes diluted or ignored due to excessive length. Context compression offers a promising solution, however, current compression methods still have notable limitations: hard prompt methods often suffer from low compression ratios, while soft prompt methods tend to lose critical task-relevant information and lack adaptability. We propose ATACompressor, an adaptive, task-aware context compressor that combines the strengths of both paradigms. ATACompressor (1) efficiently compresses context into compact soft prompts, (2) selectively preserves task-relevant information through a trained encoder, and (3) dynamically adjusts compression rates via an adaptive controller. Experiments on QA benchmarks demonstrate that ATACompressor achieves state-of-the-art performance while maintaining high efficiency.

1 Introduction

Large language models (LLMs) demonstrate remarkable performance across diverse tasks, such as natural language understanding, text generation, and question answering (Chang et al., 2024; Naveed et al., 2023; Min et al., 2023). However, their static nature poses significant challenges. For example, they cannot independently update or adapt to new information. To bridge this gap, LLMs need external context to inject dynamic, domain-specific knowledge (Parthasarathy et al., 2024; Wang et al., 2023). This dependency highlights the critical importance of contextual information. Without it, large models could be outdated or misaligned with real-world data, compromising both their accuracy and practical utility.

Techniques like retrieval-augmented generation (RAG) address this challenge by retrieving relevant information from external sources, enabling the model to access up-to-date, task-specific data (Huang and Huang, 2024; Fan et al., 2024). Despite the benefits of providing ample context, naive RAG that appends raw document tokens to the model input could create excessively long context that overwhelms LLMs (Cuconasu et al., 2024), making it difficult for them to identify critical information, especially information in the middle of the context — a phenomenon commonly referred to as the "lost in the middle" problem (Hsieh et al., 2024; Liu et al., 2024a).

One way to address this challenge is by reducing the input length. A widely adopted approach is compressing the lengthy context into a more concise form, which eases the “lost in the middle” effect and lowers inference cost and latency (Li et al., 2024a). Existing long-context compression techniques can be broadly categorized into hard-prompt and soft-prompt methods. Hard prompt methods, such as Selective-Context (Li et al., 2023) and LongLLMLingua (Jiang et al., 2023), reduce context size by identifying and removing irrelevant or low-value content. While these methods effectively preserve task-relevant information, they typically result in lower compression ratios. In contrast, soft prompt methods, such as AutoCompressor (Chevalier et al., 2023), ICAE (Ge et al., 2023), and 500Compressor (Li et al., 2024b), compress text into a sequence of special tokens, achieving higher compression and greater information density by representing hundreds of tokens with just a few. However, these methods often suffer from the loss of task-relevant content due to the absence of task-specific information (e.g., questions) during the compression process. Additionally, their compression tokens number are fixed and cannot be dynamically adjusted based on the requirements of the task. These shortcomings prevent effective compression based on task-specific needs.

To address these challenges, we propose the **A**daptive **T**ask-**A**ware **Compressor** (ATACompres-

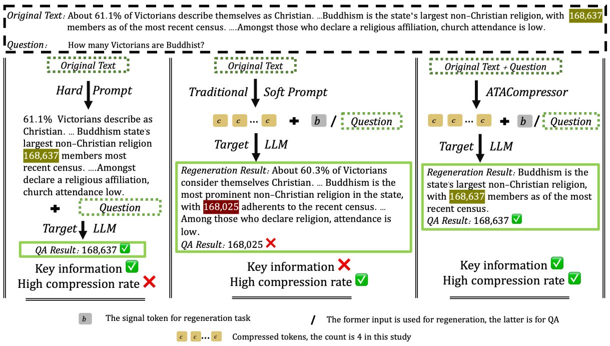

Original Text: About 61.1% of Victorians describe themselves as Christian. ...Buddhism is the state's largest non-Christian religion, with 168,637 members as of the most recent census. ....Amongst those who declare a religious affiliation, church attendance is low.
Question: How many Victorians are Buddhist?
Original Text
Hard Prompt
61.1% Victorians describe as Christian. ... Buddhism state's largest non-Christian religion 168,637 members most recent census. ....Amongst declare a religious affiliation, church attendance low.
Question
Target LLM
QA Result: 168,637 ✅
Key information ✅
High compression rate ❌
Original Text
Traditional Soft Prompt
c c ... c
b / Question
Target LLM
Regeneration Result: About 60.3% of Victorians consider themselves Christian. ... Buddhism is the most prominent non-Christian religion in the state, with 168,025 adherents to the recent census. ... Among those who declare religion, attendance is low.
QA Result: 168,025 ❌
Key information ❌
High compression rate ✅
Original Text + Question
ATACompressor
c c ... c
b / Question
Target LLM
Regeneration Result: Buddhism is the state's largest non-Christian religion, with 168,637 members as of the most recent census.
QA Result: 168,637 ✅
Key information ✅
High compression rate ✅
b The signal token for regeneration task
/ The former input is used for regeneration, the latter is for QA
c c ... c Compressed tokens, the count is 4 in this study

Figure 1: Comparative schematic of three approaches, using selective compressor to represent the hard prompt and 500Compressor to represent the traditional soft prompt.

sor), which offers three key advantages: (1) **Efficient Context Compression**: ATACompressor leverages soft prompt techniques to condense long contexts into compact token representations, preserving essential information and improving downstream task efficiency. (2) **Effective Key Information Preservation**: ATACompressor trains a selective encoder to compress only task-relevant content, filtering out irrelevant information while maximizing the retention of critical information. This task-aware compression strategy enhances downstream performance by focusing on the most important content. (3) **Adaptive Resource Allocation**: ATACompressor employs an adaptive allocation controller that infers the length of relevant content from internal states and dynamically adjusts the compression rate accordingly. It allocates fewer tokens to shorter relevant spans and more to longer ones, ensuring adequate preservation of essential information while optimizing resource utilization across diverse tasks. Figure 1 illustrates the characteristics of ATACompressor.

Our experiments on three public QA benchmarks show that ATACompressor consistently achieves state-of-the-art performance while maintaining high efficiency. Additionally, we conduct a series of ablation studies and analysis experiments to further investigate and understand the underlying effectiveness of ATACompressor.

2 Related Work

2.1 Retrieval-augmented Generation

Retrieval-Augmented Generation (RAG) enhances large language models by integrating external retrieval, improving content accuracy and factuality (Gao et al., 2023; Zhao et al., 2024a; Huang and Huang, 2024; Wang et al., 2023). It typically combines a retrieval module with a language model to generate responses based on retrieved data (Liu et al., 2024b; Gao et al., 2023; Hu and Lu, 2024). However, RAG struggles with long contexts due to issues like the "lost in the middle" effect (Cuconasu et al., 2024; Liu et al., 2024a; Hsieh et al., 2024), where critical mid-sequence information is missed. Processing long texts also increases computational cost and latency, limiting real-time or resource-constrained use (Zhao et al., 2024a; Agrawal et al., 2024). Addressing these challenges is essential for practical long-context applications.

2.2 Context Compression

A common approach to handling long contexts is extending the LLM's context window, typically via larger pretraining windows (Nijkamp et al., 2023), positional embedding interpolation (Peng et al., 2023; Zhu et al., 2023; Ding et al., 2024), or attention refinements (Chen et al., 2023). Though effective, these methods often entail significant architectural modifications.

Unlike context extension, context compression shortens inputs without modifying LLM architecture, enabling efficient long-context handling. It consists of two types: hard and soft prompt methods. Hard methods like Selective-Context (Li et al., 2023) and LongLLMLingua (Jiang et al., 2023) remove irrelevant tokens using external models or perplexity-based scoring but yield low compression ratios due to token retention. Soft methods, such as AutoCompressor (Chevalier et al., 2023), ICAE (Ge et al., 2023), and 500Compressor (Li et al., 2024b), compress contexts into dense vectors via fine-tuning or autoencoders, achieving higher ratios but often ignoring task relevance and lacking dynamic adaptability. Recent query-guided soft prompt methods like QGC (Cao et al., 2024), xRAG (Cheng et al., 2024), FlexRag (Liu et al., 2024b), and COCOM (Rau et al., 2024) improve task awareness but depend heavily on external retrievers and suffer from complex architectures, resulting in longer inference times and reliance on retriever quality.

Our ATACompressor algorithm, built on soft-prompt techniques, incorporates task information during compression and leverages the compressor's intrinsic ability to selectively extract, retain and compress the relevant portions of the context. It also dynamically adjusts the compression rate based on the task requirements. These features make it well-suited for RAG and other downstream tasks while delivering superior performance and efficiency.

2.3 Probe for LLMs

Probing techniques are used to interpret and enhance LLM behavior by attaching lightweight models to analyze internal representations (Dong et al., 2023; Ju et al., 2024; Ibanez-Lissen et al., 2024; Zhao et al., 2024b; Wang et al., 2024b). Prior work has used probes to uncover biases (Dong et al., 2023), analyze contextual encoding across layers (Ju et al., 2024), and study cross-lingual alignment (Wang et al., 2024a). In our work, we probe encoder hidden states to estimate relevant context length, enabling adaptive resource allocation.

3 Method

3.1 Problem Formulation

LLMs often take a task prompt (Q) and a context (C) as input to generate a target answer (A). However, the typically large size of C leads to challenges such as the "lost in the middle" problem, increased inference costs, longer latencies and potential performance degradation. A widely adopted way to address this challenge is context compression, the objective of ATACompressor can be formulated as:

$$\begin{aligned} &\min_{\varphi(Q,C)} d\big[\text{LLM}\big(A \mid Q, C\big), \text{LLM}\big(\tilde{A} \mid \varphi(Q,C), Q\big)\big] \\ &\text{s.t.} \quad |\varphi(Q,C)| = k \end{aligned} \tag{1}$$

Here, $\tilde{A}$ represents the output of the LLM with the compressed tokens $\varphi(Q,C)$, k represents the number of compressed tokens, and $d(\cdot,\cdot)$ is a distance function, such as KL divergence, that measures the difference between two distributions.

3.2 Architecture

As shown in Figure 2, **ATACompressor** comprises a selective encoder (φ), an adaptive allocation controller (AAC), and a target LLM (decoder). The process begins by segmenting the context C into chunks $C_1, C_2, \ldots, C_n$ using a predefined strategy. And then, the selective encoder (φ) processes the concatenated chunked context $C_{\text{ckd}} = \{C_1, C_2, \ldots, C_n\}$ along with the query Q [1], selectively compressing relevant information into a compressed token sequence, whose length k is determined by the AAC. The key-value (KV) representations of these compressed tokens are subsequently passed to the target LLM for downstream task. It is important to note that the selective encoder (φ) processes the input text and compresses the relevant parts into tokens continuously, due to the autoregressive nature of LLM. Figure 2 is split into two steps to illustrate this process more intuitively. As soon as the selective encoder (φ) processes the last token of the input text, it continues generating the first compressed token without interruption. At this point, the AAC also begins processing in parallel. Since the AAC is very efficiently due to its relatively lightweight structure, it can predict the total number of compressed tokens before the first compressed token is generated.

Compared to prior soft compression methods, ATACompressor introduces two key innovations: task-aware compression and dynamic token allocation, ensuring efficient resource utilization while preserving essential information. These are realized through two core components: the selective encoder (φ) and the adaptive allocation controller (AAC), detailed in the following sections.

3.3 Selective Encoder

Traditional soft embedding methods compress the entire context C using an encoder, but often fail to effectively identify and retain query-relevant information, potentially leading to the loss of crucial content. To address this, we train a selective encoder (φ) designed to enhance the encoder's ability to sense and extract relevant information. The selective encoder consists of a frozen LLM Θ_{LLM} with trainable LoRA parameters Θ_{LoRA}, and selectively compresses only the portions of the context needed to answer the query Q into a compact set of tokens. This improves compression by preserving relevant content and discarding irrelevant information, thereby enhancing downstream task performance.

Training the selective encoder φ is challenging because the optimization objective adopted in standard soft-prompt compressors (Ge et al., 2023; Li et al., 2024b) no longer applies. During pretraining, conventional compressors reconstruct the entire input, so the supervision signal—the input itself—is complete and internally consistent. In ATACompressor, however, φ preserves and compresses only the context fragments needed to answer Q; thus the supervision signal is the relevant parts of the whole context. This shift introduces granularity ambiguity. In real-world datasets, annotations of relevant

[1] In the following, we refer to all text inputs to the selective encoder as "input text".

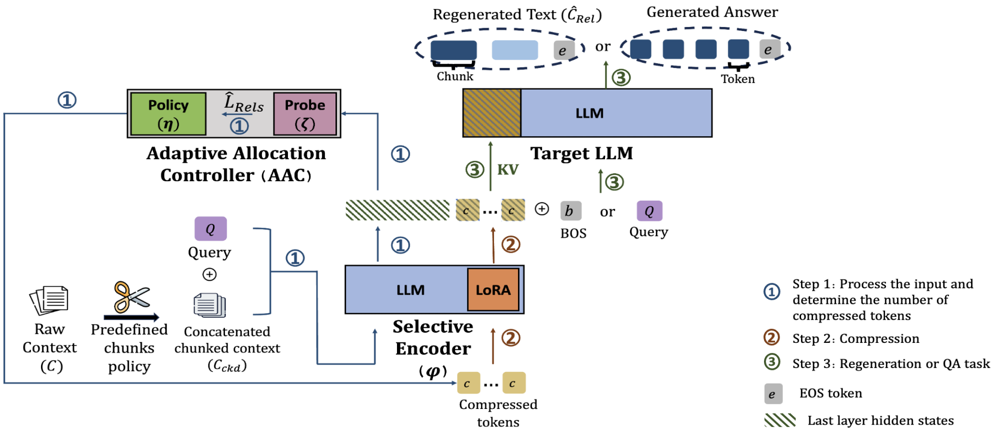

Regenerated Text ($\hat{C}_{Rel}$)
Generated Answer
e
or
e
Chunk
Token
③
LLM
Target LLM
③
KV
③
b
or
Q
BOS
Query
①
Policy (η)
$\hat{L}_{Rels}$
①
Probe (ζ)
Adaptive Allocation Controller (AAC)
①
①
Q
Query
⊕
①
LLM
LoRA
Selective Encoder (φ)
②
②
Raw Context (C)
Predefined chunks policy
Concatenated chunked context (C_{ckd})
c
c
Compressed tokens
① Step 1: Process the input and determine the number of compressed tokens
② Step 2: Compression
③ Step 3: Regeneration or QA task
e EOS token
Last layer hidden states

Figure 2: The inference workflow of ATACompressor. The illustration of the training workflow is provided in §A

context can vary in granularity (e.g., at the passage or document level). Without proper preprocessing, the inconsistent granularity of the annotated relevant context during training can confuse φ about how to process the context. For example, if some gold-truths indicate relevance at the document level while others are at the sentence level, φ will struggle with whether to group information coarsely or finely. Conversely, enforcing a single fixed granularity during training would limit its adaptability to tasks with different granularity requirements. Consequently, we must devise a mechanism that explicitly guides φ to perceive and compress relevant information at the desired granularity.

To mitigate this, we deterministically chunk the context C into uniform units $C_1, \dots, C_n$ that match the granularity of its gold labels [2], concatenate them into C_{ckd}, and feed $\langle Q, C_{\text{ckd}}\rangle$ to the selective encoder φ in one pass. φ then isolates the query-relevant subset $\hat{C}_{\text{Rel}} \subseteq C_{\text{ckd}}$ and compresses it into k tokens $c_1, \dots, c_k$, with k determined by the adaptive allocation controller (AAC). The strategy lets ATACompressor handle mixed or user-specified granularities. The formal process is described as follows [3]:

$$\varphi(Q, C) = \varphi(\langle Q, C_{\text{ckd}}\rangle) = \varphi(\hat{C}_{\text{Rel}}) = \underbrace{(c_1, \dots, c_k)}_{k \text{ determined by AAC}}, \tag{2}$$

$$\hat{C}_{\text{Rel}} = \underbrace{\{C_{t_1}, \dots, C_{t_m}\}}_{C_{t_i} \text{ relevant to } Q} \subseteq \{C_1, \dots, C_n\}, \tag{3}$$

3.4 Adaptive Allocation Controller (AAC)

The adaptive allocation controller (AAC) is composed of a probe (ζ) and a policy function (η). The probe (ζ) captures the selective encoder's hidden states to estimate the length of relevant content. This estimation directs the policy function (η) to dynamically adjust the compression rate, preserving relevant information and optimizing computational resource usage. The reason for selecting the length of $\hat{C}_{\text{Rel}}$ as a key signal for adjusting the number of compressed tokens (k) is as follows: The length of $\hat{C}_{\text{Rel}}$ represents length of the text needed to complete the task. Prior studies (Ge et al., 2023; Li et al., 2024b; Cao et al., 2024; Rau et al., 2024) have demonstrated that the performance of a soft-prompt compressor is primarily influenced by the ratio between the text length and k. With a fixed k, the performance of the compressor declines rapidly as the text length increases. By estimating the length of $\hat{C}_{\text{Rel}}$, k can be dynamically adjusted to task needs—allocating fewer tokens for shorter and more for longer relevant spans. This task-aware strategy ensures a more efficient and effective compression.

The probe (ζ) is a lightweight neural network comprising an MLP and attention layers. It analyzes the hidden states from the selective encoder (φ) to estimate the length of $\hat{C}_{\text{Rel}}$, guiding dynamic compression. Specifically, after φ processes the input, ζ takes its final-layer hidden states at the last token and outputs the estimated length $\hat{L}_{\text{Rel}}$, which is then used by the policy function (η) to determine the number of compressed tokens (k). The process

[2] At inference time the chunking granularity is determined by user or task requirements.

[3] Some examples of the preprocessed inputs are provided in §B

is formalized as:

$$\hat{L}_{\text{Rel}} = \zeta(\mathbf{H}_{\varphi}), k = \eta(\hat{L}_{\text{Rel}}) \tag{4}$$

Here, $\zeta(\cdot)$ denotes the probe's operation on the encoder's final hidden states $\mathbf{H}_{\varphi}$ to estimate $\hat{L}_{\text{Rel}}$. The policy function $\eta(\cdot)$ then determines the number of compressed tokens k based on $\hat{L}_{\text{Rel}}$. In our approach, k is set by dividing $\hat{L}_{\text{Rel}}$ by a policy ratio r, capped by a maximum k_{max}:

$$k = \eta(\hat{L}_{\text{Rel}}) = \min\left(\frac{\hat{L}_{\text{Rel}}}{r}, k_{\text{max}}\right) \tag{5}$$

Notably, the selective encoder (φ) and AAC operate independently, ensuring that the probe (ζ) structure or policy function (η) does not affect the encoder's ability. This independence allows flexibility in designing the AAC, particularly the policy function (η), to suit task-specific needs. For instance, the policy ratio r in our function can be adjusted without retraining, unlike traditional soft prompt methods that require retraining to change the compressed tokens' number (Ge et al., 2023; Li et al., 2024b).

3.5 Workflow

ATACompressor operates in three stages: *pretraining*, *finetuning*, and *inference*. For detailed information, please refer to the §A.

Pretraining. We jointly train the *selective encoder* φ and the *probe* ζ. Given a chunked context C_{ckd} and query Q, φ produces compressed key–value pairs $\mathbf{KV}$ that enable the frozen LLM to reconstruct the task-relevant subsequence C_{Rel}, while ζ predicts its token length $\hat{L}_{\text{Rel}}$. Their objectives combine into a single loss.

$$\mathcal{L}_{\text{pretrain}} = \underbrace{-\sum_{j=1}^{n} \log P\big(w_j \mid \mathbf{KV}, \text{[BOS]}, w_{1:j-1}\big)}_{\mathcal{L}_{\varphi}\text{ (cross–entropy)}} + \lambda\,\mathcal{L}_{\zeta}, \tag{1}$$

where w_j is the j-th token of C_{Rel} and $\mathcal{L}_{\zeta}$ is a Huber loss (Gokcesu and Gokcesu, 2021).

Finetuning. The selective encoder (φ) is further trained for downstream tasks. The target LLM generates task-specific outputs based on the $\mathbf{KV}$ of the compressed tokens. The loss function is defined as:

$$\mathcal{L}_F = -\sum_{j=1}^{n} \log P\big(a_j \mid \mathbf{KV}, q_{1:m}, a_{1:j-1}\big), \tag{2}$$

where a_j is the j-th gold answer token and $q_{1:m}$ is the query sequence.

Inference. During inference, all parameters are frozen. As described in §3.2, the inference process concludes with the compressed token KV pairs being passed to the target LLM to generate outputs in two modes: regenerating C_{Rel} (triggered by [BOS]) and answering the query.

For regeneration:

$$\hat{w}_i = \arg\max_{\hat{w}_i} P(\hat{w}_i \mid \mathbf{KV}, \text{[BOS]}, \hat{w}_{1:i-1}; \Theta_{\text{LLM}}) \tag{6}$$

For question answering:

$$\hat{a}_j = \arg\max_{\hat{a}_j} P(\hat{a}_j \mid \mathbf{KV}, q_{1:m}, \hat{a}_{1:j-1}; \Theta_{\text{LLM}}) \tag{7}$$

4 Experiments

4.1 Settings

4.1.1 Datasets

The experiments are based on the three datasets [4]:

- **HotpotQA** (Yang et al., 2018): Multi-hop QA dataset that demands combining information from multiple documents. We use it to evaluate compressors at the **document level**, where answers are synthesized from relevant documents.
- **MSMARCO** (Nguyen et al., 2016): A high-quality question answering dataset curated by Microsoft. We use it to evaluate compressors at the **passage level**, where answers are synthesized from relevant passages.
- **SQUAD** (Rajpurkar et al., 2018): A dataset where each question is paired with a passage, and the answer is typically a span of text found within that passage. We use it to evaluate compressors at the **sentence level**, where answers are synthesized from relevant sentences.

4.1.2 Baselines

We use three types of baselines.

1. No Compression

- **Closed-Book** The LLM directly answers questions without access to any external context.
- **Original-Context** The LLM answers questions with access to the full external context, using the original uncompressed context without any modifications.

2. Hard Prompt Compression

[4] Details of the dataset and examples of the preprocessed inputs are provided in §B.

Table 1: The experimental results on three benchmark datasets include the following metrics: EM (Exact Match), F1 (F1 score), CR (Compression Ratio), and TP (Throughput in *examples / second*). Our ATACompressor algorithm shows significant improvements in all metrics across all datasets compared to QGC and 500Compressor , with p < 0.001.

Methods	**HotpotQA**				**MSMARCO**				**SQUAD**			
	F1	EM	CR	TP	F1	EM	CR	TP	F1	EM	CR	TP
Qwen-2-7B												
Closed-book	30.61	10.35	–	**5.64**	15.48	0.99	–	**2.83**	38.84	6.33	–	**5.29**
Original-Context	59.88	39.73	1.0x	1.24	40.79	4.25	1.0x	0.41	68.52	48.75	1.0x	2.04
Selective-Context	53.76	37.10	3.41x	1.24	32.43	2.58	3.86x	0.66	59.67	40.49	4.64x	1.65
LongLLMLingua	64.63	40.08	4.37x	1.28	43.52	4.91	5.64x	0.75	64.89	48.10	5.08x	2.10
ICAE	65.39	39.71	22.92x	3.61	46.21	5.17	14.32x	1.26	61.26	45.33	21.19x	2.96
500Compressor	67.46	42.18	22.92x	3.51	47.23	5.33	14.32x	1.24	64.68	47.95	21.19x	2.95
QGC	72.41	51.54	13.75x	1.83	49.85	6.15	16.44x	0.79	66.79	49.27	16.92x	1.49
ATACompressor	**80.23**	**65.49**	**23.81x**	3.63	**53.30**	**8.15**	**25.32x**	1.35	**70.52**	**52.10**	**27.39x**	3.07
LLaMA-2-7B												
Closed-book	22.84	4.82	–	**6.37**	10.94	0.70	–	**3.47**	37.93	5.42	–	**5.67**
Original-Context	53.71	36.20	1.0x	1.21	38.72	4.09	1.0x	0.44	68.89	50.38	1.0x	2.12
Selective-Context	51.68	36.09	3.30x	1.22	30.58	2.36	3.63x	0.61	57.71	39.37	4.55x	1.36
LongLLMLingua	62.82	37.27	3.95x	1.32	42.98	3.44	4.58x	0.86	65.40	48.26	4.73x	1.72
AutoCompressor	59.66	32.05	11.96x	2.96	33.96	2.45	13.70x	1.26	60.52	41.49	14.21x	2.61
ICAE	62.10	37.72	22.92x	3.77	38.51	3.32	14.32x	1.27	64.28	46.75	21.19x	3.02
500Compressor	64.28	39.65	22.92x	3.70	40.30	3.40	14.32x	1.23	69.61	50.60	21.19x	3.13
QGC	68.21	45.15	14.32x	1.96	44.22	5.22	15.51x	0.81	68.43	50.45	15.91x	1.77
ATACompressor	**78.44**	**62.65**	**24.15x**	3.86	**50.06**	**8.00**	**27.36x**	1.29	**71.67**	**53.00**	**27.18x**	3.14

- **Selective-Context** (Li et al., 2023): It leverages self-information computed by an external language model to remove redundant words.

- **LongLLMLingua** (Jiang et al., 2023): It uses a language model to assess document importance via question-aware perplexity and applies a coarse-to-fine strategy to remove irrelevant tokens.

3. *Soft Prompt Compression*

- **AutoCompressor** (Chevalier et al., 2023): It fine-tunes an LLM to iteratively compress long contexts into summary vectors. We use the released AutoCompressor-Llama-2-7B-6K model[5] for experiments.

- **ICAE** (Ge et al., 2023): It adopts an autoencoder architecture to compress long contexts into compact memory slots.

- **500Compressor** (Li et al., 2024b): Similar to ICAE, the key difference is that it uses the KV representations of the compressed tokens instead of the embeddings.

- **QGC** (Cao et al., 2024): It compresses query-guided document representations into n-grams based on word importance to the query.

4.1.3 Main Evaluation Metrics

Following prior work (Cao et al., 2024; Li et al., 2024b), we evaluate downstream QA tasks using F1 score and Exact Match (EM). We also compute the compression ratio (CR), the ratio of original to compressed context length, and report inference throughput (TP) on two A100-40G GPUs, including compression and answer generation. In addition, Rouge-L-F is used in §5 to evaluate the performance of the regeneration task.

4.1.4 Implementation Details

We utilized a 280k dataset (180k MSMARCO and 100k HotpotQA) from training sets for pretraining and finetuning. For evaluation, 5k examples were randomly sampled from the test sets of MSMARCO, HotpotQA, and SQUAD. All reported results are averages over 5 random samplings unless stated otherwise. For ATACompressor, following §3.3, the datasets was segmented into `<PA></PA>` chunks: each document (HotpotQA), passage (MSMARCO), or sentence (SQUAD) was treated as a chunk. Other models used the same training data. To facilitate comparison with baselines, the maximum input length was set to 600 tokens (as many baselines conduct their main experiments using input lengths around 500 tokens and resource limitation), and only inputs below this limit were retained during dataset construction. Experiments were conducted using LLaMA-2-7B and Qwen-2-

[5] https://huggingface.co/princeton-nlp/AutoCompressor-Llama-2-7b-6k

Table 2: The ablation study results on MSMARCO using LLAMA-2-7B. Here, k represents the number of compressed tokens or average.

Methods	F1	EM	CR(k)	TP
ATAcompressor	**50.06**	**8.00**	**27.36x (4.18)**	1.29
w/o AAC	47.52	7.34	19.06x (6.00)	1.29
w/o Selective	40.83	3.51	24.59x (4.65)	1.29

7B as backbones, with all models trained using open-source code unless noted in §4.1.2. See §B and §C for details on datasets and training.

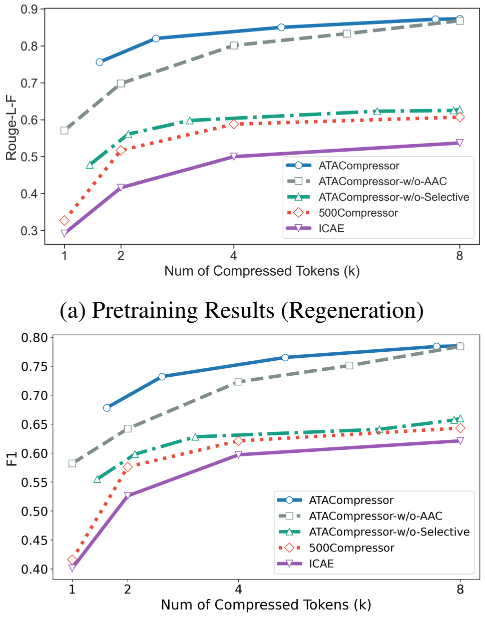
4.2 Main Results

Table 1 shows the performance of various methods across three benchmark datasets. ATACompressor consistently outperforms all other methods in both task performance (F1 and EM scores) and compression efficiency, highlighting its effectiveness and efficiency in long-context compression. A case study of the compression results can be found in §D. First of all, ATACompressor demonstrates significant advantages over non-compression methods, effectively alleviating the "lost in the middle" issue by focusing on key information within lengthy contexts. Furthermore, ATACompressor achieves the greatest relative improvement on HotpotQA, a dataset with the longest contexts and coarsest granularity. The relatively larger amount of irrelevant text highlights the advantage of ATACompressor's selective compression, demonstrating its strong ability to handle long-context scenarios with the selective encoder. Results on MSMARCO and SQUAD further demonstrate ATACompressor's capability to selectively preserve task-relevant information across varying context lengths. Meanwhile, ATACompressor consistently achieves high compression ratios across datasets, effectively compressing long contexts while preserving information quality. This is particularly obvious in datasets with longer contexts, such as HotpotQA. Its strong efficiency makes it well-suited for real-time or large-scale applications. Moreover, its consistent performance across different models highlights its adaptability.

4.3 Ablation Study

As shown in Table 2, we conduct two types of ablation studies:

(1) **w/o Adaptive Allocation Controller (AAC)**: This variant replaces the adaptive allocation controller with a fixed number of compressed tokens. As a result, both task performance and compression ratio drop. This highlights the importance of AAC in preserving essential information and dynamically adjusting compression rates based on the task. TP remains similar across settings, as AAC and the selective encoder run in parallel (§3.2).



(a) Pretraining Results (Regeneration)

(b) Finetuning Results (QA)

Figure 3: Performance on pretraining (regeneration) and finetuning (QA) tasks with varying numbers of compressed tokens using the LLAMA-2-7B model on HotpotQA.

(2) **w/o Selective**: This variant removes selective compression, applying compression uniformly to all context tokens. Without identifying task-relevant content, AAC adjusts compression rates based on total context length. This leads to clear declines in both performance and compression ratio, showing that without the selective encoder, the model fails to prioritize critical information, reducing overall compression effectiveness.

5 Analysis

We conduct experiments to further evaluate ATACompressor, using ICAE and 500Compressor as main baselines for their strong performance and generality. For ATACompressor and ATACompressor-w/o-Selective, the average number of compressed tokens k is controlled by direct adjusting the policy ratio r without training, while for other methods, k is adjusted by training separate models with different k values.

5.1 Impact of the Compressed Tokens' Number

The analysis in Figure 3 on the impact of the number of compressed tokens k demonstrates ATACompressor's robustness. As k decreases, ATACompressor shows a smaller performance drop compared to other methods, highlighting its ability

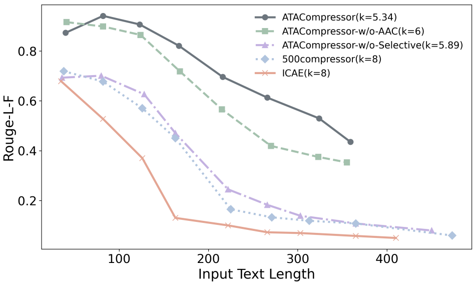


Figure 4: Performance on pretraining (regeneration) across varying input text lengths using the Qwen-2-7B. k represents the number of compressed tokens or average.

to handle varying compression levels with minimal performance loss. While all methods experience some degradation as k reduces, ATACompressor maintains relatively high task performance, even under tighter compression. In contrast, ICAE and 500Compressor exhibit sharp performance drops due to the lack of mechanisms for preserving task-relevant information. Additionally, both ATACompressor-w/o-AAC and ATACompressor-w/o-Selective also see performance drops as k decreases, underscoring the combined importance of the two key components. Together, these components enable ATACompressor to maintain competitive performance across different compression limitations.

5.2 Performance Across Input Text Lengths

Figure 4 illustrates ATACompressor's effectiveness across varying input lengths. Its lower performance on shorter texts stems from the policy yielding very few compressed tokens in such cases. This can be mitigated in practice by setting a minimum token threshold. Overall, ATACompressor shows strong robustness on longer texts, outperforming 500Compressor and ICAE as input length increases. In contrast, ATACompressor-w/o-AAC, ICAE, and 500Compressor use a fixed number of compressed tokens, performing well on shorter texts but degrading rapidly with longer inputs. ATACompressor-w/o-Selective shows a similar trend to ATACompressor but lacks the selective mechanism, resulting in significantly lower performance.

5.3 Performance of the Probe

Figure 5 shows that the adaptive allocation controller achieves high prediction accuracy across all datasets. Identifying whether a sentence is necessary for answering a query is inherently harder than judging document-level relevance, as sentence-level decisions lack broader context. This explains why HotpotQA, despite its longer inputs, yields lower prediction error—its document-level granularity provides richer context for identifying relevant chunks. In contrast, SQUAD's sentence-level granularity increases uncertainty, leading to higher error. These results highlight that higher granularity enables the selective encoder to leverage global context, improving relevance estimation and prediction accuracy. The detailed performance of the probe is shown in Table 6.

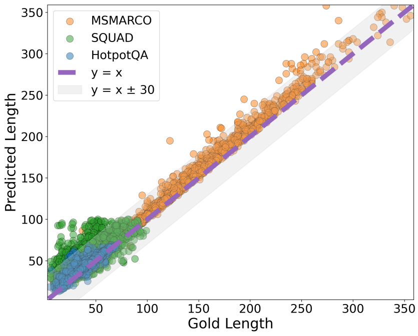


Figure 5: Comparison of gold (L_{Rel}) and predicted lengths ($\hat{L}_{Rel}$) across three datasets on Qwen-2-7B.

5.4 Computational Efficiency

Table 3: QA task efficiency evaluated on the HotpotQA using the LLaMA2-7B model on two A100-40G GPUs. k represents the number of compressed tokens or average.

Method	k	**Inference Time (ms)**	**GPU Mem. (GB)**
Closed Book	-	**156.99**	**18.79**
Original-Context	-	826.45	21.58
Selective-Context	-	819.67	23.82
LongLLMLingua	-	757.58	33.56
Autocompressor	15.33	337.84	25.56
ICAE	8.00	265.11	23.97
500Compressor	8.00	270.28	24.32
QGC	12.80	510.20	35.44
ATACompressor-w/o-AAC	8.00	254.18	24.30
ATACompressor-w/o-Selective	7.91	255.10	28.42
ATACompressor	7.59 1.62	255.08 254.89	28.66 28.49

Table 3 compares the efficiency of different methods for the QA task in terms of inference time and GPU memory cost. It shows that ATACompressor demonstrates excellent efficiency, with low inference time and GPU memory usage. The performance remains stable across a small range of k. Compared to Orginal-Context method, ATACompressor significantly reduces inference time. Compared to QGC, which uses a soft prompt framework for query-based compression, ATACompressor achieves lower inference times and GPU memory usage, demonstrating its efficiency.

Limitations

Despite the promising results, our approach still has several limitations:

Exploration in cross-model compression scenarios. One recent study (Rau et al., 2024) proposes adapting the traditional soft prompt method, ICAE (Ge et al., 2023), to the RAG setting, where a small model is used as the compressor and a larger LLM serves as the target model. Our method is naturally compatible with this setting, but we have not yet explored its application in such architectures. We leave this direction for future work.

Dependence on open-source models. Similar to other soft prompt–based approaches, our method relies on access to the internal representations and parameters of LLMs. This reliance limits its applicability in black-box scenarios, where model internals are not accessible or exposed.

Limited exploration of downstream applications. Our current experiments focus on QA tasks, following standard experimental paradigms adopted in prior work. This choice facilitates fair and meaningful comparisons with the baselines. However, this focus also limits the exploration of our method's potential in broader application scenarios. Investigating these directions in future work could further demonstrate the robustness and versatility of our approach.

References

Garima Agrawal, Sashank Gummuluri, and Cosimo Spera. 2024. Beyond-rag: Question identification and answer generation in real-time conversations. *arXiv preprint arXiv:2410.10136*.

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 preprint arXiv:2406.02376*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

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 preprint arXiv:2405.13792*.

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

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*.

Xiangjue Dong, Yibo Wang, Philip S Yu, and James Caverlee. 2023. Probing explicit and implicit gender bias through llm conditional text generation. *arXiv preprint arXiv:2311.00306*.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Tao Ge, Jing Hu, Lei Wang, 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*.

Kaan Gokcesu and Hakan Gokcesu. 2021. Generalized huber loss for robust learning and its efficient minimization for a robust statistics. *arXiv preprint arXiv:2108.12627*.

Cheng-Yu Hsieh, Yung-Sung Chuang, Chun-Liang Li, Zifeng Wang, Long T Le, Abhishek Kumar, James Glass, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and 1 others. 2024. Found in the middle: Calibrating positional attention bias improves long context utilization. *arXiv preprint arXiv:2406.16008*.

Yucheng Hu and Yuxing Lu. 2024. Rag and rau: A survey on retrieval-augmented language model in natural language processing. *arXiv preprint arXiv:2404.19543*.

Yizheng Huang and Jimmy Huang. 2024. A survey on retrieval-augmented text generation for large language models. *arXiv preprint arXiv:2404.10981*.

Luis Ibanez-Lissen, Lorena Gonzalez-Manzano, Jose Maria de Fuentes, Nicolas Anciaux, and Joaquin Garcia-Alfaro. 2024. Lumia: Linear probing for unimodal and multimodal membership inference a! acks leveraging internal llm states. *arXiv preprint arXiv:2411.19876*.

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

Tianjie Ju, Weiwei Sun, Wei Du, Xinwei Yuan, Zhaochun Ren, and Gongshen Liu. 2024. How large language models encode context knowledge? a layer-wise probing study. *arXiv preprint arXiv:2402.16061*.

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, Yinhong Liu, Yixuan Su, and Nigel Collier. 2024a. Prompt compression for large language models: A survey. *arXiv preprint arXiv:2410.12388*.

Zongqian Li, Yixuan Su, and Nigel Collier. 2024b. 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. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Zheng Liu, Chenyuan Wu, Ninglu Shao, Shitao Xiao, Chaozhuo Li, and Defu Lian. 2024b. Lighter and better: Towards flexible context adaptation for retrieval augmented generation. *arXiv preprint arXiv:2409.15699*.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset.

Erik Nijkamp, Tian Xie, Hiroaki Hayashi, Bo Pang, Congying Xia, Chen Xing, Jesse Vig, Semih Yavuz, Philippe Laban, Ben Krause, and 1 others. 2023. Xgen-7b technical report. *arXiv preprint arXiv:2309.03450*.

Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. 2024. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

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

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, and 1 others. 2023. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521*.

Hetong Wang, Pasquale Minervini, and Edoardo M Ponti. 2024a. Probing the emergence of cross-lingual alignment during llm training. *arXiv preprint arXiv:2406.13229*.

Zhengxiang Wang, Jordan Kodner, and Owen Rambow. 2024b. Evaluating llms with multiple problems at once: A new paradigm for probing llm capabilities. *arXiv preprint arXiv:2406.10786*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024a. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

Yichun Zhao, Shuheng Zhou, and Huijia Zhu. 2024b. Probe then retrieve and reason: Distilling probing and reasoning capabilities into smaller language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13026–13032.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2023. Pose: Efficient context window extension of llms via positional skip-wise training. *arXiv preprint arXiv:2309.10400*.

A Workflow Details

In this section, we provide detailed descriptions of the workflow. Figure 6 illustrates the procedures involved in the pre-training and fine-tuning stages, as outlined in § 3.5.

A.1 Pretrain

During pretraining, we jointly optimize the selective encoder (φ) and the probe (ζ). The selective encoder (φ) is trained to extract and effectively compress the relevant portions of the context C required to query Q, while the probe (ζ) is trained to accurately predict the relevant text length $\hat{L}_{Rel}$. The overall loss function for this stage combines the encoder loss ($\mathcal{L}_\varphi$) and the probe loss ($\mathcal{L}_\zeta$), with a weighting factor λ to balance their contributions. The total loss is defined as:

$$\mathcal{L}_{\text{pretrain}} = \mathcal{L}_\varphi + \lambda \cdot \mathcal{L}_\zeta \tag{8}$$

Encoder Loss ($\mathcal{L}_\varphi$) The encoder loss is formulated using a cross-entropy objective to measure the alignment between the predicted token distribution and the gold-truth token distribution. C_{Rel} is a context formed by concatenating chunks from the context C that are explicitly labeled as relevant for addressing the task Q, representing the annotated gold truth of $\hat{C}_{Rel}$. We ensure that the chunk granularity of $\hat{C}_{Rel}$ is consistent with that of C_{Rel}. During training, teacher forcing (Ge et al., 2023; Li et al., 2024b) is used to guide the LLM in reconstructing the gold-truth sequence by providing true tokens as input, enhancing the model's ability to predict the correct sequence. The loss is defined as follows:

$$\mathcal{L}_\varphi = -\sum_{j=1}^{n} \log P(w_j \mid \mathbf{KV}, [\text{BOS}], w_{1:j-1}; \Theta_{\text{LLM}}, \Theta_{\text{LoRA}}) \tag{9}$$

Where w_j is the j-th token in the C_{Rel}, and $\mathbf{KV}$ is the key-value representations of the compressed tokens generated by the selective encoder (φ), passed to the target LLM. The sequence starts with the beginning-of-sequence token $[\text{BOS}]$ as a signal for pretraining, while Θ_{LLM} denotes the frozen parameters of the target LLM, and Θ_{LoRA} the trainable parameters of the LoRA adapter in the selective encoder. $P(w_j \mid \cdot)$ represents the predicted probability distribution of the j-th token. The sequence ends when the model generates the end-of-sequence token $[\text{EOS}]$, implicitly included in the token generation process.

Probe Loss ($\mathcal{L}_\zeta$) The probe loss is calculated using the huber loss (Gokcesu and Gokcesu, 2021), which measures the error between the estimated length $\hat{L}_{Rel}$ and the gold-truth length L_{Rel} (length of C_{Rel}), δ is a hyperparameter:

$$\mathcal{L}_\zeta = \begin{cases} \frac{1}{2}(\hat{L}_{Rel} - L_{Rel})^2 & \text{if } |\hat{L}_{Rel} - L_{Rel}| \leq \delta, \\ \delta(|\hat{L}_{Rel} - L_{Rel}| - \frac{\delta}{2}) & \text{otherwise} \end{cases} \tag{10}$$

A.2 Finetune

During the finetuning, the selective encoder (φ) is further trained for downstream tasks. The target LLM generates task-specific outputs based on the key-value representations of the compressed tokens. This process also employs teacher forcing. The loss function during finetuning is defined as:

$$\mathcal{L}_F = -\sum_{j=1}^{n} \log P(a_j \mid \mathbf{KV}, q_{1:m}, a_{1:j-1}; \Theta_{\text{LLM}}, \Theta_{\text{LoRA}}) \tag{11}$$

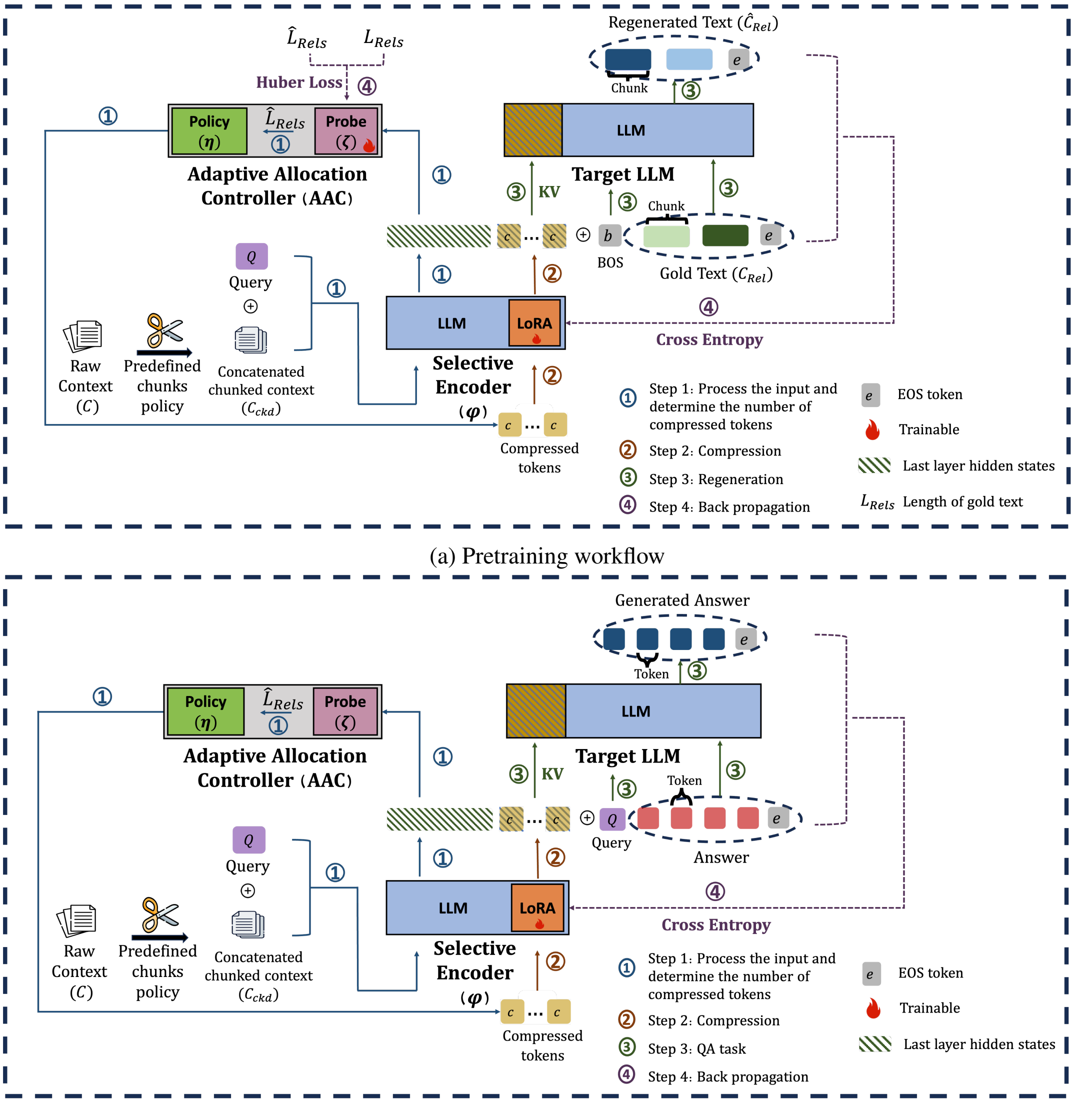
where a_j denotes the j-th gold-truth answer token for the task, and $q_{1:m}$ is the query Q.

A.3 Inference

During inference, all components' parameters are frozen. As illustrated in §3.2, the concatenated chunked context along with the query are compressed by the selective encoder into a set of compressed tokens. The number of compressed tokens is determined by the adaptive allocation controller. And then, the key and value representations of the compressed tokens are passed to the target LLM to generate outputs in two situations: the regeneration of C_{Rel} (triggered by the [BOS] token) and the generation of answers based on the query. For regeneration, the target LLM predicts each token $\hat{w}_i$ in the sequence using the probability distribution conditioned on the compressed representations and previously generated tokens:

$$\hat{w}_i = \arg\max_{\hat{w}_i} P(\hat{w}_i \mid \mathbf{KV}, [\text{BOS}], \hat{w}_{1:i-1}; \Theta_{\text{LLM}}) \tag{12}$$

For generating answers, the target LLM produces each token $\hat{a}_j$ in the task-specific output conditioned on the compressed representations, the input query $q_{1:m}$, and previously generated answer



(a) Pretraining workflow

(b) Fintuning workflow

Figure 6: Traing workflows

tokens:

$$\hat{a}_j = \arg\max_{\hat{a}_j} P(\hat{a}_j \mid \mathbf{KV}, q_{1:m}, \hat{a}_{1:j-1}; \Theta_{\text{LLM}}) \tag{13}$$

B Dataset Details

B.1 Dataset Information

The experiments are based on the three datasets:

- **HotpotQA** (Yang et al., 2018): HotpotQA[6] is a multi-hop question answering dataset where the answer requires information from more than one document. We use it to evaluate models at the **document level**, where the LLM needs to aggregate information from multiple docs to generate a correct answer.

- **MSMARCO** (Nguyen et al., 2016): MS-MARCO (Question Answering v2.1) [7] is a high-quality question answering dataset curated by Microsoft. In this study, we employ the dataset to assess models at the **passage level**, where the LLM is tasked with synthesizing information from relevant passages to produce the correct answer.

- **SQUAD** (Rajpurkar et al., 2018): SQUAD[8] is a question-answering dataset where each question is paired with a passage, and the answer is typically a span of text found within that passage. We utilize SQUAD is structured to assess models at the **sentence level**, demand-

[6] https://hotpotqa.github.io/

[7] https://huggingface.co/datasets/microsoft/ms_marco/viewer/v2.1

[8] https://huggingface.co/datasets/rajpurkar/squad_v2

ing the LLM to aggregate information from sentences to generate a correct answer.

B.2 Data Preprocessing

As described in §3.2, the context C is first segmented into chunks $C_1, C_2, \ldots, C_n$ using a predefined strategy. These chunks are then concatenated and processed by the selective encoder (φ) together with the query Q.

It is worth noting the following: (1) the chunking process occurs before the selective encoder's processing, meaning it is a preprocessing step rather than a task for the selective encoder (φ); (2) the chunking process is the procedure of labeling the context C according to a predefined chunking policy (e.g., using passages as the chunking unit, where each passage of C is enclosed within `<PA></PA>` tags). All the chunks are then concatenated to form a preprocessed context C_{ckd}, which is input into the selective encoder (φ) along with the query Q and processed in a single pass rather than being processed individually in multiple passes; (3) if the length of raw context C exceeds the selective encoder's input length limit, we can first divide C into smaller segments and then apply the selective encoder (φ) to compress each segment individually. This segmentation process is different from the chunking process we mentioned above.

Examples of preprocessed data following this procedure are provided in Figure 7, Figure 8 and Figure 9.

C Implementation Details

In this section, we provide a detailed implementation. As described in Section 4.1.4, to ensure fair comparison with baseline methods, the maximum input length was constrained to 600 tokens. During dataset construction, only input samples that fell within this limit were retained. All experiments were conducted using LLaMA-2-7B and Qwen-2-7B as backbone models. Unless otherwise noted in Section 4.1.2, all models were trained using open-source implementations. For ATACompressor, we set the hyperparameter λ in Eq.(8) to 10^{-4} and δ in Eq.(10) to 10 during pretraining. The policy ratio r in Eq. (5) was randomly selected from the set 1, 5, 10, 20, 50 for each training batch in the pretraining stage. During finetuning and evaluation, r was fixed at 10 unless specified otherwise. The maximum number of compressed tokens, denoted by k_{max}, was set to 8 for both training and inference phases.

We evaluated the generation quality using several widely adopted automatic metrics, including ROUGE, BLEU, Exact Match, and F1 score. The evaluation was implemented in Python, leveraging the NLTK (version 3.8.1) and rouge (version 1.0.1) libraries.

Further hyperparameter configurations and implementation details can be found in Table 4 and Table 5. Meanwhile, figure 10 shows the input prompt for the selective encoder φ.

D Case Study

Table 7 presents the results of a case study comparing ICAE and ATACompressor. Unlike ICAE, which performs full-text compression, ATACompressor selectively compresses relevant context according to task-specific needs, ensuring critical information is preserved and reducing the risk of key errors. For instance, in Question 3, ICAE Introduced a critical error by incorrectly stating "over the first half of the 11th century" instead of the correct text "in the first half of the 10th century". Additionally, ATACompressor employs adaptive compression, dynamically adjusting token usage based on the length of relevant content. This mechanism optimizes resource efficiency while maintaining high performance across tasks.

Illustrative Preprocessed Sample from the HotpotQA Dataset

Question: "Which writer was from England, Henry Roth or Robert Erskine Childers?"

Context: "<PA> Asgard is a 51 ft gaff rigged yacht. She was owned by the English-born writer and Irish nationalist Erskine Childers and his wife Molly Childers. She is most noted for her use in the Howth gun-running of 1914. </PA> <PA> Henry Roth (February 8, 1906 – October 13, 1995) was an American novelist and short story writer. </PA> <PA> The R509 road, following part of the Childers Road (named after Erskine Childers), is a regional road in Ireland, running through the southeastern side of Limerick City. It forms what is somewhat akin to an inner ring road (albeit mostly two-lane only). </PA> <PA> Mary Alden Osgood Childers, MBE (14 December 1875 – 1 January 1964) was an American-born Irish writer and Irish nationalist. She was the daughter of Dr Hamilton Osgood and Margaret Cushing Osgood of Beacon Hill, Boston, Massachusetts. Her older sister was Gretchen Osgood Warren. Molly married the writer and Irish nationalist, Robert Erskine Childers. Their son, Erskine Hamilton Childers, became the fourth President of Ireland. </PA> <PA> Gretchen Osgood Warren (19 March 1868 – September 1961), the wife of Fiske Warren, was an actress, singer and poet. The daughter of Dr. Hamilton Osgood and Margaret Cushing Osgood of Beacon Hill, Boston, Massachusetts, her younger sister was Mary Alden Childers, the wife of writer and Irish nationalist Robert Erskine Childers. Her nephew Erskine Hamilton Childers served as the fourth President of Ireland from 1973–74. </PA> <PA> Robert Caesar Childers (1838 – 25 July 1876) was a British Orientalist scholar, compiler of the first Pāli–English dictionary. Childers was the husband of Anna Barton of Ireland. He was the father of Irish nationalist Robert Erskine Childers and grandfather to the fourth President of Ireland, Erskine Hamilton Childers. </PA> <PA> Robert Erskine Childers DSC (25 June 1870 – 24 November 1922), universally known as Erskine Childers, was a British writer, whose works included the influential novel "The Riddle of the Sands", and a Fenian revolutionary who smuggled guns to Ireland in his sailing yacht "Asgard". He was executed by the authorities of the nascent Irish Free State during the Irish Civil War. He was the son of British Orientalist scholar Robert Caesar Childers; the cousin of Hugh Childers and Robert Barton; and the father of the fourth President of Ireland, Erskine Hamilton Childers. </PA> <PA> The Irish Bulletin was the official gazette of the government of the Irish Republic. It was produced by the Department of Propaganda during the Irish War of Independence. and its offices were originally located at No. 6 Harcourt Street, Dublin. The paper’s first editor was Desmond FitzGerald, until his arrest and replacement by Robert Erskine Childers. "The Bulletin" appeared in weekly editions from 11 November 1919 to 11 July 1921. </PA>"

Gold context: "<PA> Henry Roth (February 8, 1906 – October 13, 1995) was an American novelist and short story writer. </PA> <PA> Robert Erskine Childers DSC (25 June 1870 – 24 November 1922), universally known as Erskine Childers, was a British writer, whose works included the influential novel "The Riddle of the Sands", and a Fenian revolutionary who smuggled guns to Ireland in his sailing yacht "Asgard". He was executed by the authorities of the nascent Irish Free State during the Irish Civil War. He was the son of British Orientalist scholar Robert Caesar Childers; the cousin of Hugh Childers and Robert Barton; and the father of the fourth President of Ireland, Erskine Hamilton Childers. </PA> "

Answer: "Robert Erskine Childers DSC"

Figure 7: Illustrative Preprocessed Sample from HotpotQA.

Illustrative Preprocessed Sample from the MSMACRO Dataset

Question: "Is Bob Hewitt a citizen of a different country than Ray Ruffels?"

Context: "<PA> The presence of communication amid scientific minds was equally important to the success of the Manhattan Project as scientific intellect was. The only cloud hanging over the impressive achievement of the atomic researchers and engineers is what their success truly meant; hundreds of thousands of innocent lives obliterated. </PA> <PA> The Manhattan Project and its atomic bomb helped bring an end to World War II. Its legacy of peaceful uses of atomic energy continues to have an impact on history and science. </PA> <PA> Essay on The Manhattan Project - The Manhattan Project The Manhattan Project was to see if making an atomic bomb possible. The success of this project would forever change the world forever making it known that something this powerful can be manmade. </PA> <PA> The Manhattan Project was the name for a project conducted during World War II, to develop the first atomic bomb. It refers specifically to the period of the project from 194 . . . 2-1946 under the control of the U.S. Army Corps of Engineers, under the administration of General Leslie R. Groves. </PA> <PA> versions of each volume as well as complementary websites. The first website–The Manhattan Project: An Interactive History–is available on the Office of History and Heritage Resources website, http://www.cfo.doe.gov/me70/history. The Office of History and Heritage Resources and the National Nuclear Security </PA> <PA> The Manhattan Project. This once classified photograph features the first atomic bomb — a weapon that atomic scientists had nicknamed Gadget.. The nuclear age began on July 16, 1945, when it was detonated in the New Mexico desert. </PA> <PA> Nor will it attempt to substitute for the extraordinarily rich literature on the atomic bombs and the end of World War II. This collection does not attempt to document the origins and development of the Manhattan Project. </PA> "

Gold context: "<PA> Raymond Owen R̈ayR̈uffels (born 23 March 1946 in Sydney) is an Australian former professional tennis player and coach. </PA> <PA> Robert Anthony John Hewitt (born 12 January 1940) is a former professional tennis player from Australia. In 1967, after marrying a South African, he became a South African citizen. He has won 15 major titles and a career Grand Slam in both men's and mixed doubles.</PA>"

Answer: "yes"

Figure 8: Illustrative Preprocessed Sample from MSMACRO.

Illustrative Preprocessed Sample from the SQUAD Dataset

Question: "When was the Duchy of Normandy founded?"

Context: "<PA> In the course of the 10th century, the initially destructive incursions of Norse war bands into the rivers of France evolved into more permanent encampments that included local women and personal property. </PA> <PA> The Duchy of Normandy, which began in 911 as a fiefdom, was established by the treaty of Saint-Clair-sur-Epte between King Charles III of West Francia and the famed Viking ruler Rollo, and was situated in the former Frankish kingdom of Neustria. </PA> <PA> The treaty offered Rollo and his men the French lands between the river Epte and the Atlantic coast in exchange for their protection against further Viking incursions. </PA> <PA> The area corresponded to the northern part of present-day Upper Normandy down to the river Seine, but the Duchy would eventually extend west beyond the Seine. </PA> <PA> The territory was roughly equivalent to the old province of Rouen, and reproduced the Roman administrative structure of Gallia Lugdunensis II (part of the former Gallia Lugdunensis). </PA>"

Gold context: "<PA> The Duchy of Normandy, which began in 911 as a fiefdom, was established by the treaty of Saint-Clair-sur-Epte between King Charles III of West Francia and the famed Viking ruler Rollo, and was situated in the former Frankish kingdom of Neustria. </PA>"

Answer: "911"

Figure 9: Illustrative Preprocessed Sample from SQUAD.

Input prompt for the selective encoder φ

<QUESTION> {{Question}} </QUESTION> <CONTEXT> {{Context}} </CONTEXT> <INST> Please identify and extract the <PA> sections that can answer the question (which may not be unique) </INST>

Figure 10: Input prompt for the selective encoder φ

Table 4: Hyperparameters for Pretraining

Hyperparameter	**Assignment**
learning Rate	1e-5
lr scheduler type	constant with warmup
warmup steps	300
weight decay	0.2
overall batch size	16
optimizer	AdamW
epochs	3
LoRa layers	all linear layers
LoRa r	64
LoRa alpha	32
LoRa dropout	0.2
LoRa bias	None
mixed-precision	fp16
GPU	4 × A100 40GB
max context length	600
λ in Eq. (8)	1e-4
policy ratio r	randomly chosen from {1, 5, 10, 20, 50} per batch.
maximum number of compressed tokens k_{max}	8

Table 5: Hyperparameters for Finetuning

Hyperparameter	**Assignment**
learning Rate	1e-5
lr scheduler type	constant with warmup
warmup steps	300
weight decay	0.2
overall batch size	16
optimizer	AdamW
epochs	1
LoRa layers	all linear layers
LoRa r	64
LoRa alpha	32
LoRa dropout	0.2
LoRa bias	None
mixed-precision	fp16
GPU	4 × A100 40GB
max context length	600
policy ratio r	10
maximum number of compressed tokens k_{max}	8

Dataset	**Mean Gold Length**	**Mean Predicted Length**	**MAE**
HotpotQA	239.50	243.72	4.52
MSMARCO	137.76	145.30	7.63
SQUAD	40.35	52.73	13.99

Table 6: Mean absolute error (MAE) of probe predictions on different datasets

Table 7: Case study of ATACompressor and ICAE on Qwen-2-7B. red highlights errors in key information, typically referring to incorrect statements directly impacting the answer. yellow indicates mistakes in less critical details that do not directly affect the core answer. green denotes hallucinations, where the text contains information not present in the original source. blue signals information loss, meaning less important content from the original text is missing. Finally, brown indicates paraphrasing, where the original content is reworded without changing its meaning. Also, k represents the number of compressed tokens or average.

Original Text (The input to ICAE does not include the labels <PA> and </PA>.)
<PA> The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave their name to Normandy, a region in France. </PA> <PA> They were descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Francia. </PA> <PA> Through generations of assimilation and mixing with the native Frankish and Roman-Gaulish populations, their descendants would gradually merge with the Carolingian-based cultures of West Francia. </PA> <PA> The distinct cultural and ethnic identity of the Normans emerged initially in the first half of the 10th century, and it continued to evolve over the succeeding centuries. </PA>

Question	**ATACompressor Regeneration Output (The value of k is 2, 4, and 2 for the following three questions, respectively.)**	**ICAE Regeneration Output (k=8)**
Q1: In what country is Normandy located?	The Normans (Norman ; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave their name to Normandy ; a region in France.	The Normans (Normand; Nourmands : French ; Latin: Normanni) were the people who were the Normans in the 10th and 11th centuries who gave their name to Normandy, a region in France. They descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Francia. Through generations of assimilation and mixing of the Norman and Frankish cultures , their first- and second-generation descendants would inherit the Roman-Gallic cultural identity of the Carolingian Normans . The distinctive cultural and ethnic identity emerged gradually over the first half of the 11th century, and it was the native Frankish culture that initially dominated Western Europe.
Q2: From which countries did the Norse originate?	The Normans (Norman ; French: Normands; Latin: Normanni) were the people who in the 10th and 11th gave their name to Normandy ; a region in France. They were descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland, and Norway who, under the leadership of Rollo , agreed to swear fealty to King Charles III of West Francia.	
Q3: What century did the Normans first gain their separate identity?	The distinctive cultural and ethnic identity of the Normans emerged initially in the first half of the 10th century, and it continued to evolve over the succeeding centuries.	