Efficient Encoder-Only Context Compression via Marginal Contribution Scoring

Thao Do [1] **Dinh Phu Tran** [1] **An Vo** [1] **Seon Kwon Kim** [1] **Daeyoung Kim** [1]

Abstract

Efficient context compression is critical for retrieval-augmented question answering in resource-constrained settings, where long retrieved contexts increase latency, memory use, and LLM reader cost. We propose a lightweight encoder-only framework for query-driven sentence pruning that preserves answer-critical evidence while aggressively reducing irrelevant context. Our method learns marginal contribution scores for sentences using counterfactual training signals, and optimizes a contrastive ranking objective that separates critical evidence from noncritical context. Unlike decoder-heavy compressors, our approach scores all sentences from a single full-context encoding, enabling fast inference with low computational overhead. Experiments show that it maintains accuracy comparable to the strongest baseline while using 3.7× less peak memory and substantially lower compression latency, making it suitable for practical resource-constrained deployment.

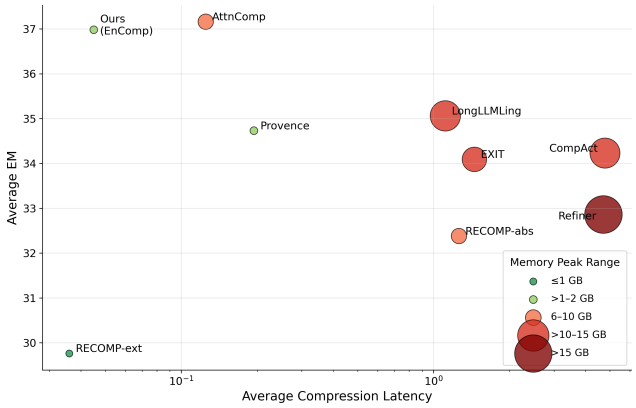

Figure 1. Quality–latency–memory trade-off across five datasets using Llama-3.3-70B-Instruct as the reader at retrieval depth $k = 20$. EM denotes average exact-match answering accuracy, while bubble size indicates the peak VRAM required. Our method achieves competitive EM while requiring 3.7× less VRAM and nearly 3× lower compression latency than the strongest baseline.

1. Introduction

Retrieval-augmented generation (RAG) has emerged as a powerful paradigm for enhancing large language models (LLMs) with external knowledge, improving factual accuracy and reducing hallucinations (Lewis et al., 2020; Ram et al., 2023). However, retrieving more documents improves coverage yet expands context length, increasing memory consumption and inference latency (Lu et al., 2025), while also introducing extraneous content that can distract the model and degrade generation quality (Liu et al., 2023).

Context compression has emerged as a potential solution to this challenge, enabling RAG systems to retain essential information while reducing computation (Li et al., 2023; Wingate et al., 2022). Existing approaches broadly fall into two categories: abstractive methods that generate condensed summaries (Yoon et al., 2024; Xu et al., 2024; Li et al., 2024) and extractive methods that select relevant text segments (Jiang et al., 2023). Abstractive methods achieve high compactness by rewriting content into concise summaries, but their token-by-token generation process incurs additional latency, which can exceed the time saved from reduced context length. Extractive methods generally offer faster compression because they avoid generation, but many rely on rigid selection criteria that fail to adapt to varying query requirements or preserve evidence distributed across multiple sentences.

Recent approaches have sought to mitigate some of these limitations. EXIT (Hwang et al., 2025) and AttnComp (Luo et al., 2025) use context-aware extractive compression that leverages full-document context to make sentence-level or chunk-level decisions in parallel, enabling more efficient filtering over long contexts. However, they still rely on decoder-based LLMs for high filtering quality, creating computational overhead and high memory requirements for a sentence selection task. Meanwhile, comparative studies show that encoder-only models can match or surpass decoder-only models on various natural language understanding (NLU) tasks (Benayas et al., 2025; Nielsen et al.,

[1]School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea. Correspondence to: Thao Do <thaodo@kaist.ac.kr>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

2024). In a related direction, Provence (Chirkova et al., 2025) pursues efficient pruning with token-level supervision. However, using sentence relevance labels as supervision for every token, including non-informative words, can introduce noise since many tokens in an important sentence (e.g., function words) are not themselves informative. Its sentence rounding heuristic further makes sentence selection depend on aggregated token predictions rather than direct sentence-level scoring.

In this work, we revisit the fundamental design choices in query-driven context pruning to address the trade-off among quality, latency, and memory requirements. Our key premise is that compression should not treat each sentence as an isolated relevance label, but should estimate how much the sentence contributes to the answerability of the full passage. We therefore propose a lightweight encoder-only compressor that learns sentence-level contribution scores under full-passage context. During training, we estimate these contributions by counterfactually removing sentences and measure the resulting change in passage answerability. These counterfactual redactions provide soft supervision for these contributions, while a contrastive ranking objective encourages answer-critical evidence to outrank distracting context. At inference time, the model scores all sentences from a single full-context encoding, avoiding the computational burden of decoder-heavy compression pipelines. Our main contributions are as follows:

1. We introduce **EnComp**, a lightweight encoder-only framework for query-driven context compression. Its main novelty is to model sentence selection as marginal contribution scoring for passage answerability where EnComp learns these scores from counterfactual redaction signals and contrastive ranking, while requiring only a single full-context encoding at inference time.
2. We demonstrate the effectiveness and efficiency of EnComp through comprehensive experiments on five standard QA benchmarks with both open-source and proprietary LLM readers. Compared with recent strong baselines, EnComp preserves competitive answer fidelity and compression quality while substantially reducing compression latency and GPU memory usage.

2. Related Work

2.1. Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Gao et al., 2023) improves LLMs by grounding generation in external knowledge sources. Recent iterative, multi-hop, and recursive retrieval methods (Khattab et al., 2022; Trivedi et al., 2022a) refine queries or retrieve additional documents during generation to access dispersed evidence, but often increase latency and token usage while introducing irrelevant details that can reduce precision (Liu et al., 2023). This has motivated token-reduction and document-compression techniques that preserve essential evidence while supporting efficient, high-quality generation.

2.2. Query-aware Context Compression in RAG

Context compression in RAG can be divided into soft and hard compression. Soft compression shortens embedding-level representations (Mu et al., 2023; Ge et al., 2023), offering controllable rates but often requiring fine-tuning or architectural changes. Hard compression filters or rewrites natural-language text. Abstractive methods summarize retrieved passages with generative models (Xu et al., 2024; Yoon et al., 2024; Li et al., 2024), achieving high token reduction but often increasing resource use and latency.

Extractive methods, such as RECOMP-Extr (Xu et al., 2024) and the LLMLingua family (Jiang et al., 2023; Pan et al., 2024), select salient sentences or tokens from retrieved documents, preserving original evidence and reducing hallucination risk. However, query-agnostic selection can disrupt coherence or omit critical information. LongLLMLingua adds query awareness and configurable compression rates, but balancing coverage and efficiency remains challenging. EXIT (Hwang et al., 2025) prunes context under the full-context condition, yet its architecture introduces high compression cost, and binary thresholding yields modest compression gains. AttnComp (Luo et al., 2025) uses a sizeable decoder-only LLM as an encoder and gates chunks with cumulative attention weights during prefill, but still incurs memory overhead and potentially produces unstable compression rates due to coarse chunk granularity.

Meanwhile, Provence (Chirkova et al., 2025) improves efficiency through token-level pruning, but token-centric supervision is only an indirect proxy for sentence-level salience. Aggregating token predictions with heuristics may miss discourse- and structure-level cues needed for coherent, utility-driven extraction.

To balance quality, latency, and memory efficiency in hard compression, we adopt extractive sentence selection with a lightweight encoder-only architecture. Our method directly estimates each sentence's clue contribution under full-passage context. This design preserves answer-critical evidence from the retrieved text while keeping compression fast and memory-light.

3. Methodology

Fig. 2 gives the overall inference flow of our compressor between the retriever and the reader. Given the user question and the retrieved chunks, EnComp compresses each chunk independently while conditioning on the question. For each chunk, a single encoder-only pass scores both the chunk and

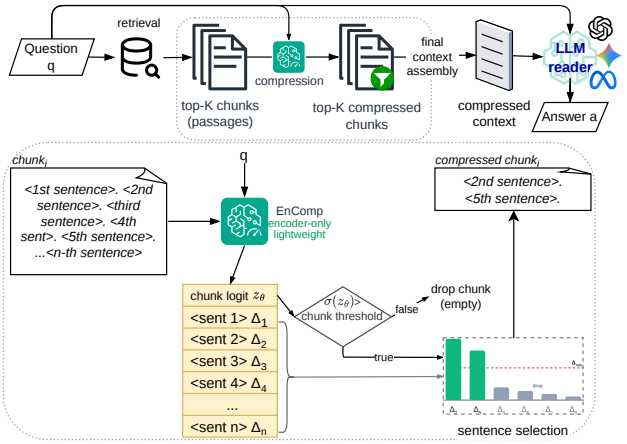

Figure 2. Overview of our framework. Our proposed lightweight context pruner includes three steps. (1) Model predicts the changes in clue richness, denoted as Δ, assuming the sentence is omitted. A larger Δ indicates that the sentence is more critical. (2) Apply thresholdings at chunk level and sentence's deltas to retain most essential sentences. (3) Compressed chunks are joined in retrieved order to create final context.

all of its sentences; these scores support the two-stage filtering shown in the figure, where chunks with little evidence are discarded and evidence-bearing sentences are retained from the remaining chunks. The retained sentences preserve their original order within each chunk, and the compressed chunks are then concatenated in retriever order to form the final context for the reader. We next formalize this compression setting, then describe amortized marginal contribution scoring, the training objective, and inference procedure.

Problem Formulation Given a query q and top-k retrieved passages $D_k = \langle d_1, \ldots, d_k \rangle$, a reader model M generates an answer y from the provided context. We aim to learn a compression function π that produces a compact query-relevant context:

$$D'_k = \pi(q, D_k), \qquad \ell(D'_k) \ll \ell(D_k),$$

where $\ell(\cdot)$ denotes context length, measured in tokens. The compressed context should preserve the evidence needed for M to answer q.

RAG Pipeline with Compression Let $\mathcal{K}$ denote the external document corpus. For a query q, the retriever returns the top-k passages $D_k = \langle d_1, \ldots, d_k \rangle = \text{Retriever}(q, \mathcal{K})$, which are then compressed as:

$$D'_k = \text{Compressor}(q, D_k), \quad \text{s.t. } \ell(D'_k) \ll \ell(D_k).$$

The language model generates the final answer using D'_k as context. Effective compression should retain evidence, remain lightweight, and substantially reduce token usage.

3.1. Amortized Marginal Contribution Scoring

For a passage $P = \langle s_1, \ldots, s_n \rangle$, spaCy (Honnibal et al., 2020) provides the sentence boundaries, which are mapped to the encoded token sequence through a sentence-token mask indicating which tokens belong to each sentence. Contextual token representations within each sentence span are pooled into a sentence vector, giving the scoring head one aligned representation for each sentence.

For a query–passage pair (q, P), our encoder-only pruner produces contextual representations for the full query–passage input. A pooling head summarizes the sequence into a global passage vector g, and a document head predicts an answerability logit:

$$z_\theta(q, P) = f_{\text{doc}}(g).$$

For each sentence $s_i \in P$, the model obtains a contextual sentence vector e_i from the same full-passage encoding and predicts a contribution score:

$$\hat{\Delta}_i = f_{\text{sent}}(e_i, g).$$

The score $\hat{\Delta}_i$ is not an isolated relevance label; it estimates the marginal contribution of s_i, i.e., how much removing that sentence would change passage answerability. A high positive score therefore marks evidence whose removal would weaken the passage for answering q. This scoring is amortized: all sentence contributions are predicted from one full-context encoding, avoiding explicit per-sentence counterfactual re-encoding at inference time.

3.2. Training Objective

Each sentence has a binary label $y_i \in \{0, 1\}$, where 1 indicates a critical sentence. The passage-level target is

$$y_{\text{doc}} = \mathbb{I}\left[\exists i : y_i = 1\right].$$

The training signal follows the same marginal-contribution view: critical sentences should cause a larger answerability drop when removed than noncritical sentences. The model should therefore assign higher contribution scores to answer-critical evidence and separate them from distractors within the same passage.

Counterfactual Contribution Targets During training only, each sentence is paired with a counterfactual passage, i.e., the same passage with that sentence redacted. This gives a soft contribution target:

$$\Delta_i^\star = \text{sg}\left(z_\theta(q, P) - z_\theta(q, P \setminus \{s_i\})\right),$$

where $\text{sg}(\cdot)$ denotes stop-gradient. This target captures the answerability drop caused by removing s_i. The sentence head is trained to approximate this counterfactual effect from the full-context encoding, so inference does not require evaluating counterfactual passages.

Marginal-Contrastive Objective Let $C = \{i : y_i = 1\}$ and $N = \{j : y_j = 0\}$. The main objective combines soft matching to marginal contribution targets with contrastive separation:

$$\mathcal{L}_{\text{mc}} = \lambda_{\text{dist}}\mathcal{L}_{\text{dist}} + \lambda_{\text{rank}}\mathcal{L}_{\text{rank}},$$

$$\mathcal{L}_{\text{dist}} = \frac{1}{n}\sum_{i=1}^{n} \text{SmoothL1}(\hat{\Delta}_i, \Delta_i^{\star}),$$

$$\mathcal{L}_{\text{rank}} = \frac{1}{|C||N|}\sum_{i\in C}\sum_{j\in N} \max\left(0, \mu - \hat{\Delta}_i + \hat{\Delta}_j\right).$$

The $\mathcal{L}_{\text{dist}}$ calibrates each predicted contribution score against the observed answerability drop caused by removing that sentence, while the $\mathcal{L}_{\text{rank}}$ forces critical sentences to outrank noncritical ones in the same context. We set $\mathcal{L}_{\text{rank}} = 0$ when either C or N is empty. This relative objective is more informative than independent binary classification because it directly models the competition between useful evidence and distracting context.

We further include a document-level BCE term $\mathcal{L}_{\text{doc}}$ on $z_\theta(q, P)$ to calibrate whether the passage contains any useful evidence. A lightweight zero-anchor regularizer $\mathcal{L}_{\text{zero}}$ (detailed in Appendix A.1) keeps noncritical sentence scores, and all sentence scores in clue-free passages, close to zero. The complete loss is:

$$\mathcal{L} = \mathcal{L}_{\text{mc}} + \lambda_{\text{doc}}\mathcal{L}_{\text{doc}} + \lambda_{\text{zero}}\mathcal{L}_{\text{zero}}.$$

3.3. Inference

At inference time, the compressor segments each passage into sentences and runs the model once for the full query–passage pair. The document logit first acts as a coarse gate: if $\sigma(z_\theta(q, P))$ is below a tunable d_{min} threshold, the passage is treated as non-informative. Otherwise, sentences are ranked by their predicted contribution scores $\hat{\Delta}_i$, and positive-scoring sentences above a tunable δ_{min} threshold are retained under a small retention budget. The selected sentences are returned in their original order to form the compressed context.

4. Experiment

4.1. Experimental Setup

Training Phase We use a random subset 30% out of 90447 questions from the *original* HotpotQA (Yang et al., 2018) train set to train with a 9:1 split, where there are passages (and annotated critical sentences) for each question. We merged very short passages into longer ones to mitigate the dataset's severe passage-length imbalance, shorten training time, and enable additional data augmentation. Training is performed on a single RTX 4090 GPU using LoRA (Hu et al., 2022) to finetune the backbone with the standard separator token (e.g., [SEP]) automatically defined.

RAG Configuration The main pipeline consists of 3 primary components. The Contriever-MSMARCO (Izacard et al., 2021) was used as chunk retriever on the *2018 Wikipedia corpus* dataset (Karpukhin et al., 2020) to create passages for each question from Question Answering (QA) datasets. Our compressor is based on an encoder-only model – ModernBERT (Warner et al., 2024), large version (395M) used as default, base (139M) used in ablation.

For reading to answer, we mainly employed Llama-3.1-8B-Instruct, Llama-3.3-70B-Instruct (Dubey et al., 2024); and with proprietary Google Gemini-3-flash (Google, 2025), OpenAI GPT-5-mini for further exploration. Vertex AI, OpenRouter, and OpenAI API services were used to provide the mentioned reader APIs.

To evaluate, we use HotpotQA (HQA) (Yang et al., 2018), 2WikiMultihopQA (2WIKI) (Ho et al., 2020), and Musique (Trivedi et al., 2022b) datasets as multi-hop QA benchmarks; and Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (TQA) (Joshi et al., 2017) datasets as single-hop QA benchmarks. The evaluated sets are the same as (Yoon et al., 2024; Hwang et al., 2025).

Baselines We compare our method with eight context compression baselines: **RECOMP-Abs/RECOMP-Ext** (Xu et al., 2024), abstractive and extractive variants; **CompAct** (Yoon et al., 2024), an iterative Mistral-7B compressor; **Refiner** (Li et al., 2024), based on Llama2-7B; **LongLLMLingua** (Jiang et al., 2023) (LongLLMLin), based on Llama2-7B; **EXIT** (Hwang et al., 2025), based on Gemma-2B; **Provence** (Chirkova et al., 2025), based on DeBERTaV3; and **AttnComp** (Luo et al., 2025), based on Llama-3.1-8B Instruct. For a fair comparison, all baselines use the same preprocessing pipeline, FlashAttention-2 when available (Dao, 2024), and the same thread count. We also include an uncompressed *Raw* baseline in the main results.

Evaluation Metrics Exact Match (EM↑) and F1↑ score are used to measure accuracy in question answering. Compression latency (Lat↓, in seconds), compression ratio (Rate↓) are used to evaluate compression speed and compactness. The latency here is the compressed time measured in an end-to-end manner to simulate a realistic processing timeline. The compression rate is defined as the ratio of the length of the compressed sequence to the length of the original sequence using $cl100k\_base$ tokenizer. Each experiment is run 3 times (different batch sizes) on a single RTX 4090 GPU and the average is reported.

4.2. Main Results

Tab. 1 shows evaluation results across five datasets using two commonly used readers at $k = 20$. Note that the compression latency and rate depend on the top-k value and are

Table 1. Performance across models and datasets by readers Llama-3.1-8B Instruct and Llama-3.3-70B Instruct with $k = 20$. Best (**bold**) and second-best (underlined) are determined using higher-precision values before rounding. Compression Latency (Lat↓) is in seconds, and Compression Ratio (Rate↓) is reported as a percentage.

	NQ				TQA				HQA				2Wiki				Musique			
	EM	F1	Lat	Rate	EM	F1	Lat	Rate	EM	F1	Lat	Rate	EM	F1	Lat	Rate	EM	F1	Lat	Rate
Llama-3.1-8B Instruct																				*Top-20 retrieved chunks*
Raw	37.2	49.3	0.000	100.0	64.0	72.7	0.000	100.0	31.1	41.7	0.000	100.0	25.6	32.0	0.000	100.0	4.6	11.1	0.000	100.0
CompAct	33.7	45.4	4.312	3.7	60.0	69.0	4.102	3.5	30.9	42.1	4.696	3.7	21.6	28.2	4.905	3.5	5.3	12.4	5.869	4.1
EXIT	33.9	45.5	1.392	41.9	60.7	69.8	1.434	36.9	29.0	39.4	1.465	34.0	22.5	28.5	1.492	29.4	4.1	10.8	1.473	33.4
RECOMP-abs	32.9	44.9	1.223	**2.1**	55.9	65.3	1.332	**2.0**	28.5	39.8	1.206	**1.8**	22.4	28.2	1.145	**1.4**	5.2	11.9	1.393	**1.6**
RECOMP-ext	30.9	41.3	**0.036**	11.4	54.8	62.6	**0.037**	11.5	23.3	32.1	**0.035**	12.4	15.8	22.3	**0.036**	12.2	4.0	9.7	**0.037**	12.0
Refiner	31.0	42.0	6.727	8.8	59.1	68.0	5.013	6.5	29.1	39.3	3.915	4.4	22.2	28.1	3.342	3.7	5.1	11.6	4.551	5.0
LongLLMLin	31.7	43.8	1.098	30.1	62.6	71.2	1.108	30.0	30.8	41.3	1.104	30.0	24.0	29.9	1.133	30.2	4.9	11.3	1.120	30.3
Provence	34.2	46.2	0.188	20.8	62.3	71.1	0.189	18.9	30.0	40.5	0.199	13.8	22.5	28.6	0.197	11.9	5.3	11.9	0.196	12.2
AttnComp	34.1	45.9	0.123	10.2	**62.9**	**72.0**	0.124	13.5	32.0	42.8	0.124	11.0	22.6	28.8	0.130	7.0	5.5	11.8	0.126	4.8
Ours	**36.2**	**48.6**	0.044	20.3	62.6	71.8	0.045	18.5	**33.2**	**44.4**	0.045	13.8	**26.2**	**32.4**	0.046	11.5	**5.9**	**13.7**	0.045	15.9
Llama-3.3-70B Instruct																				*Top-20 retrieved chunks*
Raw	39.1	53.4	0.000	100.0	68.8	77.9	0.000	100.0	38.2	50.3	0.000	100.0	30.6	38.5	0.000	100.0	10.0	18.9	0.000	100.0
CompAct	35.2	48.5	4.312	3.7	64.9	74.0	4.102	3.5	36.1	47.8	4.696	3.7	26.5	33.2	4.905	3.5	8.4	15.9	5.869	4.1
EXIT	36.0	49.9	1.392	41.9	65.8	75.3	1.434	36.9	35.0	46.6	1.465	34.0	25.7	32.9	1.492	29.4	8.0	17.0	1.473	33.4
RECOMP-abs	34.2	47.2	1.223	**2.1**	60.2	69.9	1.332	**2.0**	33.1	45.4	1.206	**1.8**	27.7	33.4	1.145	**1.4**	6.7	14.8	1.393	**1.6**
RECOMP-ext	33.5	45.7	**0.036**	11.4	64.0	72.6	**0.037**	11.5	28.3	38.9	**0.035**	12.4	17.1	24.6	**0.036**	12.2	6.0	14.1	**0.037**	12.0
Refiner	33.4	46.4	6.727	8.8	64.1	73.6	5.013	6.5	33.4	45.0	3.915	4.4	25.7	32.0	3.342	3.7	7.8	16.1	4.551	5.0
LongLLMLin	34.5	48.8	1.098	30.1	67.3	76.6	1.108	30.0	36.3	48.7	1.104	30.0	28.7	36.2	1.133	30.2	8.5	17.5	1.120	30.3
Provence	37.0	51.3	0.188	20.8	66.6	76.0	0.189	18.9	35.4	47.5	0.199	13.8	26.1	32.8	0.197	11.9	8.5	17.8	0.196	12.2
AttnComp	37.6	51.3	0.123	10.2	**68.1**	**77.7**	0.124	13.5	37.9	50.6	0.124	11.0	**32.7**	**39.5**	0.130	7.0	9.6	19.0	0.126	4.8
Ours	**37.7**	**52.2**	0.044	20.3	67.0	76.8	0.045	18.5	**38.1**	**51.0**	0.045	13.8	32.2	38.4	0.046	11.5	**9.9**	**19.2**	0.045	15.9

the same regardless of the reader. Although trained only on HQA questions with the original HQA context corpus, our pruner generalizes well across datasets, covering both single-hop and multi-hop settings. Compared with other compressors, our method mostly yields the best or second-best EM/F1 metrics across datasets and readers. It also achieves approximately equal answer quality to, or surpasses, the *Raw* baseline in several cases. Regarding compression efficiency, our method remains highly responsive–achieving the second-best latency among compressors with less than 0.05s and retains only 11–20% of the original context.

Meanwhile, existing methods often struggle to maintain a good trade-off across answer quality, compression rate, and latency. For example, RECOMP-ext is the fastest compressor but usually yields much lower answer quality. Similarly, CompAct or RECOMP-abs compresses context to a very compact level, but at the cost of slow execution. A strong baseline such as AttnComp achieves comparable EM/F1 but is still $\sim 2.75\times$ slower than ours.

For an overall view, we aggregate main experimental results into a single metric vector per compressor as in Tab. 2. For each metric, we take unweighted means in 2 steps: (i) average across the 4 experimental settings – 2 readers $\times$ 2 retrieval depths $k = 5, 20$; then (ii) average across the 5 datasets. In general, we achieve among the best results on answer accuracy while holding the second-fastest position and a decent compactness ratio. Results at $k = 5$ retrieval are shown in Tab. 5 in the Appendix.

Table 2. Overall performance across compressors, via 2-stage averaging: first over 4 settings of 2 readers $\times$ 2 retrieved depths ($k = 5, 20$), then over all five datasets.

	EM	F1	Time	Rate
CompAct	32.2	41.5	3.670	7.7
EXIT	31.2	40.5	0.896	37.2
RECOMP-abs	30.4	39.7	0.910	**4.3**
RECOMP-ext	29.1	38.0	**0.023**	30.4
Refiner	31.6	40.9	3.230	8.5
LongLLMLin	31.5	40.8	0.682	33.5
Provence	32.3	41.9	0.126	17.9
AttnComp	33.8	43.4	0.080	16.5
Ours	**34.3**	**44.1**	0.029	19.2

In Tab. 6 in the Appendix, we conducted experiments with proprietary Gemini-3-flash and GPT-5-mini at a low thinking budget as readers on HQA (intra-domain) and 2Wiki (inter-domain) at retrieval depth $k = 20$ for further comparison. Results show a similar pattern and confirm the efficiency: we mostly achieve the best or second-best answer quality with the fastest latency in the table.

Robustness To assess robustness as the retrieved set enlarges, we vary the retrieval depth $k \in \{5, 10, 20, 30\}$ and report the results in Fig. 3. Our method shows stable gains as k increases, with EM improving from 31.47 at $k = 5$ to 33.49 at $k = 30$. It consistently achieves competitive EM across all retrieval depths, while some baselines exhibit unstable or degraded performance when more retrieved

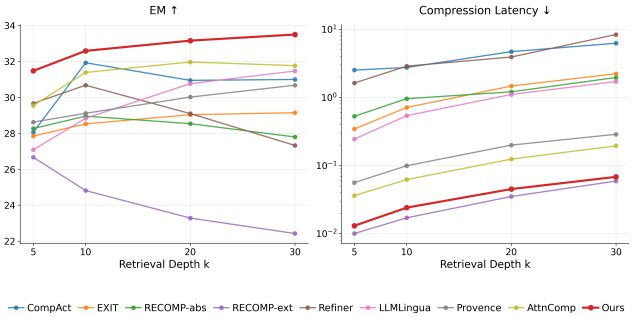

Figure 3. Analysis on HQA at increasing top-k $= \{5, 10, 20, 30\}$ by reader Llama-3.1-8B Instruct, comparing EM, and compression latency between baselines and our proposed method.

chunks are introduced. In terms of efficiency, our method maintains low compression latency, increasing moderately from 0.013s to 0.068s as k grows. Although RECOMP-ext is slightly faster, its EM drops substantially at larger k, whereas our method preserves both high accuracy and low latency. These results suggest that our approach scales robustly with retrieval depth while maintaining a favorable accuracy–efficiency trade-off.

To visualize model comparisons at $k = 20$, Fig. 1 summarizes the quality–latency–memory trade-off across five datasets using the Llama-3.3-70B-Instruct reader. Our method achieves comparable EM to the best while requiring only 1.86 GB peak VRAM and 0.045s compression latency, making it substantially more efficient than the strongest accuracy baseline, which requires 3.7× more VRAM and nearly 3× higher latency. This efficiency is particularly important for resource-constrained deployment, where edge-class devices often operate under tight few-GB shared-memory budgets. These results show that our method preserves high answering accuracy while remaining practical for memory- and latency-sensitive scenarios.

5. Ablation Study

To assess the training objective, we compare the default model with a variant trained using direct binary classification at each sentence head. As shown in Tab. 3, the default model achieves the better or tied performance on most metrics across the five datasets. The binary-loss variant (the 2nd row) remains competitive, with identical latency and a slightly more compact output, but generally performs worse, except on HQA where the results are nearly identical. This suggests that the proposed loss $\mathcal{L}$ provides a better generalization.

We also study backbone capacity by replacing ModernBERT-large with ModernBERT-base one while keeping $\mathcal{L}$ unchanged. The base variant (the 3rd row in Tab. 3) performs lower across all reported metrics, indicating the benefit of the larger backbone. However, it is approximately 30% faster in our measurements, although with a slightly less compact output, showing a clear trade-off between quality and speed.

Table 3. Performance across 5 datasets on our model trained with the direct binary classification (at each sentence head), and a small variant using ModernBERT-base backbone (still with proposed loss $\mathcal{L}$) by Llama-3.1-8B Instruct at $k = 20$. *default* is our one uses -large backbone with loss $\mathcal{L}$.

Variants	NQ		TQA		HQA		2Wiki		Musique	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
default	36.2	48.6	62.6	71.8	33.2	44.4	26.2	32.4	5.9	13.7
-large & $\mathcal{L}_{bin}$	35.3	47.9	62.2	70.6	33.2	44.5	25.6	31.7	5.3	12.3
-base & $\mathcal{L}$	33.8	46.3	61.3	70.7	31.9	43.0	25.5	31.5	5.1	12.4

With a different retriever We evaluate compressors on HotpotQA using BM25 retriever. As shown in Tab. 4, our method achieves leading EM and F1 with low inference time. RECOMP-ext is fastest but much less accurate, while RECOMP-abs has the lowest compression rate but higher latency and lower accuracy.

Table 4. Performance across compressors using Llama-3.1-8B Instruct as the reader at $k = 10$ with the BM25 retriever on a subset of 500 HotpotQA examples.

	EM	F1	Time	Rate
CompAct	31.8	42.9	2.787	6.5
EXIT	29.8	40.8	0.697	36.6
RECOMP-abs	29.9	42.4	0.698	**3.6**
RECOMP-ext	28.2	37.3	**0.016**	27.3
Refiner	32.4	43.3	2.750	8.3
LongLLMLin	30.4	40.7	0.520	31.3
Provence	30.0	41.0	0.099	19.0
AttnComp	33.6	44.7	0.061	21.1
Ours	**34.2**	**46.2**	0.023	15.4

6. Conclusion

We introduced **EnComp**, a lightweight encoder-only framework for query-driven context compression in RAG. EnComp scores sentences from a single full-context encoding and trains the scorer with marginal contribution signals and contrastive ranking, keeping compression extractive while avoiding decoder-based rewriting. Across single-hop and multi-hop QA benchmarks, multiple readers, retrieval depths, and retriever settings, EnComp achieves strong answer quality, including the leading aggregate EM and F1 in our main evaluations, while maintaining low compression latency and under 2 GB peak compressor memory. These results support encoder-only sentence pruning as a practical alternative to larger compression pipelines for resource-constrained RAG.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00573160), and the High-Performance Computing Support Project, funded by the Government of the Republic of Korea (Ministry of Science and ICT).

References

Benayas, A., Sicilia, M. A., and Mora-Cantallops, M. A comparative analysis of encoder only and decoder only models in intent classification and sentiment analysis: Navigating the trade-offs in model size and performance. *Language Resources and Evaluation*, 59(3):2007–2030, 2025.

Chirkova, N., Formal, T., Nikoulina, V., and Clinchant, S. Provence: efficient and robust context pruning for retrieval-augmented generation. *arXiv preprint arXiv:2501.16214*, 2025.

Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H., and Wang, H. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1), 2023.

Ge, T., Hu, J., Wang, L., Wang, X., Chen, S.-Q., and Wei, F. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

Google. Gemini 3 flash: Frontier intelligence built for speed. https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/, December 2025. Accessed: 2026-04-29.

Ho, X., Nguyen, A.-K. D., Sugawara, S., and Aizawa, A. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. spaCy: Industrial-strength Natural Language Processing in Python. 2020. doi: 10.5281/zenodo.1212303.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Hwang, T., Cho, S., Jeong, S., Song, H., Han, S., and Park, J. C. EXIT: Context-aware extractive compression for enhancing retrieval-augmented generation. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4895–4924, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.253. URL https://aclanthology.org/2025.findings-acl.253/.

Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

Jiang, H., Wu, Q., Luo, X., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*, 2023.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Karpukhin, V., Oguz, B., Min, S., Lewis, P. S., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pp. 6769–6781, 2020.

Khattab, O., Santhanam, K., Li, X. L., Hall, D., Liang, P., Potts, C., and Zaharia, M. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*, 2022.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Li, Y., Dong, B., Lin, C., and Guerin, F. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*, 2023.

Li, Z., Hu, X., Liu, A., Zheng, K., Huang, S., and Xiong, H. Refiner: Restructure retrieval content efficiently to advance question-answering capabilities. *arXiv preprint arXiv:2406.11357*, 2024.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

Lu, S., Wang, H., Rong, Y., Chen, Z., and Tang, Y. TurboRAG: Accelerating retrieval-augmented generation with precomputed KV caches for chunked text. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6588–6601, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.334. URL https://aclanthology.org/2025.emnlp-main.334/.

Luo, L., Cao, Y., and Luo, P. AttnComp: Attention-guided adaptive context compression for retrieval-augmented generation. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 8456–8472, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.449. URL https://aclanthology.org/2025.findings-emnlp.449/.

Mu, J., Li, X., and Goodman, N. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352, 2023.

Nielsen, D. S., Enevoldsen, K., and Schneider-Kamp, P. Encoder vs decoder: Comparative analysis of encoder and decoder language models on multilingual nlu tasks. *arXiv preprint arXiv:2406.13469*, 2024.

Pan, Z., Wu, Q., Jiang, H., Xia, M., Luo, X., Zhang, J., Lin, Q., Rühle, V., Yang, Y., Lin, C.-Y., et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *arXiv preprint arXiv:2403.12968*, 2024.

Ram, O., Levine, Y., Dalmedigos, I., Muhlgay, D., Shashua, A., Leyton-Brown, K., and Shoham, Y. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023.

Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022a.

Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022b.

Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., Gallagher, A., Biswas, R., Ladhak, F., Aarsen, T., et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*, 2024.

Wingate, D., Shoeybi, M., and Sorensen, T. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. *arXiv preprint arXiv:2210.03162*, 2022.

Xu, F., Shi, W., and Choi, E. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*, 2024.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Yoon, C., Lee, T., Hwang, H., Jeong, M., and Kang, J. Compact: Compressing retrieved documents actively for question answering. *arXiv preprint arXiv:2407.09014*, 2024.

Limitations

Our approach relies on explicit sentence-level annotations for training, which in the HQA dataset were obtained manually. Although such labels could, in principle, be generated by advanced large language models (e.g., commercial systems such as GPT-5), this raises concerns about the cost of API usage and the reliability of LLM-as-judge annotations. Moreover, because our method performs sentence-level pruning, complex sentences that are long, noisy, or contain extraneous details remain only partially optimized for length. We anticipate that a finer-grained strategy–such as pruning at the phrase or clause level–could address this limitation.

Appendix

In the Appendix, additional implementation details, supplementary results and analyses which are not covered in the main text is provided.

A. Extended Implementation Details

This section reports our training environment and prompt templates. All training experiments are done on single NVIDIA RTX 4090 GPUs (multiple similar workstations: same GPU, similar CPUs and RAM sizes).

A.1. Training Configuration

We adopt LoRA (Hu et al., 2022) to finetune the pruning model at bfloat16 precision for parameter-efficient training while preserving generalization from the original weights. The model was trained with the following hyper-parameters:

- Batch size: $\{16, 32\}$
- Gradient accumulation steps: 8
- Learning rate: 7e-5
- Weight decay: 0.02
- Warmup steps: 200
- Training epochs: up to 6
- Optimizer: AdamW
- LoRA configuration: Rank = 64, Scaling = 16, Dropout = 0.1
- Loss hyperparameters: $\mu = 0.35$, $\lambda_{\text{dist}} = 1.0$, $\lambda_{\text{rank}} = 1.5$, $\lambda_{\text{zero}} = 1.0$, $\lambda_{\text{doc}} = 0.6$; document BCE positive-class weight: 5.0

Zero-anchor regularizer $\mathcal{L}_{\text{zero}}$ is a minor calibration regularizer. It applies $\text{SmoothL1}(\hat{\Delta}_i, 0)$ to noncritical sentences. This discourages clearly unhelpful sentences from receiving positive contribution scores.

Regarding the detailed model architecture, our model is a lightweight scoring architecture built on a pretrained ModernBERT encoder plus a learnable multi-head attention pooling layer. Instead of CLS or mean pooling, H learned query vectors (one per pooling head; H = number of pooling heads $= 8$, with hidden size divisible by H) attend over the token sequence to produce head-specific summaries that are concatenated and linearly projected. Padding is masked before and after softmax for stability. A dropout layer and a final linear unit map the pooled representation to a single scalar score.

Model selection was guided by validation loss and it took around 7 hours for each training (up to 6 epochs).

A.2. Data Augmentation & Pre-processing.

We also add some data augmentation operations: **(1)** Randomly drop extra noncritical sentences when inspecting (dropping) a critical sentence: 10%; **(2)** Randomly drop extra noncritical sentences when inspecting (dropping) a noncritical sentence: 10%; **(3)** Randomly insert punctuation between sentences: 20%; **(4)** Randomly add start word(phrase), end word(phrase): 5%. We only used the query and the corresponding context set as the **original** HotpotQA set provided; no cross-pairing (i.e., a query with a non-corresponding context set) was used.

Some pre-processing we used: **(1)** merge overlapped chunks from a same document to reduce duplication in parts and titles (not longer than 20 sentences); **(2)** too short chunks–having less than 4 sentences, were also merged to become longer chunks for less extreme length imbalance in dataset (not longer than 12 sentences).

A.3. More Inference Configuration

Document-level threshold and delta min $d_{min}, \delta_{\min}$ were determined by grid search for each model weight set as mentioned before. Our best result checkpoint is configured with around $d_{min} = 0.12, \delta_{\min} = 0.01$

A.4. QA Prompt Template

Below is the QA prompt template for LLM readers to consider the query with the provided compressed context and answer.

```
Context information is:
```{Merged Compressed Context}```

Given provided context (might not be
    sufficient for below query), answer the
     query without any explanation.
Query: `{question}`
Answer (in plain text):
```
```

Table 5. Performance across models and datasets by readers Llama-3.1-8B Instruct and Llama-3.3-70B Instruct with $k = 5$. Compression Latency (Lat↓) is in seconds, and Compression Ratio (Rate↓) is reported as a percentage. Best results are in **bold**.

	NQ				TQA				HQA				2Wiki				Musique			
	EM	F1	Lat	Rate	EM	F1	Lat	Rate	EM	F1	Lat	Rate	EM	F1	Lat	Rate	EM	F1	Lat	Rate
Llama-3.1-8B Instruct																				*Top-5 retrieved chunks*
Raw	35.4	47.3	0.000	100.0	60.9	69.8	0.000	100.0	30.2	40.9	0.000	100.0	23.5	30.2	0.000	100.0	4.3	10.8	0.000	100.0
CompAct	31.4	42.0	2.573	12.1	60.0	69.1	2.594	12.0	28.1	37.5	2.517	11.9	22.1	28.8	2.354	10.6	**6.1**	**13.6**	2.781	11.9
EXIT	31.3	42.6	0.332	45.8	58.1	66.9	0.335	41.6	27.8	38.2	0.343	39.6	20.7	26.5	0.350	33.4	4.1	10.3	0.346	36.2
RECOMP-abs	33.5	45.1	0.765	**7.8**	53.8	63.5	0.578	**7.6**	28.3	39.4	0.527	**7.0**	24.4	29.6	0.436	**5.5**	4.3	11.8	0.499	**6.6**
RECOMP-ext	33.8	45.1	**0.009**	47.4	57.4	65.7	**0.009**	47.7	26.7	36.5	**0.010**	50.6	19.2	25.9	**0.008**	50.1	4.2	10.5	**0.009**	48.6
Refiner	33.2	44.9	2.403	15.3	60.3	69.4	1.967	13.2	29.7	40.5	1.624	10.4	22.2	28.2	1.325	8.3	5.3	12.0	1.436	8.9
LongLLMLin	28.3	39.4	0.258	37.2	58.6	67.0	0.253	36.5	27.1	37.1	0.244	37.2	21.3	26.8	0.253	37.3	4.0	10.3	0.245	36.4
Provence	33.4	44.8	0.059	26.4	59.9	69.1	0.056	25.4	28.6	39.5	0.056	19.5	21.4	27.6	0.059	15.5	4.5	11.2	0.056	14.7
AttnComp	33.3	45.0	0.036	27.1	**61.2**	**70.2**	0.035	30.3	29.5	40.0	0.036	27.2	21.7	28.0	0.036	21.1	4.7	10.9	0.036	12.9
Ours	**34.4**	**46.5**	0.012	27.7	60.8	69.9	0.013	25.4	**31.5**	**42.3**	0.013	22.0	**24.9**	**30.8**	0.013	17.5	5.5	12.8	0.013	19.8
Llama-3.3-70B Instruct																				*Top-5 retrieved chunks*
Raw	36.8	51.1	0.000	100.0	65.8	75.2	0.000	100.0	35.6	47.8	0.000	100.0	27.2	34.9	0.000	100.0	8.2	16.8	0.000	100.0
CompAct	35.3	48.4	2.573	12.1	65.2	74.6	2.594	12.0	36.6	48.6	2.517	11.9	27.0	34.6	2.354	10.6	8.8	17.4	2.781	11.9
EXIT	33.4	46.3	0.332	45.8	64.1	73.5	0.335	41.6	33.4	44.9	0.343	39.6	23.5	30.0	0.350	33.4	7.1	15.6	0.346	36.2
RECOMP-abs	33.9	46.6	0.765	**7.8**	56.8	66.8	0.578	**7.6**	32.2	44.3	0.527	**7.0**	26.7	32.3	0.436	**5.5**	6.5	14.2	0.499	**6.6**
RECOMP-ext	36.4	49.1	**0.009**	47.4	65.0	74.1	**0.009**	47.7	32.7	44.1	**0.010**	50.6	22.0	30.0	**0.008**	50.1	7.0	15.3	**0.009**	48.6
Refiner	35.1	48.6	2.403	15.3	65.6	75.2	1.967	13.2	35.8	47.6	1.624	10.4	25.7	32.7	1.325	8.3	8.3	16.6	1.436	8.9
LongLLMLin	31.7	44.6	0.258	37.2	65.2	74.2	0.253	36.5	32.5	44.2	0.244	37.2	24.8	31.4	0.253	37.3	7.6	16.2	0.245	36.4
Provence	35.7	49.8	0.059	26.4	66.1	75.6	0.056	25.4	35.1	47.4	0.056	19.5	27.0	33.7	0.059	15.5	6.8	15.9	0.056	14.7
AttnComp	**36.5**	**50.2**	0.036	27.1	**67.7**	**77.1**	0.035	30.3	36.8	49.3	0.036	27.2	**32.4**	**39.3**	0.036	21.1	8.6	18.2	0.036	12.9
Ours	36.2	50.1	0.012	27.7	66.4	75.8	0.013	25.4	**37.0**	**49.6**	0.013	22.0	32.1	38.1	0.013	17.5	**9.0**	**18.4**	0.013	19.8

B. Additional Experimental Results

Tab. 5 reports the additional results under a smaller retrieval budget, $k = 5$. Our method remains competitive across both readers, achieving the best results on several datasets and staying close to the strongest baseline in the remaining cases. It also keeps consistently low compression latency, second only to RECOMP-ext, while retaining a moderate portion of the context. These results suggest that the proposed pruner remains effective even when the initial retrieved context is limited.

Tab. 6 provides additional results with stronger LLM readers, including Gemini-3-flash and GPT-5-mini under the low-effort setting. Across both datasets and readers, our method remains among the strongest compressors in answer quality while consistently achieving the lowest compression latency. AttnComp performs best with Gemini-3-flash, whereas our method is best or tied-best with GPT-5-mini. These results suggest that the proposed pruner remains effective with more capable readers and provides a favorable quality–efficiency trade-off.

Tab. 7 reports the numerical values used to produce Fig. 1 by Llama-3.3-70B-Instruct as LLM reader. The results show that our method attains competitive EM while requiring substantially lower peak memory ($\leq 2GB$) and compression latency ($\sim 45ms$) than most high-accuracy baselines, supporting its suitability for resource-constrained deployment. Similarly, we also provide results at $k = 20$ with Llama-3.1-8B-Instruct as the reader in Tab. 8.

Table 6. Performance across compressors on HotpotQA and 2WikiMultiHop datasets using Gemini-3-flash (low effort) and GPT-5-mini (low effort) readers with $k = 20$.

Model	HQA				2Wiki				Reader
	EM	F1	Lat	Rate	EM	F1	Lat	Rate	
CompAct	43.8	57.3	4.696	3.7	45.9	51.7	4.905	3.5	Gemini-3-flash (low)
EXIT	42.2	56.0	1.465	34.0	42.4	49.0	1.492	29.4	
RECOMP-abs	41.4	55.5	1.206	**1.8**	45.5	52.3	1.145	**1.4**	
Refiner	43.9	57.8	3.915	4.4	47.8	54.5	3.342	3.7	
LongLLMLin	42.3	56.6	1.104	30.0	42.7	49.8	1.133	30.2	
Provence	44.2	58.4	0.199	13.8	47.6	54.1	0.197	11.9	
AttnComp	**47.4**	**62.4**	0.124	11.0	**55.2**	**63.8**	0.130	7.0	
Ours	46.8	61.7	**0.045**	13.8	52.3	59.0	**0.046**	11.5	
CompAct	41.8	57.1	4.696	3.7	34.4	43.3	4.905	3.5	GPT-5-mini (low)
EXIT	40.3	55.8	1.465	34.0	36.7	47.5	1.492	29.4	
RECOMP-abs	38.5	53.1	1.206	**1.8**	35.8	43.3	1.145	**1.4**	
Refiner	39.9	54.8	3.915	4.4	35.8	44.0	3.342	3.7	
LongLLMLin	40.1	55.5	1.104	30.0	37.1	48.6	1.133	30.2	
Provence	43.0	58.9	0.199	13.8	41.9	50.9	0.197	11.9	
AttnComp	43.7	60.1	0.124	11.0	**44.6**	54.9	0.130	7.0	
Ours	**44.2**	**60.8**	**0.045**	13.8	44.6	**54.9**	**0.046**	11.5	

Table 7. Detailed quality–latency–memory results corresponding to Fig. 1. EM, F1, and latency are averaged across five datasets at $k = 20$ by Llama-3.3-70B-Instruct. Memory denotes the peak VRAM required. Rate denotes the compression rate (%).

Method	Memory (GB) ↓	EM ↑	F1 ↑	Lat ↓	Rate ↓
CompAct	14.34	34.23	43.85	4.777	3.7
EXIT	11.53	34.09	44.32	1.451	35.1
RECOMP-abs	6.78	32.38	42.12	1.260	**1.8**
RECOMP-ext	**0.88**	29.76	39.17	**0.036**	11.9
Refiner	18.10	32.86	42.61	4.710	5.7
LongLLMLing	14.48	35.06	45.55	1.113	30.1
Provence	1.95	34.73	45.08	0.194	15.5
AttnComp	6.80	**37.16**	**47.65**	0.125	9.3
Ours	1.86	36.98	47.53	0.045	16.0

Table 8. Detailed quality–latency–memory results with EM, F1, and latency are averaged across five datasets at $k = 20$ by Llama-3.1-8B-Instruct. Memory denotes the peak VRAM required. Rate denotes the compression rate (%).

Method	Memory (GB) ↓	EM ↑	F1 ↑	Lat ↓	Rate ↓
CompAct	14.34	30.29	39.43	4.777	3.7
EXIT	11.53	30.05	38.79	1.451	35.1
RECOMP-abs	6.78	28.98	38.00	1.260	**1.8**
RECOMP-ext	**0.88**	25.75	33.62	**0.036**	11.9
Refiner	18.10	29.30	37.82	4.710	5.7
LongLLMLing	14.48	30.78	39.51	1.113	30.1
Provence	1.95	30.86	39.65	0.194	15.5
AttnComp	6.80	31.39	40.27	0.125	9.3
Ours	1.86	**32.81**	**42.16**	0.045	16.0

The **codebase** is attached in the submission form and **model checkpoints** will be released after the review session.