Attention-Guided Context Pruning in Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

While RAG demonstrates remarkable capabilities in LLM applications, its effectiveness is hindered by the ever-increasing length of retrieved contexts, which introduces information redundancy and substantial computational overhead. Existing context pruning methods, such as LLMLingua, lack contextual awareness and offer limited flexibility in controlling compression rates, often resulting in either insufficient pruning or excessive information loss. In this paper, we propose ATTENTIONRAG, an attention-guided context pruning method for RAG systems. The core idea of ATTENTIONRAG lies in its attention focus mechanism, which reformulates RAG queries into a next-token prediction paradigm. This mechanism isolates the query's semantic focus to a single token, enabling precise and efficient attention calculation between queries and retrieved contexts. Extensive experiments on LongBench and Babilong benchmarks show that ATTENTIONRAG achieves up to 6.3x context compression while outperforming LLMLingua methods by around 10% in key metrics.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2021) demonstrated remarkable capabilities in Large Language Model (LLM) applications such as reasoning (Huang and Chang, 2023), question-answering (Rajpurkar et al., 2016; Wang et al., 2024), and open-ended text generation (Que et al., 2024). Although LLMs have demonstrated superior performance, they often lack domain knowledge. By directly leveraging external documents, RAG provides a lightweight, cost-efficient solution to expand the LLMs' knowledge base without retraining their parameters.

While effective, RAG-based systems encounter significant challenges in handling long contexts. As the number of retrieved documents grows, the context becomes excessively long, introducing large amounts of redundant and irrelevant information. This issue, as highlighted by Shi et al. (2024), can lead to hallucinations and a decline in the performance of the LLM (Chiang and Cholak, 2022).

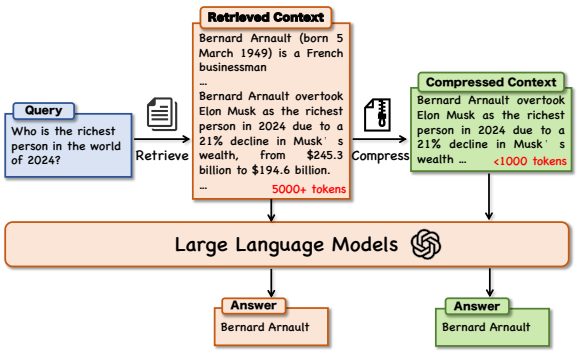

Figure 1: Illustration of RAG context compression

To mitigate these issues, recent work has explored context compression techniques (Jiang et al., 2023; Cheng et al., 2024; Verma, 2024). For example, LLMLingua (Jiang et al., 2023) compresses prompts by using a budget controller that allocates varying compression ratios to different components of the prompt, such as instructions and demonstrations. However, these methods lack context-awareness, making it challenging to determine the optimal compression ratio for a given LLM, resulting in context redundancy or over-compression.

Although attention mechanisms are central to LLMs for content selection, there is limited research on using them specifically for context pruning during retrieval. This gap is primarily due to two fundamental challenges that hinder the effectiveness of attention in the retrieval architecture: (1) The long nature of RAG contexts exacerbates attention dilution (Hsieh et al., 2024; Zhu et al., 2024). When the context is excessively long, the attention scores are spread thin, causing the model to allocate less focus to any single token. This dilution of attention reduces the ability of the model to concentrate on the most relevant parts of the

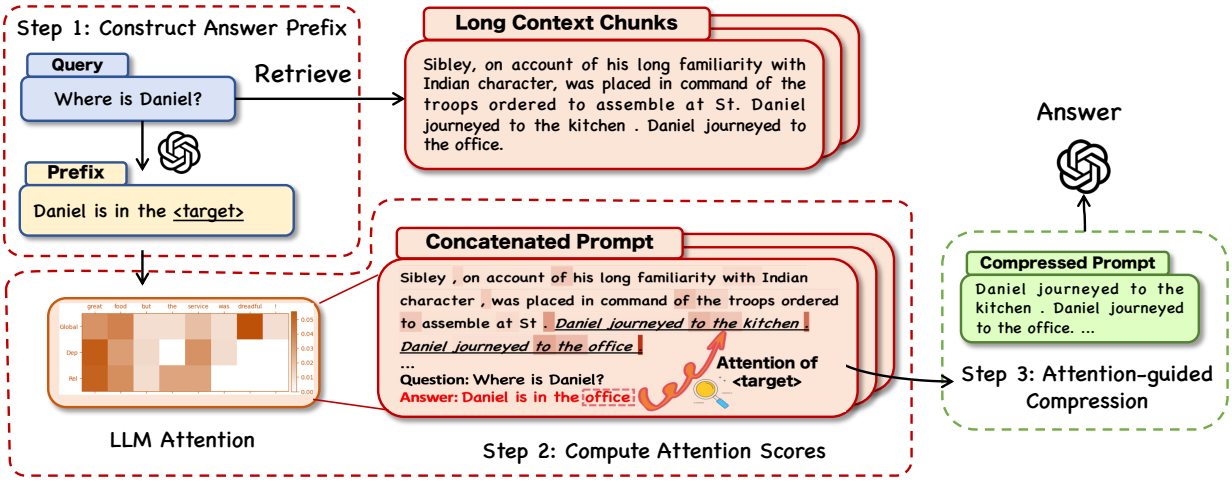

Figure 2: Illustration of ATTENTIONRAG: 1) We first generate a hint prefix using the original query; if generation fails, we fallback to a fixed hint prefix. 2) The long context is then divided into smaller chunks for localized processing. 3) For each chunk, we batch the input with the hint prefix (1 token only) and use an LLM to compute attention features over the full context, focusing on the generated anchor token. 4) Using these attention features, we identify and extract the most relevant content from each chunk to construct a compressed context.

retrieved context; and (2) The sentence-based nature of the query makes it challenging to directly align with critical content in the context. The attention score distribution varies across tokens within a sentence, with some tokens focusing on semantic information while others attend to details (Clark et al., 2019; Gu et al., 2022). For instance, in the sequence "Mary is in the car," the attention score of "in" tends to focus on semantic relationships, while "car" focuses more on a specific detail. Therefore, the query lacks the explicit or semi-structured reference points (such as keywords or pivotal terms) needed to guide the attention mechanism toward the most salient parts of the context.

In this paper, we propose a novel methodology called ATTENTIONRAG (Attention-Guided Retrieval-Augmented Generation), which improves the relevance of the information extracted from the retrieved context by leveraging attention scores from intermediate layers of LLMs. To enable attention across queries and context, ATTENTIONRAG introduces an attention focus mechanism that isolates the query's semantic focus to a single token, enabling precise and efficient attention computation between queries and retrieved contexts in a single pass. More specifically, for each query (e.g, "Where is Daniel?"), we construct an answer hint prefix (e.g., "Daniel is in the ____") in a <next-token-prediction> format. Next, we take the retrieved context, the query, and the constructed answer prefix as input to an LLM. The missing next token in the prefix stands for the focal point of the query, guiding the LLM's attention to each token in the context. Finally, we produce a compressed context by selecting sentences from the original context with the top-k attended tokens.

Extensive experiments on LongBench (Bai et al., 2024) and BABILong (Kuratov et al., 2024) demonstrate that our method achieves up to 6.3x context compression while maintaining or exceeding the performance of uncompressed contexts. The results suggest that ATTENTIONRAG not only facilitates the extraction of relevant information but also enhances the model's reasoning capabilities. Particularly, it achieves these benefits without requiring additional training, making it highly adaptable across different models and practical for real-world applications.

Our key contributions are as follows:

- We propose a lightweight, transferable, and question-aware method for long-context pruning in RAG systems.
- We introduce a novel attention focus mechanism by reformulating RAG queries into a next-token prediction template, enabling precise and efficient computations of attention between queries and retrieved contexts.
- We conduct extensive experiments on LongBench and Babilong benchmarks. Results demonstrate the effectiveness of ATTENTIONRAG in long-context RAG systems.

2 Preliminaries

2.1 Retrieve-Augmented Generation

Retrieve-Augmented Generation (RAG) is a framework that enhances the capabilities of LLMs by integrating external knowledge through retrieval. A RAG system typically consists of two components: a *retriever*, which fetches relevant documents, called contexts, from a large corpus based on a query, and a *generator*, which generates an answer using both the retrieved context and the model's internal knowledge. This combination enables more accurate and contextually relevant outputs, especially for tasks requiring detailed or up-to-date information that might not be present in the model's training data.

2.2 Attention Mechanism in LLMs

The attention mechanism is a key component in modern LLMs, allowing the model to focus on different parts of the input sequence (Vaswani et al., 2023). For a given context c, the mechanism calculates a score for each token $t \in c$ based on its relevance to other tokens:

$$\text{Attention}_l(c,t) = \text{Softmax}\left(\frac{Q_l K_l^T}{\sqrt{d_k}}\right) V_l \quad (1)$$

where: Q_l is the query matrix at layer l, K_l is the key matrix at layer l, V_l is the value matrix at layer l, d_k is the dimensionality of the key vectors.

The scores indicate how much "focus" the token receives from the model, providing insights into which tokens are most relevant in a given context. This enables it to be used for text compression, where selecting tokens with high attention scores can reduce the input to the model while retaining the most important information (Tarzanagh et al., 2023). Although attention is central to LLMs, it suffers from dilution in handling long contexts (Hsieh et al., 2024).

3 Method

In this work, we propose a novel approach to optimize RAG systems by compressing the retrieved context without compromising performance.

Problem Formulation Given a query q, a RAG system retrieves a relevant set of documents $\{d_1, \ldots, d_K\}$ from a text corpus D. The retrieved documents are concatenated into a retrieved context $c = d_1 \oplus \ldots \oplus d_N$. A large language model $P_\phi(a|q,c)$ takes the question and the retrieved context c as input and produces an answer a to the question. Our goal is to compress the retrieved context c into a dense one c' such that $|c'| \ll |c|$ while the LLM maintains the quality of generated answers when taking c' as input.

In this paper, we propose a novel attention-guided context pruning method called ATTENTIONRAG. The key idea of ATTENTIONRAG is reformulating each RAG query into a next-token-prediction template (called answer hint prefix). This strategy allows the LLM to calculate the query-context attention through one token, therefore significantly improving the alignment between query and context, and reducing the time complexity for attention calculation.

Figure 2 shows the overall structure of our method. The pipeline involves three key steps: First, we generate an answer hint prefix for each query (§3.1); Next, the generated prefix is appended to the original query and context as input to the LLM. The LLM is instructed to predict the follow-up token to the answer prefix, obtaining the attention scores (§3.2). Finally, we perform attention-guided compression: using attention scores from a language model, we identify and retain the most relevant parts of the retrieved context. Each of the steps is elaborated in the following sections.

3.1 Construct Answer Hint Prefix

To improve the alignment between the query and context, we associate each query with an answer hint prefix that allows the LLM to calculate the query-context attention through one focal token.

For each query, an *answer hint prefix* is defined as an incomplete answer to the query in a *next-token-prediction* format, where the blank token to be predicted serves as the focal token of the query, directing the LLM's attention to the most relevant parts of the context. For instance, as shown in Figure 2, for the query "Where is Daniel?", the corresponding answer hint prefix can be "Daniel is in the ___". When we take the context, query, and the answer hint prefix as input to the LLM, the token to be predicted by the model, such as "park", becomes the focal point of the query, directing the model's attention to the most relevant parts of the context and ensuring that the attention calculation focuses on the crucial information related to the query.

We prompt the LLM to construct the answer

hint prefix[1]. In detail, according to the query's grammatical attributes, we categorize the answer hint prefix into two types: empty and non-empty ones. For example, for queries like wh-questions, the hint prefix can be derived from the query itself. In contrast, queries like yes/no-questions, where the answer's first token is "Yes/No", already align with the next-token-prediction paradigm. However, some type of questions may not have such hint prefix (like summarization task), we also provide a fixed hint prefix that prompt the model to output the keyword of context, ensuring robustness of our method. Our fixed hint prefix is: *"Please output the most relevant keyword or phrase that is relevant to the answer of the question."* Leveraging the semantic understanding capabilities of LLMs, we prompt them with example answer hint prefix of various types questions to automatically determine the type and generate the answer prefix hint.

By focusing attention on the focal token, this approach enhances both the precision and efficiency of the attention mechanism. The single-token focus accelerates computations by reducing the number of tokens involved in attention calculations. Simultaneously, concentrating on a target token—such as "car" in the sentence "Daniel is in the car"—enables the model to effectively identify and prioritize the most relevant information in the context.

3.2 Compute Attention Features from LLM

In this part, we aim to compute attention features based on the focal token. To address the issue of diluted attention scores, we divide the retrieved context C produced by the RAG framework into smaller chunks. We adopt a uniform chunking strategy, assuming each chunk consists of m tokens. Let $c_1, \ldots, c_n$ represent the resulting chunks, where $n = \lceil |C|/m \rceil$. For each chunk c_j, we concatenate the chunked context, query, and instruction (we instruct the LLM to generate "none" after the hint prefix if the chunk is irrelevant, which will be used in §3.3) and the generated answer hint prefix and feed this into the LLM. The LLM is then instructed to perform next-token prediction and compute the attention scores.

We define a_j as the first token generated following the answer hint prefix in chunk j. The attention feature A_j for a_j is computed as the sum of attention scores over all layers of the model, focusing on context tokens:

$$A_j = \sum_{l=0}^{L} \text{Attention}_l(c_j, a_j) \tag{2}$$

where L is the total number of layers in the model, and $\text{Attention}_l(c_j, a_1)$ is the attention score at layer l for the token a_1 relating to the context chunk c_j. This score is computed by the self-attention mechanism at each layer, capturing both local and global dependencies in the input.

The total attention feature A_i reflects the model's focus on the most relevant components of the input when generating the first token, and is the sum of the attention values across all layers. We choose to sum across all layers for analysis because the attention distribution in each layer can vary depending on the task. For easier tasks, earlier layers may already capture sufficient information to generate the final answer, while for more difficult tasks, the model might rely on the later layers (Jin et al., 2025). Since the function of each layer can vary from task to task, focusing on a single layer or a subset of layers could introduce bias in the attended information. To mitigate this issue, we sum the attention across all layers, which helps reduce task-specific bias. The choice of attention layers will be further explored in the ablation study in §5.4.

3.3 Compress with Attention

For each chunk, after generating the focal token, we first check whether this token is "none." If this is the case, we skip the chunk, as it is deemed irrelevant to the task. If the focal token is valid, we proceed by identifying the tokens in the context that have the highest attention features with respect to the focal token. These attention features represent how much each token in the context is relevant to the focal token, which serves as the focal point of the query.

Next, we select the top-k tokens based on their attention features, as these tokens are considered the most relevant to the focal token. To ensure that the context used for generating the final response is both relevant and concise, we focus on the sentences that contain these top-k tokens. By selecting these sentences, we retain the information most pertinent to the focal token. These selected sentences are then concatenated to form a compressed

[1]We use GPT-4o Mini for the generation, the prompt detail and fixed hint prefix can be referred to §B.1 and §B.2.

context c'_j.

$$c'_j = \text{Concat}\left(\left\{s \mid t_r \in \text{Top-}k(A_j) \text{ and } t_r \in s\right\}\right) \quad (3)$$

where s denotes a selected sentence.

3.4 Time Efficiency and Batch Generation

Since we employ a next-token prediction paradigm, only one focal token needs to be generated for each chunk. Furthermore, as each chunk is processed independently, batch generation can be used to accelerate the process. This approach results in high time efficiency. Moreover, we can use quantified model to further accelerate the compression process. We provide the pseudocode of our method in Algorithm 1 in Appendix.

4 Experimental Setup

In this experiment, we evaluate the efficacy of ATTENTIONRAG in long context compression.

4.1 Datasets

Due to fluctuations in the experimental results, each experiment was conducted three times, and the final result is the average of these trials.

Long Context Reasoning For our experiments, we incorporate datasets TriviaQA (Joshi et al., 2017), HotpotQA (Yang et al., 2018), and 2WikiMQA (Ho et al., 2020) from Longbench (Bai et al., 2024): TriviaQA assesses fact retrieval over long contexts, HotpotQA emphasizes multi-hop reasoning across dispersed clues, and 2WikiMQA tests the model's ability to synthesize information from multiple sources.

Babilong (Kuratov et al., 2024): The BABILong benchmark provides a comprehensive framework for evaluating an LLM's ability to reason and retrieve over long contexts. To assess ATTENTIONRAG's capabilities on relatively shorter and sparse content, we select test splits of 1kqa1, 2kqa1, and 4kqa1 tokens to demonstrate its performance across different circumstances.

Long Context Summarization To demonstrate the robustness of ATTENTIONRAG, we also evaluate it on GovReport (Huang et al., 2021) from Longbench. In this task, we employ fixed hint prefix, showing the stability of our method.

4.2 Metrics

We measure the effectiveness of compression using three metrics:

Exact Match (EM): Measures the percentage of predicted answers that exactly match the ground-truth answers.

LLM-as-a-judge scores: We leverage GPT-4 (et al., 2024) to assess the correctness of the model-generated answers. Specifically, we input the question, the model-generated answer, and the ground truth answer into GPT-4o, asking it to determine whether the provided answer is correct. The prompt used for the scoring can be found in §B.

Compression Ratio (CR): The ratio of tokens in the original context to the compressed context, defined as $CR = \frac{\text{\# tokens in original context}}{\text{\# tokens in compressed context}}$.

4.3 Baselines

We compare ATTENTIONRAG with state-of-the-art context compression baselines, including LLMLingua2 (Pan et al., 2024) and LongLLMLingua (Jiang et al., 2024), which represent question-unaware and question-aware approaches, respectively. This diverse selection underscores the robustness of ATTENTIONRAG. For fair comparison, we match or exceed their compression rates. We also include the uncompressed context ("Original Prompt"), a random compression baseline ("Random"), and a vanilla RAG baseline using BGE-M3 (Chen et al., 2024).

4.4 LLMs for Compression

We select two open-source LLMs for evaluation, Llama-3.1-8B-Instruct (Dubey et al., 2024) and Qwen-2.5-7B-Instruct (Yang et al., 2025). We experiment with both 8B and 70B versions of Llama-3.1-Instruct as our compression model, denoted as ATTENTIONRAG (8B) and ATTENTIONRAG (70B) respectively. Detailed hyperparameter configurations are provided in the Appendix §B.4.

5 Results

5.1 Overall Results

As the results in Tables 1 and 2 show, ATTENTIONRAG outperforms nearly all baseline methods across the benchmarks, consistently achieving superior results. On the LongBench benchmark, which includes contexts of tens of thousands of tokens, ATTENTIONRAG outperforms the LLMLingua methods in most metrics, particularly in BABILong 1k, where it achieves an 18% higher

Model	Method	2WikiMQA			HotpotQA			TriviaQA			Gov	
		EM↑	LLM Judge↑	CR↑	EM↑	LLM Judge↑	CR↑	EM↑	LLM Judge↑	CR↑	Rouge↑	CR↑
Llama-3.1-8B-Instruct	Original Prompt	0.49	0.46	1.0x	0.50	0.62	1.0x	0.87	0.80	1.0x	0.32	1.0x
	Random	0.21	0.15	5.0x	0.05	0.01	3.3x	0.76	0.73	1.7x	–	–
	LLMLingua2 (small)	0.24	0.26	5.0x	0.40	0.45	3.3x	0.86	**0.82**	1.7x	0.23	3.3x
	LLMLingua2 (large)	0.33	0.30	5.0x	0.44	0.51	3.3x	0.84	0.76	1.7x	0.24	3.3x
	LongLLMLingua	0.23	0.22	3.3x	0.33	0.40	3.3x	0.83	0.72	1.7x	0.23	3.1x
	BGE-M3	0.40	0.35	3.5x	0.42	0.50	3.0x	0.80	0.77	1.9x	0.21	3.0x
	AttentionRAG (small)	0.39	0.36	6.3x	0.45	0.55	3.7x	0.84	0.77	2.0x	**0.28**	3.4x
	AttentionRAG (large)	**0.42**	**0.38**	15x	**0.48**	**0.61**	5.6x	**0.89**	0.81	1.7x	**0.28**	4.2x
Qwen2.5-7B-Instruct	Original Prompt	0.56	0.46	1.0x	0.58	0.68	1.0x	0.94	0.82	1.0x	0.31	1.0x
	Random	0.18	0.06	5.0x	0.04	0.01	3.3x	0.65	0.51	1.7x	–	–
	LLMLingua2 (small)	0.29	0.27	5.0x	0.43	0.47	3.3x	**0.91**	0.80	1.7x	0.21	3.3x
	LLMLingua2 (large)	0.40	0.28	5.0x	0.48	**0.58**	3.3x	0.87	**0.81**	1.7x	0.22	3.3x
	LongLLMLingua	0.31	0.18	4.0x	0.28	0.31	3.3x	0.83	0.73	1.7x	0.23	3.1x
	BGE-M3	0.40	0.35	3.5x	0.42	0.50	3.0x	0.80	0.77	1.9x	0.22	3.0x
	AttentionRAG (small)	**0.41**	0.28	6.3x	0.51	0.54	3.7x	0.83	0.73	2.0x	**0.28**	3.4x
	AttentionRAG (large)	**0.41**	**0.30**	15x	**0.53**	0.56	5.6x	0.88	0.80	1.7x	**0.28**	4.2x

Table 1: Performance comparison on QA datasets (2WikiMQA, HotpotQA, TriviaQA) and the Gov summarization dataset.

Model	Method	1K			2K			4K		
		EM↑	LLM Judge↑	CR↑	EM↑	LLM Judge↑	CR↑	EM↑	LLM Judge↑	CR↑
Llama-3.1-8B-Instruct	Original Prompt	0.74	0.64	1.0x	0.67	0.57	1.0x	0.71	0.60	1.0x
	Random	0.11	0.04	2.9x	0.09	0.05	3.3x	0.05	0.04	3.3x
	LLMLingua2 (small)	0.55	0.41	2.9x	0.38	0.29	3.3x	0.36	0.29	3.3x
	LLMLingua2 (large)	0.69	0.51	2.9x	0.53	0.41	3.3x	0.49	0.37	3.3x
	LongLLMLingua	0.63	0.59	2.9x	0.43	0.38	3.3x	0.47	0.36	3.3x
	AttentionRAG (small)	**0.74**	**0.65**	3.8x	**0.58**	**0.49**	3.8x	**0.53**	**0.41**	5.6x
Qwen2.5-7B-Instruct	Original Prompt	0.87	0.78	1.0x	0.75	0.71	1.0x	0.80	0.75	1.0x
	Random	0.04	0.04	2.9x	0.05	0.03	3.3x	0.04	0.01	3.3x
	LLMLingua2 (small)	0.49	0.38	2.9x	0.37	0.30	3.3x	0.34	0.24	3.3x
	LLMLingua2 (large)	0.70	0.52	2.9x	0.55	0.41	3.3x	0.50	0.38	3.3x
	LongLLMLingua	0.54	0.47	2.9x	0.35	0.29	3.3x	0.37	0.29	3.3x
	AttentionRAG (small)	**0.88**	**0.76**	3.8x	**0.62**	**0.57**	3.8x	**0.66**	**0.59**	5.6x

Table 2: Comparison of different methods on BABILong.

score than the best LLMLingua model. Specifically, in TriviaQA, AttentionRAG outperforms the uncompressed method by 2% in EM score, demonstrating the effectiveness of our approach. However, the low compression ratio is due to two parallel factors: the scattering of relevant information across the context and the use of uniform hyperparameters. All these factors contribute to the lower ratio, which we aim to enhance. We discuss this in more detail in the Ablation Study. Also, our method excels in summarization tasks, showing the stability of our method. Similar results can be observed in the BABILong benchmark, which involves a wider range of RAG context lengths. Our method consistently outperforms all LLMLingua methods while achieving the highest compression ratio, further demonstrating the effectiveness of AttentionRAG in varying context sizes.

5.2 Efficiency Analysis

As discussed in §3, we divide the context into smaller chunks and use attention mechanisms for compression. The overall time cost is spent in the forward pass of the compression model plus the answering of the generation model. We employ LLaMA-3.1-Instruct with int4 quantization for compression. We choose HotpotQA, where the average context length is about tens of thousands tokens. As shown in Table 3, the overall time cost is significantly reduced compared to baseline methods, and the quantized model can also ensure similar performance on results.

5.3 Case Study

Table 4 presents a practical example of AttentionRAG, demonstrating how our method effectively compresses the context while maintaining both accuracy and readability. Unlike the origi-

Method	EM ↑	LLM Judge ↑	CR ↑	Compression Latency (s)	Answering Latency (s)
Original Prompt	0.50	0.62	1.0x	–	19.36
Vanilla RAG (BGE-M3)	0.42	0.50	3.0x	6.11	5.99
LLMLingua2 (small)	0.40	0.45	3.3x	0.44	5.39
LLMLingua2 (large)	0.44	0.51	3.3x	0.88	5.43
LongLLMLingua	0.42	0.52	3.3x	7.12	5.41
AttentionRAG	**0.45**	**0.54**	**4.0x**	**3.99**	**4.94**

Table 3: Time latency for inference by various methods on HotpotQA

nal context, which contains redundant information that can negatively affect the response quality, ATTENTIONRAG generates a concise and coherent output with minimal token usage. Other methods, such as LLMLingua2, while providing a more compact result, produce fragmented and less readable responses that lose coherence and relevance. Similarly, LongLLMLingua, despite reducing the context, fails to provide a clear and focused answer. In contrast, ATTENTIONRAG generates the correct answer, "Ozalj," with the highest compression ratio, illustrating its ability to preserve the essential information. This highlights ATTENTIONRAG 's capacity to enhance overall response quality, effectively balancing compression and clarity without introducing unnecessary complexity.

5.4 Ablation Study

Fixed Hint Prefix As discussed in §3.1, we use a fixed hint prefix for questions that cannot generate one dynamically. The summarization performance under this setting is reported in Table 1. To further validate the robustness of this approach, we conduct experiments on three additional datasets: 2WikiMQA, HotpotQA, and TriviaQA. As shown in Table 5, we experiment with LLaMA-3.1-8B-Instruct, using only a fixed hint prefix results in a slight performance drop compared to our original method, yet still consistently outperforms all baselines. This demonstrates that even a static hint prefix serves as an effective anchor token, reliably guiding the model to identify important information within the context.

Combination with other RAG methods To explore the integration potential of our method with retrieval-based techniques, we implemented a two-stage pipeline: we first use vanilla RAG (BGE-M3) for initial context retrieval, followed by our method for compression. Specifically, we retain the RAG chunk size at 300 tokens, while retrieve the top 15 relevant chunks, and then apply our compression method. We use LLaMA-3.1-8B-Instruct as the generation model and conduct experiments on the HotpotQA, 2WikiMQA, and TriviaQA benchmarks (See Table 6.

Hyperparameters We conduct ablation studies on two hyperparameters: chunk size (§3) and the number of Top-K tokens used to select sentences within each chunk (§3.3). For long contexts (e.g., LongBench), we use larger chunks and higher K to handle dispersed information; for shorter contexts (e.g., Babilong), we use smaller chunks and lower K for finer granularity.[2] Using TriviaQA, we test various configurations and observe (Table 7) that increasing chunk size and reducing K lowers the compression ratio with minimal performance drop. This suggests that dynamic hyperparameter tuning offers a better trade-off between efficiency and accuracy than fixed-ratio compression.

Size of Compression Models To evaluate the model-size sensitivity of our compression approach, we compare the performance of ATTENTIONRAG when using LLaMA-3.1-Instruct 8B versus 70B to compute attention scores. As shown in Table 1, the performance gap is minimal, indicating that even a lightweight model can effectively capture the attention patterns needed for compression. This suggests that ATTENTIONRAG is largely agnostic to model size and can maintain strong performance without relying on large-scale models—offering substantial efficiency benefits. Furthermore, as demonstrated in §5.2, our method can be further accelerated through quantization. Overall, ATTENTIONRAG is highly scalable and future-proof, with the potential to continuously benefit from advances in foundation model development while remaining efficient and adaptable.

6 Related Work

Retrieval-Augmented Generation RAG has shown strong performance in tasks like open-domain QA (Han et al., 2024), but its effectiveness

[2]Details in §B.

Query:	Where was the wife of Francis I Rákóczi born?	**Answer: Ozalj**
Original Context:	Passage 1: Waldrada of Lotharingia Waldrada was the mistress, and later the wife, of Lothair II of Lotharingia. Biography Waldrada's family origin is uncertain. The prolific 19th-century French writer Baron Ernouf suggested that Waldrada was of noble Gallo-Roman descent, sister of Thietgaud... (**7003 tokens**)	city of Gyulafehérvár, Transylvania. ×
Random:	drada,, of. th-century French Baron Er of-R sister ofga bishopther arch of, not any socialoli,... (**1400 tokens**)	The text does mention it. ×
LLMLingua2:	Waldrada Lotharingia mistress Lothair II Gallo sister Thietgaud niece Gunther Vita Sancti related Eberhard II Etichonids 855 Lothar II married Teutberga 858 862 Nicholas 863Charles ...(**632 tokens**)	Munkács.×
LongLLMLingua:	Passage:Waldrada theressairia. is The proific 1th French Baron Ern thatadaoman, sister of Th Trier, Gun of and have suggested of social though anatic.itactoli thatada Ehard II,edbourgichon . ... (**920 tokens**)	Hungary. ×
ATTENTIONRAG:	... Life Early years and family Ilona was born Ilona Zrínyi in Ozalj ... She was the daughter of Petar Zrinski, Ban (viceroy) of Croatia, the niece of both Miklós Zrínyi and Fran Krsto Frankopan and the wife of Francis Rákóczi I ... (**273 tokens**)	Ozalj ✓

Table 4: Examples of compression results by various methods

Dataset	EM↑	LLM Judge↑	CR↑
HotpotQA	0.44	0.53	3.3x
2WikiMQA	0.44	0.37	5.2x
TriviaQA	0.81	0.75	2.0x

Table 5: Performance of ATTENTIONRAG using fixed hint prefix across datasets.

EM	LLM	Judge	CR
HotpotQA	0.46	0.58	9.1x (+5.4x)
2wikimqa	0.43	0.40	7.9x (+1.6x)
triviaqa	0.88	0.83	3.2x (+1.5x)

Table 6: Combination with other RAG method

Chunk Size	Top-K	EM↑	LLM Judge ↑	CR↑
300	5	0.80	0.73	**3.2x**
300	10	0.84	0.75	2.2x
300	15	**0.87**	**0.77**	1.9x
100	10	**0.90**	**0.79**	1.9x
200	10	0.88	0.77	2.2x
400	10	0.85	0.77	**2.4x**

Table 7: Performance on TriviaQA with different chunk sizes and top-K values

is often hindered by noisy or redundant retrieved content (Shi et al., 2023). To address this, recent work has focused on improving retrieval quality. Wang et al. (2023) trains models to filter irrelevant content, while Xu et al. (2023) uses extraction-based compression to retain key information. Unlike these approaches, ATTENTIONRAG does not require additional training. It leverages internal attention signals to identify informative content, offering strong performance and broad transferability across tasks and models.

Prompt Compression To reduce the cost of long-context generation, both soft and hard prompt compression methods have been proposed. Soft prompt methods include Gist (Mu et al., 2024) and 500x Compressor (Li et al., 2024), which compress context into dense tokens with minimal loss. Hard prompt approaches like LLMLingua (Jiang et al., 2023), LongLLMLingua (Jiang et al., 2024), and LLMLingua2 (Pan et al., 2024) use token-level filtering, achieving substantial speedups and compression. In contrast, ATTENTIONRAG enhances LLM performance by selecting explainable, model-attended content without retraining. We compare against LongLLMLingua and LLMLingua2 to demonstrate its efficiency and robustness.

7 Conclusion

In this paper, we propose ATTENTIONRAG, a novel attention-guided context pruning method for RAG systems. The core part of our method is the formatted attention focus mechanism, which constructs an answer hint prefix and utilize a fixed hint prefix in a *next-token-prediction* format for each query, guiding the LLM to pay attention to relevant tokens in the retrieved context through one token. We conduct extensive experiments on the 2WikiMQA, HotpotQA, TriviaQA, GovReport and Babilong benchmarks, demonstrating its strong performance.

Limitation

In this section, we faithfully discuss the current limitations and potential avenues for future research.

Regarding the attention feature computation, we currently aggregate attention scores across all layers. However, we believe this process can be optimized using more sophisticated algorithms to improve efficiency.

Additionally, while we propose a dynamic compression ratio, we have not yet developed methods for explicitly controlling or instructing the desired ratio. Determining and setting precise parameters to achieve a specific compression ratio is a challenging task. In future work, we aim to investigate ways to provide more flexible and accurate control over the compression ratio.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. *Preprint*, arXiv:2308.14508.

Amit Ben-Artzy and Roy Schwartz. 2024. Attend first, consolidate later: On the importance of attention in different llm layers. *Preprint*, arXiv:2409.03621.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

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*.

David Chiang and Peter Cholak. 2022. Overcoming a theoretical limitation of self-attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7654–7664, Dublin, Ireland. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

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

OpenAI et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Xiaodong Gu, Zhaowei Zhang, Sang-Woo Lee, Kang Min Yoo, and Jung-Woo Ha. 2022. Continuous decomposition of granularity for neural paraphrase generation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6369–6378, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Rujun Han, Yuhao Zhang, Peng Qi, Yumo Xu, Jenyuan Wang, Lan Liu, William Yang Wang, Bonan Min, and Vittorio Castelli. 2024. Rag-qa arena: Evaluating domain robustness for long-form retrieval augmented question answering. *Preprint*, arXiv:2407.13998.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

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 Tomas Pfister. 2024. Found in the middle: Calibrating positional attention bias improves long context utilization. *Preprint*, arXiv:2406.16008.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. *Preprint*, arXiv:2212.10403.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

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

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. *Preprint*, arXiv:2310.06839.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, Fan Yang, Mengnan Du, and Yongfeng Zhang. 2025. Exploring concept depth: How large language models acquire knowledge and concept at different layers? *Preprint*, arXiv:2404.07066.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Preprint*, arXiv:2406.10149.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

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

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2024. Learning to compress prompts with gist tokens. *Preprint*, arXiv:2304.08467.

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. *Preprint*, arXiv:2403.12968.

Haoran Que, Feiyu Duan, Liqun He, Yutao Mou, Wangchunshu Zhou, Jiaheng Liu, Wenge Rong, Zekun Moore Wang, Jian Yang, Ge Zhang, Junran Peng, Zhaoxiang Zhang, Songyang Zhang, and Kai Chen. 2024. Hellobench: Evaluating long text generation capabilities of large language models. *Preprint*, arXiv:2409.16191.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *Preprint*, arXiv:1606.05250.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. *Preprint*, arXiv:2302.00093.

Yunxiao Shi, Xing Zi, Zijing Shi, Haimin Zhang, Qiang Wu, and Min Xu. 2024. Enhancing retrieval and managing retrieval: A four-module synergy for improved quality and efficiency in rag systems. *Preprint*, arXiv:2407.10670.

Davoud Ataee Tarzanagh, Yingcong Li, Xuechen Zhang, and Samet Oymak. 2023. Max-margin token selection in attention mechanism. *Preprint*, arXiv:2306.13596.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Sourav Verma. 2024. Contextual compression in retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2409.13385*.

Minzheng Wang, Longze Chen, Fu Cheng, Shengyi Liao, Xinghua Zhang, Bingli Wu, Haiyang Yu, Nan Xu, Lei Zhang, Run Luo, Yunshui Li, Min Yang, Fei Huang, and Yongbin Li. 2024. Leave no document behind: Benchmarking long-context LLMs with extended multi-doc QA. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5627–5646, Miami, Florida, USA. Association for Computational Linguistics.

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

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Yun Zhu, Jia-Chen Gu, Caitlin Sikora, Ho Ko, Yinxiao Liu, Chu-Cheng Lin, Lei Shu, Liangchen Luo, Lei Meng, Bang Liu, and Jindong Chen. 2024. Accelerating inference of retrieval-augmented generation via sparse context selection. *Preprint*, arXiv:2405.16178.

A Pseudocode

The Pseudocode of ATTENTIONRAG is provided in Algorithm 1. The algorithm takes a retrieved long context, query, and two language models as input. First, it generates an answer hint prefix based on the query to guide the attention mechanism. Then, it splits the long context into chunks of size m. For each chunk, it generates an anchor token using the compression model. If the anchor token is valid (not "none"), it computes attention features using the anchor token and compresses the chunk accordingly. Finally, all compressed chunks are concatenated and used with the original query to generate the final answer.

Algorithm 1 Pseudocode of ATTENTIONRAG.

Input: Retrieved long context C, query q, generation model L, compression model L_C
Output: Generated sequence y

Generate Answer Hint Prefix
Get answer hint prefix p through L with q
Chunking
Generate chunks $c_1, \ldots, c_n$ by partitioning C with chunk size m, where $n = \lceil |C|/m \rceil$
Initialize empty variable C'
Compressing with Attention
for $j = 1$ to n **do**
    Generate the anchor token a_1 with L_C, c_j, q, and p
    if a_1 is "none" **then**
        continue
    else
        Obtain Attention Features A_1 with a_1 and c_j ▷ Eq. (2)
        Get compressed c'_j according to A_1 and c_j ▷ Eq. (3)
        Append c'_j to C'
    end if
end for
Generate y from L with C' and q
Return Generated sequence y

B Implementation Details

B.1 Prompt for generating answer prefix hint

We use the following prompt for generating answer prefix hint according to each query.

> You are a formatting assistant. Given a question, your task is to generate a corresponding answering format. The format should maintain the same structure as the question but transform it into an incomplete answer template. If it is impossible to generate a format, return "None".
>
> The format is like an complete answer, but truncated before the key word, and the key word is not included in the format.
>
> For instance, if the question is "Where is Daniel?", the format should be "Daniel is in the", as the next word is the key word.
>
> Note: For yes/no questions, such as "Is Tom here?", return "None" because these questions are typically answered with "yes" or "no" and do not have a natural continuation that leads to a single keyword.
>
> Examples:
> 1. Question: Where is Daniel?
> Format: Daniel is in the
>
> 2. Question: What time is it?
> Format: It is
>
> 3. Question: Who is responsible for this?
> Format: The person responsible for this is
>
> 4. Question: Which film was released more recently, Dance With A Stranger or Miley Naa Miley Hum?
> Format: The film released more recently was
>
> 5. Question: Is Tom here?
> Format: None
>
> In generation , you should only return the format, not any other text.
> Now, here's a new question:
> Question: question
> Format:

B.2 Fixed Hint Prefix

Our fixed hint prefix is: *"Please output the most relevant keyword or phrase that is relevant to the answer of the question."*

B.3 Prompt for generating anchor token

We use the following prompt to generate the anchor token for computing attention featrues.

> You will be given a long context begin with 'Context:', a question begin with 'Question:', and a hint begin with 'Hint:'. Please answer the question.
> Context: {chunk}
> Hint: You should answer begin with {prefix_hint}, if there is no useful information in the context for the question in the context and you really don't know the answer, just answer prefix_hint none.
> Question: {question}
> Answer:
> {prefix_hint}

B.4 Hyperparameter settings

To reduce the randomness, we use greedy decoding in open-source LLMs generation. For the chunk size and K in the attention-based compression process, we set them according to the context length in different benchmarks. In LongBench, where contexts are quite long, we use larger chunk size and K, in contrast, in BABILong, where we choose to experiment with mid-sized context, we use smaller chunk size and K. The detailed setting is shown in Table 8.

Dataset	**Chunk_Size**	K
HotpotQA	300	12
2WikiMQA	300	15
TriviaQA	150	8
BABILong 1k	50	8
BABILong 2k	100	10
BABILong 4k	200	12

Table 8: Hyperparameter settings of the experiment

B.5 Attention Layer Choice

In LLMs, attention layers capture different levels of information—shallow layers focus on syntax, while deeper ones encode semantics (Ben-Artzy and Schwartz, 2024; Jin et al., 2025). To fully exploit this hierarchy, we aggregate attention scores across all layers for compression, as detailed in §3.2. This mitigates layer-specific bias and captures a broader information spectrum. We compare this approach with using shallow, middle, or deep layers alone on HotpotQA with Llama-3.1-8B-Instruct.

Layer Subsets	EM↑	LLM Judge ↑	CR↑
0 - 10	0.35	0.43	**4.5x**
11 - 20	0.38	0.50	3.6x
21 - 31	0.40	0.48	3.7x
0 - 31	**0.42**	**0.54**	3.6x

Table 9: Performance on HotpotQA with different subset of layers

As shown in Table 9, full-layer aggregation yields superior performance, validating our strategy.