EFFCOMP: Efficient Prompt Compression via Hybrid Reinforcement & Supervised Learning

Romtham Tungsununtham[1], **Piyawat Lertvittayakumjorn**[2], **Atiwong Suchato**[1], **Peerapon Vateekul**[1*]
[1] Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Thailand
[2] Google, United States
6672061921@student.chula.ac.th, piyawat@google.com, atiwong.s@chula.ac.th, peerapon.v@chula.ac.th

Abstract

Prompt compression is aimed at reducing input prompt lengths to enable cheaper, faster, and potentially higher-quality LLM predictions. However, existing prompt compression methods are often limited by modest compression gains, a risk of hallucination, and/or high compression latency. We propose EFFCOMP, an **Eff**icient prompt **Comp**ression framework for RAG-based question answering (QA). EFFCOMP employs BERT-style document reranker and sentence selector models to allow fast extractive prompt compression at the sentence level. Its extractive nature prevents hallucinations in the compressed prompts. Additionally, the training process using a hybrid reinforcement and supervised learning approach is designed to optimize the compression ratio while preserving LLM accuracy. Experiments on various QA datasets demonstrate that EFFCOMP outperforms state-of-the-art prompt compression methods achieving the best trade-off between compression ratio and accuracy with minimal latency. This makes it practical for real-world applications.

1 Introduction

Retrieval-augmented generation (RAG) enhances large language models (LLMs) by grounding their predictions in retrieved external documents to reduce hallucinations and address domain gaps (Lewis et al., 2020; Guu et al., 2020; Izacard and Grave, 2021). However, the retrieved documents could often be long, redundant, and partially irrelevant (Barnett et al., 2024; Lertvittayakumjorn et al., 2025). The resulting lengthy augmented prompts not only undermine the quality of LLM predictions (Shi et al., 2023) but also increase computation costs and inference latency (Duman Keles et al., 2023). Therefore, recent research has proposed prompt compression methods to reduce the input prompt lengths (e.g., by strategically compressing the retrieved contexts) to address these efficiency and quality problems (Li et al., 2025; Wingate et al., 2022; Chevalier et al., 2023).

Despite these advancements, existing methods face several challenges and trade-offs. Abstractive prompt compressors generate a shorter context from a given long one, but they risk fabricating details or answering the questions themselves (Xu et al., 2024; Alansari and Luqman, 2025). Extractive prompt compressors, by contrast, preserve grounding by selecting and keeping the most important parts of the original context, but they often yield only modest compression ratios, e.g., 2x–7x in (Jiang et al., 2024; Pan et al., 2024; Fei et al., 2025; Zhao et al., 2025a,b). Furthermore, some compressors rely on large models or iterative algorithms, which can cause substantial latency and offset the benefits of compression (Yoon et al., 2024).

To overcome these challenges, we propose EFFCOMP, a novel prompt compression framework for RAG-based question answering (QA). The framework aims to boost the efficiency of prompt compression and subsequent LLM inference while maintaining (or even improving) the quality of the final output. EFFCOMP consists of two main steps. The first step reranks and filters the retrieved documents using a reranker. Then the second step selects important sentences in the remaining documents and concatenates them to form a compressed context for LLM inference. The novelty of EFFCOMP also lies in the training process of its sentence selector. This process first intermediately finetunes the model on a QA dataset and finally finetunes it using a hybrid reinforcement and supervised learning approach. We compare our framework against reranker-only and other state-of-the-art prompt compression methods on four QA datasets. The experiments show that EFFCOMP achieves higher QA accuracy and a strong compression ratio, outperforming other compression

*Corresponding author

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 969–995
November 6-10, 2026 ©2026 Association for Computational Linguistics

methods. Overall, our contributions are threefold:

We introduce EFFCOMP, a sentence-level extractive prompt compression framework for RAG-based QA. Its key innovations include (1) a sentence selector that jointly considers all shortlisted documents, (2) a reward function tailored to prompt compression for RAG-based QA, (3) a training data construction strategy for effective compressor learning, and (4) a hybrid RL-SFT training recipe that combines supervised and reward-driven learning. (See Section 3)

We evaluate EFFCOMP on four QA datasets, demonstrating that it achieves substantially high extractive compression while maintaining strong QA performance. (See Section 4)

We conduct additional analyses to test performance of EFFCOMP on LongBench, its performance of EFFCOMP with commercial target LLMs, the ability to capture relevant context sentences, and the effects of different training strategies of EFFCOMP. (See Section 5)

2 Problem Formulation

In a RAG-based QA setting, let q denote a question (so-called an input query) and $D = \langle d_1, d_2, \ldots, d_n \rangle$ denote a list of n retrieved documents ordered by the retrieval scores decreasingly. The full context $\mathbb{D}$ before compression is the concatenation of all $d_i \in D$ and has the length of $|\mathbb{D}|$ tokens. RAG combines q and $\mathbb{D}$ to form an input prompt and feeds it to a target LLM M, which then outputs $M(q, \mathbb{D})$ to answer the question q. Prompt compression aims to find a compressed context $\mathbb{C}$ such that $|\mathbb{C}| < |\mathbb{D}|$ and $M(q, \mathbb{C})$ is a correct answer to q. We measure correctness using accuracy, where the accuracy (Acc) for a given context X equals 1 if $M(q, X)$ contains the reference answer of q (denoted as y_q) after text normalization[1]; otherwise, 0. Following Jiang et al. (2024), we define the compression ratio as $|\mathbb{D}|/|\mathbb{C}| = 1/\tau$. EFFCOMP aims to optimize the correctness metric while achieving a competitively high compression ratio.

3 EFFCOMP: The Efficient Prompt Compression Framework

Based on the question q, EFFCOMP compresses the retrieved context in two main steps, one at the document level and the other at the sentence level, as illustrated in Figure 1.

Step 1: Document reranking. Given n retrieved documents $\langle d_1, d_2, \ldots, d_n \rangle$, an off-the-shelf document reranker is used to compute a relevancy score for each document d_i. The reranker takes the question q and a document d_i as input and outputs a score indicating the relevance of d_i to q. Only the top k documents with the highest scores are kept and then sorted by their scores in decreasing order. This is performed because evidence appearing earlier in an input prompt is more likely to receive stronger attention from LLMs (Tang et al., 2025). We experiment with three BERT-style document rerankers (Devlin et al., 2019), including BGE (Xiao et al., 2023), Jina (Günther et al., 2023), and GTE (Zhang et al., 2024). All of them are relatively lightweight models and can compute the relevancy scores in batches, ensuring high efficiency.

Step 2: Sentence selection. EFFCOMP concatenates the reranked documents from Step 1 and splits the entire concatenated text into a list of sentences $S = \langle s_1, s_2, \ldots, s_m \rangle$. After that, a sentence selector model is applied to identify sentences that should be kept in the final compressed context. Our sentence selector also relies on the BERT architecture for high efficiency. It takes the question q and all the m sentences in S as input and predicts either 0 or 1 for each sentence s_i. Specifically, the input representation $I(q, S)$ at the token level is "`[CLS]` $q_1 q_2 \ldots q_l$ `[SEP][SEN]` $t_{11} t_{12} \ldots t_{1h_1}$ `[SEN]` $t_{21} \ldots t_{2h_2}$ $\ldots$ `[SEN]` $t_{m1} \ldots t_{mh_m}$ `[SEP]`" where t_{ij} is the j-th token of s_i and q_j is the j-th token of q. Apart from the standard `[CLS]` and `[SEP]` tokens, we prepend a special `[SEN]` token to each sentence to support sentence-level classification. The sentence selector produces token-level predictions, but only the outputs at the `[SEN]` tokens are used, indicating whether each corresponding sentence is retained or discarded. The retained sentences are concatenated in order to form the compressed context $\mathbb{C}$.

Overall, the design of EFFCOMP offers several advantages. First, the finest level of selection is at the sentence level (rather than the token level). This renders a compressed context that remains understandable even under extreme compression and aligns naturally with how target LLMs were trained. Furthermore, the sentence selector in Step 2 considers all remaining context sentences simultaneously, enabling it to discard useful but redundant

[1] Following the evaluation script for SQuAD version 2.0 (Rajpurkar et al., 2018), we perform text normalization by lowercasing the text, removing punctuation and articles, and collapsing extra whitespace.

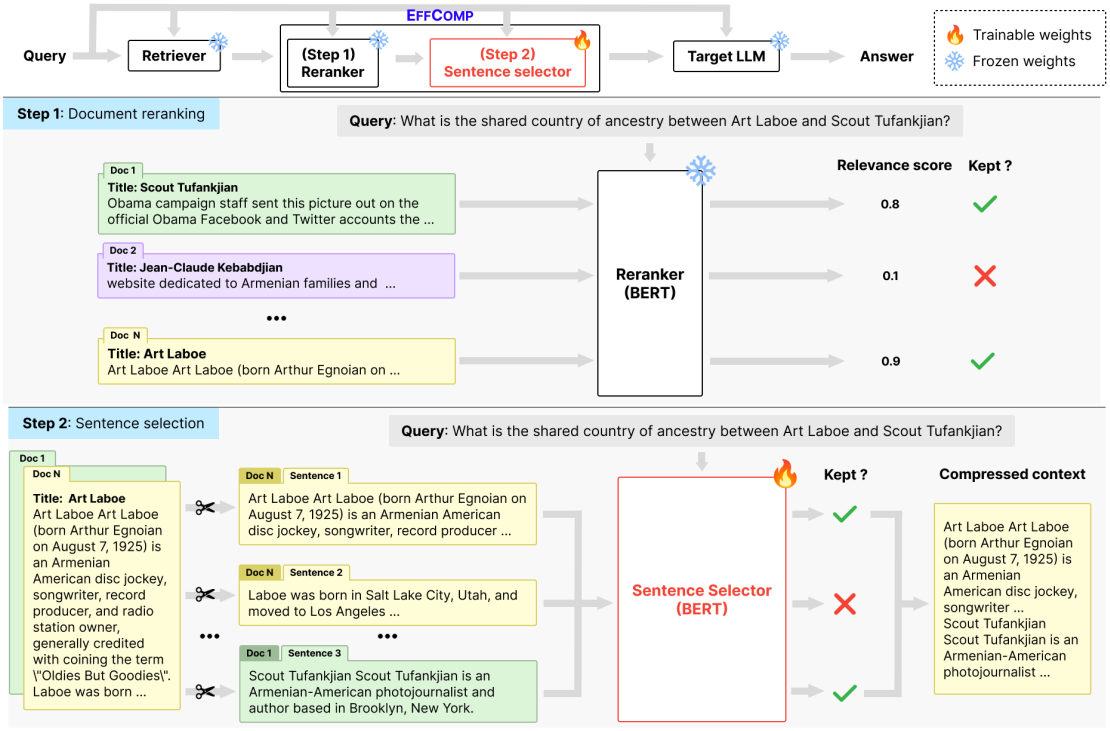


Figure 1: The inference workflow of EFFCOMP. The top row in the figure illustrates where EFFCOMP is in the prediction pipeline. In Step 1, retrieved documents are filtered, reordered, and concatenated. In Step 2, they are decomposed into sentences, from which necessary ones are selected and concatenated to form the compressed context for LLM prediction.

sentences to maximize the compression ratio. This simultaneous processing remains efficient thanks to Step 1, which already discards a significant number of irrelevant documents.

The remainder of this section explains the training process of the BERT-based sentence selector, which has two phases. Phase I first finetunes the model to identify sentences that likely contain the answer. Phase II further finetunes it to select other necessary context sentences and return a concise prompt sufficient for the target LLM.

3.1 Training Process Phase I: Intermediate Finetuning

Training set construction. We adapt training examples from a QA dataset to be our supervised finetuning examples. To explain, each finetuning example consists of a question q from the dataset and a list of retrieved documents D with $|D|= 30$. We concatenate all the documents in D, split the concatenated text into sentences S, and format them (together with q) as input to the sentence selector. Inspired by document-level label assignment in CompAct (Yoon et al., 2024), we assign a label to each sentence using the following rule: a sentence is labeled as *retained* if it contains the ground-truth answer string; otherwise, it is labeled *discarded*. In Phase I, we first train the sentence selector to predict these labels and then focus on additional necessary contexts in Phase II.

Training objective. The training objective is defined at the positions of `[SEN]` tokens. For each sentence $s_i \in S$, the action (retaining or discarding) is predicted at the position of its preceding `[SEN]` token. A masked cross-entropy loss is applied such that only `[SEN]` positions contribute to the objective, while all other tokens are ignored. This encourages the model to learn sentence-level decisions by treating each `[SEN]` token as the supervision point for its corresponding sentence.

3.2 Training Process Phase II: Final Finetuning

The model from Phase I focuses only on selecting sentences that contain answers. So, this Phase II aims to further finetune it to select other sentences that may not contain the answers but are necessary for the target LLM to make correct predictions (e.g., bridge sentences in multi-hop reasoning). We propose using a hybrid reinforcement learning (RL) and supervised learning (SL) approach for Phase II.

We choose RL, specifically a multi-armed bandit setting, as it is an optimization approach that suits the nature of our sentence selector. By design, RL can help further remove redundant and irrelevant context sentences while maintaining the final answer accuracy. However, as noted by Shandilya et al. (2024) and Wei et al. (2025), RL fine-tuning under a bandit formulation is computationally expensive and time-consuming because a single non-differentiable reward must be used to optimize multiple actions. To overcome this issue, we incorporate SL, which incurs little overhead and generally converges faster because actions are directly supervised. This results in our hybrid strategy that, for each epoch, updates the model weights using an RL loss followed by an SL loss, as explained next.

3.2.1 Reinforcement Learning

We consider our sentence selector as a policy model π_θ and use reward signals derived from the target LLM and ground-truth QA answers to optimize it.

Problem formulation. We formulate our task as a contextual multi-armed bandit problem (Jung and Kim, 2024; Lu et al., 2010), which terminates after a single decision step. A state σ is the input $I(q, S)$ to the policy model where q is a question and S is the list of context sentences from the top-k reranked documents. The policy π_θ takes σ as input and produces output probability distributions for all input tokens. However, consistent with the intermediate finetuning phase, only the probability distributions at the `[SEN]` tokens are used to derive binary actions $a_i \in \{0, 1\}$ indicating whether the corresponding sentences are retained (1) or discarded (0). With m sentences in S, one state produces multiple actions $A = \langle a_1, a_2, \ldots, a_m \rangle$ in parallel, where each action is sampled from Categorical $(\pi_\theta(\cdot|\sigma, i))$ The compressed context $\mathbb{C}$ is then formed by concatenating all retained sentences. The target LLM M produces $M(q, \mathbb{C})$, and a reward r for A is then computed. A trial, therefore, consists of a single state σ, the full list of binary actions A, and the resulting reward r.

Reward function. We define the reward r as

$$r = \begin{cases} \alpha + (1-\alpha)(1-\tau), & \text{if Accuracy = 1} \\ -(1-\alpha)(1-\tau), & \text{otherwise} \end{cases}$$

where $\tau = |\mathbb{C}|/|\mathbb{D}|$ and the Accuracy is obtained by comparing $M(q, \mathbb{C})$ against the ground truth answer of q. This reward jointly maximizes QA accuracy and compression. During validation, we use it to select the best model checkpoint for evaluation where $\alpha \in (0, 1)$ controls the relative importance between the accuracy and the compression ratio. In practice, we prefer a relatively high α to avoid extreme compression at the expense of accuracy.

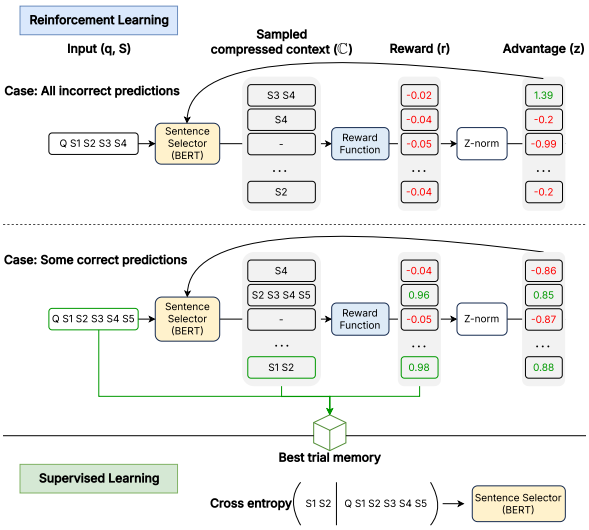


Figure 2: A single epoch of the final finetuning (Phase II). In each iteration, we perform reinforcement learning, followed by supervised learning. Reward normalization encourages higher compression when some predictions are correct and lower compression otherwise.

Optimization. To train the policy π_θ, we adapt Group Relative Policy Optimization (GRPO) (Shao et al., 2024), a policy gradient method that removes the value function used in PPO (Schulman et al., 2017) and instead efficiently estimates advantages by normalizing rewards within a group of sampled trials. Given a group size b, for each state, we sample a group of actions $\mathbf{A} = \{A_1, \ldots, A_b\}$ from π_θ, each producing a compressed context $\mathbb{C}_i$ and reward r_i. The reward set $R = \{r_1, \ldots, r_b\}$ is z-normalized in group to yield relative advantages $Z = \{z_1, \ldots, z_b\}$ used for updating the policy.

Thanks to our reward design, when at least one A_i results in a correct LLM output, the A_i with the highest compression ratio receives the greatest advantage, reinforcing the policy to increase compression. Conversely, if all $A_i \in \mathbf{A}$ lead to incorrect LLM outputs, the A_i with the lowest compression ratio is assigned the highest advantage, incentivizing the policy to reduce compression and preserve more information. See Figure 2 for an illustration.

Training set construction. For each QA training example (q, y_q), we perform Step 1 (document reranking) to obtain context sentences S from the

top-k documents to form the input $I(q, S)$. However, we exclude some training examples. First, we exclude examples that the target LLM cannot answer correctly even with all sentences in S provided, as further compression is unlikely to help. Given our reward function, including such examples would bias the policy to always retain all sentences, making the training inefficient. Second, we exclude examples that the target LLM can answer correctly even without any context. Keeping these examples would encourage the policy to discard all the sentences, resulting in non-grounded LLM outputs. This example filtering step ensures that the training set only contains examples where context is necessary and compression choices truly matter.

3.2.2 Supervised Learning

During GRPO, we sample multiple A_i for each state σ. To enhance training efficiency, we introduce a memory C to store the best trial of each state observed so far (i.e., the best list of sentence-level binary actions so far for each example). For each state σ, we begin storing a trial (σ, A_i, r_i) in C when $r_i > 0$ and update it when we find another trial (σ, A_j, r_j) where $r_j \geq r_i$. After the RL update in each epoch, these best trials in C are used as silver labels to finetune the policy (i.e., the sentence selector) in a supervised learning manner. This concludes the epoch. Then we begin the next iteration with RL and SL again. We refer readers to Appendix A for the detailed algorithm.

4 Experiments

We trained EFFCOMP on the HotpotQA training set and evaluated it on four open-domain QA datasets: HotpotQA (HQA; Yang et al. (2018)), 2WikiMultiHopQA (2WikiQA; Ho et al. (2020)), Natural Questions (NQ; Kwiatkowski et al. (2019)), and TriviaQA (TQA; Joshi et al. (2017)).

4.1 Experimental Setup

Implementation details. Following prior work (Yoon et al., 2024; Xu et al., 2024), we implemented the document retrieval system using Contriever (Izacard et al., 2021), finetuned on MSMARCO (Bajaj et al., 2018). For each question, we retrieved $n = 30$ documents from the 2018 Wikipedia corpus (Karpukhin et al., 2020). Additionally, we used Gemma-2-9B-IT[2] (Team et al., 2024) as the target LLM. For EFFCOMP, we tried three document rerankers, including BGE[3] (Xiao et al., 2023), Jina[4] (Günther et al., 2023), and GTE[5] (Zhang et al., 2024), to rerank and select the top-k documents ($k = 10$) in Step 1 of Figure 1. For Step 2, we used ModernBERT-base[6] (Warner et al., 2024) as the sentence selector and used the BGE reranker to select documents as input for finetuning the sentence selector. We also evaluated EFFCOMP without Step 1, which passed sentences from n documents to Step 2 without reranking. This variant is denoted as "No reranker" in the result tables.

Baselines. We compared EFFCOMP against several types of baselines. The simplest one was "Raw Documents", which inputs the n retrieved documents to the target LLM without compression. Next, reranker-only filtering baselines used a reranker (BGE, Jina, or GTE) to select top-k documents to feed into the target LLM. This is equivalent to EFFCOMP without Step 2. Furthermore, we compared EFFCOMP with state-of-the-art compression frameworks, including five extractive methods (Recomp-Extractive (Xu et al., 2024), LLMLingua2 (Pan et al., 2024), DAC (Zhao et al., 2025a), CPC (Liskavets et al., 2025), LongLLMLingua (Jiang et al., 2024)) and two abstractive methods (Recomp-Abstractive (Xu et al., 2024) and CompAct (Yoon et al., 2024)). Implementation details of the baselines are provided in Appendix B.6.

Metrics. We evaluate all methods using three metrics: compression ratio ($1/\tau$), accuracy (Acc), and C_β score. The C_β score combines compression ratio and accuracy into a single metric for easier comparison. It is defined as the weighted harmonic mean between accuracy and compression rate:[7]

$$\mathrm{C}_\beta = \frac{(1+\beta^2) \cdot \mathrm{Acc} \cdot (1-\tau)}{\beta^2 \cdot \mathrm{Acc} + (1-\tau)}$$

where β is the weighting factor. Lower β places more emphasis on accuracy. We report $\mathrm{C}_{0.5}$, which places greater emphasis on accuracy, and C_1, which equally balances accuracy and compression.

[2]https://huggingface.co/google/Gemma-2-9B-IT

[3]https://huggingface.co/BAAI/bge-reranker-base

[4]https://huggingface.co/jinaai/jina-reranker-v2-base-multilingual

[5]https://huggingface.co/Alibaba-NLP/gte-reranker-modernbert-base

[6]https://huggingface.co/answerdotai/ModernBERT-base

[7]The compression rate $(1 - 1/\tau)$ is a normalized transformation of the compression ratio into the range $[0, 1]$, enabling it to be combined with accuracy in the harmonic mean.

Method	HQA				2WikiQA				NQ				TQA			
	$1/\tau$	Acc	$C_{0.5}$	C_1	$1/\tau$	Acc	$C_{0.5}$	C_1	$1/\tau$	Acc	$C_{0.5}$	C_1	$1/\tau$	Acc	$C_{0.5}$	C_1
Raw Documents	1.0x	34.6	0.00	0.00	1.0x	29.1	0.00	0.00	1.0x	30.1	0.00	0.00	1.0x	77.4	0.00	0.00
Reranker-only methods																
BGE reranker	3.0x	35.2	0.39	0.46	3.0x	23.8	0.27	0.35	3.0x	29.6	0.33	0.41	3.0x	79.9	0.77	0.73
Jina reranker	3.0x	34.6	0.38	0.46	3.0x	24.4	0.28	0.36	3.0x	30.6	0.34	0.42	3.0x	80.1	0.77	0.73
GTE reranker	3.0x	34.7	0.38	0.46	3.0x	23.9	0.27	0.35	3.0x	30.6	0.34	0.42	3.0x	80.4	0.77	0.73
Compression frameworks																
LLMLingua2	5.4x	25.0	0.29	0.38	5.5x	19.8	0.23	0.32	5.3x	23.8	0.28	0.37	5.4x	72.9	0.75	0.77
DAC	5.1x	24.7	0.29	0.38	5.0x	21.0	0.25	0.33	5.2x	22.7	0.27	0.35	5.2x	73.5	0.75	0.77
CPC	5.6x	30.5	0.35	0.45	5.9x	21.8	0.26	0.35	5.2x	28.0	0.32	0.42	5.4x	78.6	0.79	0.80
LongLLMLingua	5.1x	31.8	0.36	0.46	5.1x	21.8	0.26	0.34	5.1x	25.7	0.30	0.39	5.1x	78.4	0.79	0.79
$\text{Recomp}_{\text{Extractive}}$	29.2x	25.6	0.30	0.40	29.3x	17.4	0.21	0.30	29.8x	22.8	0.27	0.37	32.0x	73.1	0.77	0.83
$\text{Recomp}_{\text{Abstractive}}$	**132.1x**	27.9	0.33	0.44	**126.9x**	21.0	0.25	0.35	**48.3x**	27.6	0.32	0.43	**139x**	73.0	0.77	0.84
CompAct	48.7x	31.2	0.36	0.47	54.7x	16.8	0.20	0.29	44.2x	27.8	0.32	0.43	50.2x	77.6	0.81	0.87
Our work: EFFCOMP																
No reranker	26.8x	**37.4**	**0.43**	**0.54**	11.7x	**28.9**	**0.34**	**0.44**	9.8x	**31.1**	**0.36**	**0.46**	32.8x	78.9	0.82	0.87
BGE reranker	39.5x	36.3	0.42	0.53	19.4x	25.0	0.29	0.40	14.1x	29.0	0.34	0.44	49.6x	79.5	**0.83**	**0.88**
Jina reranker	38.5x	35.8	0.41	0.52	19.3x	25.8	0.30	0.41	18.8x	29.7	0.34	0.45	50.0x	79.6	**0.83**	**0.88**
GTE reranker	40.4x	35.5	0.41	0.52	20.4x	25.0	0.29	0.40	19.2x	29.7	0.34	0.45	54.1x	**79.8**	**0.83**	**0.88**

Table 1: Results on four QA benchmarks for Gemma-2-9B-IT. We report compression ratio ($1/\tau$), accuracy, and C_β for $\beta \in \{0.5, 1\}$. Higher is better. Best values are in bold.

4.2 Results

Table 1 reports the results on four datasets. EFFCOMP consistently achieved the highest C_β scores across all baselines, demonstrating a superior balance between compression efficiency and QA performance. Notably, although EFFCOMP was trained only on HQA, it maintained strong performance on the out-of-distribution datasets, including 2WikiQA, NQ, and TQA. This indicates that the learned sentence selection strategy generalizes effectively beyond the training dataset rather than being specific to HQA. The improvements are especially notable on multi-hop QA datasets such as HQA and 2WikiQA, where preserving supporting evidence is crucial. For example, on HQA, EFFCOMP achieved a C_1 score of 0.54, representing an approximately 15% relative improvement over the strongest prior compression framework. Compared to using the Raw Documents, EFFCOMP with rerankers achieved competitive or sometimes higher Acc with compression ratios of 14.1x–54.1x. For example, on HQA, EFFCOMP with BGE reranker achieved 36.3% accuracy with the compression ratio of 39.5x, while the Raw Documents got 34.6% accuracy with no compression (1x). This demonstrates that context compression, if done appropriately, can enhance rather than degrade the quality of LLM predictions. Without the rerankers, EFFCOMP got significantly lower compression ratios but usually slightly higher Acc, resulting in higher C_β. Compared to reranker-only methods, EFFCOMP with the same reranker achieved significantly higher compression ratios (e.g., from 3x to 32.8x–54.1x on TQA) and competitive or higher Acc scores. This highlights the immense impact of the sentence selection step of EFFCOMP on compression ratios and indicates that our sentence selector, trained on BGE reranked data, generalized effectively to other rerankers such as Jina and GTE. Furthermore, EFFCOMP outperformed all other compression frameworks in terms of accuracy and C_β, suggesting a better balance between the compression ratios and the QA quality.

5 Additional Analyses

In this section, we conduct additional analyses to better understand EFFCOMP. Specifically, we evaluate EFFCOMP on additional QA datasets and with commercial LLMs. We also analyze the contributions of the two finetuning phases to sentence selection and compare our hybrid RL-SL training strategy with alternative training strategies.

RQ1: How does EFFCOMP perform on the Longbench benchmark? We investigate how EFFCOMP performs in real-world out-of-distribution scenarios using the HQA-trained version of our compressor model. To do so, we use

Multi-hop															
Method	**HQA**					**2WikiQA**					**MuSiQue**				
	$1/\tau$	Acc	F1	$C_{0.5}$	C_1	$1/\tau$	Acc	F1	$C_{0.5}$	C_1	$1/\tau$	Acc	F1	$C_{0.5}$	C_1
Raw Documents	1.0x	36.5	46.6	0.00	0.00	1.0x	40.5	43.7	0.00	0.00	1.0x	20.5	26.5	0.00	0.00
DAC	6.5x	25.0	34.9	0.29	0.39	5.4x	24.0	28.4	0.28	0.37	6.7x	7.0	11.8	0.09	0.13
LLMLingua2	5.4x	33.0	42.0	0.37	0.47	5.7x	27.0	30.3	0.31	0.41	5.5x	16.0	21.8	0.19	0.27
LongLLMLingua	3.5x	32.0	41.9	0.36	0.44	3.5x	38.5	43.9	0.42	0.5	3.5x	15.5	21.1	0.18	0.25
CPC	5.1x	35.0	45.6	0.39	0.49	5.2x	38.5	43.2	0.43	0.52	5.1x	17.0	21.5	0.20	0.28
Our work: EFFCOMP															
Ours (5x)	5.2x	40.0	50.1	0.44	0.54	5.1x	43.0	47.0	0.47	0.56	5.1x	22.0	29.3	0.26	0.35
Ours (No reranker)	39.3x	35.5	46.1	0.41	0.52	13.0x	43.0	47.3	0.48	0.59	13.3x	**23.5**	**29.8**	**0.28**	**0.37**
Ours (BGE)	**65.8x**	**40.5**	49.0	**0.46**	**0.57**	22.5x	**50.0**	**53.7**	**0.55**	**0.66**	40.3x	20.0	26.2	0.24	0.33
Ours (GTE)	58.5x	**40.5**	**50.5**	**0.46**	**0.57**	21.9x	48.0	52.2	0.53	0.64	**43.2x**	18.0	25.5	0.22	0.30
Ours (Jina)	63.4x	38.0	47.7	0.43	0.55	**25.7x**	47.5	51.5	0.53	0.64	32.6x	22.5	28.1	0.27	**0.37**

Single-hop															
Method	**MultiFieldQA**					**NarrativeQA**					**Qasper**				
	$1/\tau$	Acc	F1	$C_{0.5}$	C_1	$1/\tau$	Acc	F1	$C_{0.5}$	C_1	$1/\tau$	Acc	F1	$C_{0.5}$	C_1
Raw Documents	1.0x	14.0	40.6	0.00	0.00	1.0x	**12.5**	**23.6**	0.00	0.00	1.0x	22.0	39.6	0.00	0.00
DAC	5.9x	4.7	26.9	0.06	0.09	7.2x	4.5	13.7	0.06	0.09	5.0x	15.5	24.3	0.18	0.26
LLMLingua2	5.6x	10.7	31.1	0.13	0.19	6.0x	12.0	20.8	0.14	**0.21**	**5.2x**	18.5	28.7	0.22	0.30
LongLLMLingua	4.4x	12.0	33.7	0.14	0.21	3.6x	10.5	19.6	0.13	0.18	3.3x	16.0	22.3	0.19	0.26
CPC	5.1x	20.0	45.3	0.24	0.32	5.5x	8.5	20.0	0.1	0.15	5.1x	19.5	34.9	0.23	0.31
Our work: EFFCOMP															
Ours (5x)	5.0x	**21.3**	**48.3**	**0.25**	**0.34**	5.4x	10.5	18.6	0.13	0.19	5.1x	22.5	39.0	0.26	**0.35**
Ours (No reranker)	24.3x	19.3	45.8	0.23	0.32	1.6x	12.0	22.1	0.14	0.18	3.0x	**24.0**	**40.2**	**0.28**	**0.35**
Ours (BGE)	28.2x	20.0	44.4	0.24	0.33	22.5x	10.5	18.2	0.13	0.19	3.0x	**24.0**	**40.2**	**0.28**	**0.35**
Ours (GTE)	**32.3x**	19.3	45.7	0.23	0.32	**26.1x**	9.5	16.1	0.12	0.17	3.0x	**24.0**	**40.2**	**0.28**	**0.35**
Ours (Jina)	28.6x	18.7	44.2	0.22	0.31	20.3x	12.0	19.6	**0.15**	**0.21**	3.0x	**24.0**	**40.2**	**0.28**	**0.35**

Table 2: Results on LongBench QA datasets (out of distribution) using Gemma-2-9B-IT as the target LLM. The baselines are evaluated at a 5x compression ratio. We report compression ratio ($1/\tau$), accuracy, token level F1 score, and C_β. Higher is better, and best values are in bold.

the QA (English) subset of the LongBench benchmark (Bai et al., 2024), which includes both multi-hop and single-hop QA datasets. The multi-hop datasets consist of HQA, 2WikiQA, and MuSiQue, which require reasoning over multiple pieces of evidence. In contrast, the single-hop datasets represent a more distinct out-of-distribution setting from the HQA training data and cover diverse long-context domains: MultiFieldQA requires answering questions based on long articles, NarrativeQA focuses on questions about stories and scripts, and Qasper requires answering questions based on NLP research papers. We truncate the context length to match the maximum input length of Gemma-2-9B-IT. Although EFFCOMP is designed as an adaptive compression method that dynamically determines the compression level for each example, for this research question we additionally evaluate it under fixed compression settings for fair comparison with prior extractive baselines. For fixed compression, we use an EFFCOMP variant that selects the top-ranked sentences to achieve the target compression ratio. We report the compression ratio, Acc, F1, and C_β. For reranker-based variants, we first use the reranker to select the top 10 documents before applying sentence-level compression. In Table 2, EFFCOMP achieves the best C_β scores across all multi-hop settings compared to the other baselines and outperforms all compression baselines in single-hop settings. Furthermore, reranker-based EFFCOMP achieves the highest C_β scores in five out of six evaluated datasets, highlighting the importance of document-level filtering.

RQ2: How does EFFCOMP perform with commercial LLMs? To assess whether EFFCOMP generalizes to larger commercial LLMs, we further evaluate EFFCOMP with larger commercial LLMs, including GPT-4o mini[8] and Claude Sonnet 4.5[9], using temperature 0. We apply the same EFFCOMP model reported in Table 1 to the first 500 examples

[8]https://developers.openai.com/api/docs/models/gpt-4o-mini

[9]https://platform.claude.com/docs/en/models/sonnet-4-5/overview

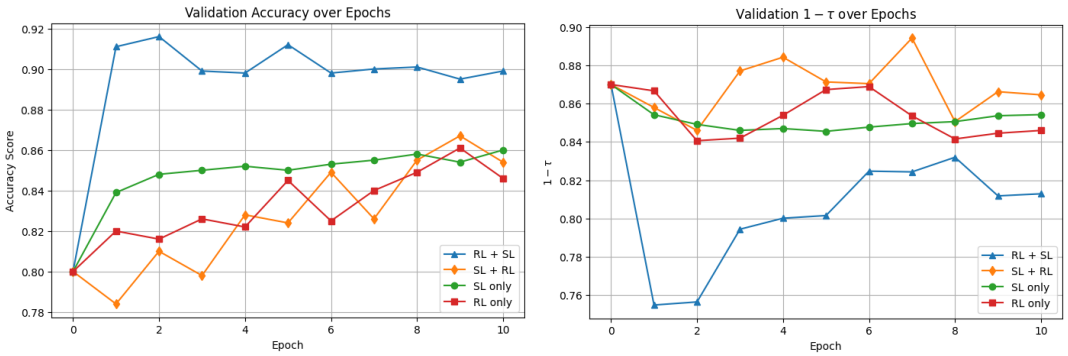


Figure 3: Ablation study of different training strategies for EFFCOMP on HQA. The plots compare (left) validation accuracy and (right) compression rate $1 - \tau = 1 - |\mathbb{C}|/|\mathbb{D}|$ during validation across ten epochs. For both metrics, the higher values are better.

Method	$1/\tau$	Acc	$C_{0.5}$	C_1
Target LLM: Claude Sonnet 4.5				
Raw Documents	1.0x	28.0	0.00	0.00
LLMLingua2	5.4x	40.2	0.45	0.54
DAC	5.1x	40.2	0.45	0.54
CPC	5.6x	42.4	0.47	0.56
LongLLMLingua	5.1x	43.6	0.48	0.57
Recomp Extractive	28.5x	36.8	0.42	0.53
Recomp Abstractive	134.7x	31.4	0.36	0.48
CompAct	49.0x	36.4	0.42	0.53
Our work: EFFCOMP				
EFFCOMP (No reranker)	28.0x	41.6	0.47	0.58
EFFCOMP (BGE reranker)	**43.6x**	**45.0**	**0.50**	**0.62**
Target LLM: GPT-4o mini				
Raw Documents	1.0x	35.8	0.00	0.00
LLMLingua2	5.4x	30.4	0.35	0.44
DAC	5.1x	27.0	0.31	0.40
CPC	5.6x	32.8	0.37	0.47
LongLLMLingua	5.1x	35.8	0.40	0.50
Recomp Extractive	28.5x	28.6	0.33	0.44
Recomp Abstractive	134.7x	28.6	0.33	0.44
CompAct	49.0x	32.6	0.38	0.49
Our work: EFFCOMP				
EFFCOMP (No reranker)	28.0x	**36.2**	**0.41**	**0.53**
EFFCOMP (BGE reranker)	43.6x	35.0	0.40	0.52

Table 3: Performance of prompt compression methods with commercial LLMs on HQA.

of the HQA test set. The results are presented in Table 3.

These results demonstrate that EFFCOMP generalizes effectively across commercial LLMs in terms of C_β. With Claude Sonnet 4.5, EFFCOMP with the BGE reranker achieves the highest $C_{0.5}$ and C_1 scores of 0.50 and 0.62, respectively, while attaining a 43.6x compression ratio. With GPT-4o mini, EFFCOMP without reranking achieves the highest $C_{0.5}$ and C_1 scores of 0.41 and 0.53, respectively, while the BGE-reranked variant remains competitive with $C_{0.5} = 0.40$ and $C_1 = 0.52$ at a higher compression ratio of 43.6x. Overall, these results show that EFFCOMP maintains a strong trade-off between compression and QA performance across different commercial target LLMs. This generalization possibly stems from our training set construction (Section 3), which uses highly informative examples identified by a relatively small LLM that generalize effectively to different target LLMs.

RQ3: How do the two finetuning phases contribute to the accuracy of sentence selection? This RQ investigates the effects of Phase I (intermediate finetuning) and Phase II (final finetuning) on the quality of compressed contexts. Specifically, we conduct an ablation study by training our sentence selector under three configurations: Phase I only, Phase II only, and our proposed Phase I + Phase II. We evaluate these models on HQA, a multi-hop QA dataset that provides gold labels specifying supporting context sentences required to answer a question. We measure the recall of these supporting sentences by checking whether at least 50% of the tokens from each supporting sentence appear as a subsequence in the compressed output. The results show that the Phase I-only model achieves a supporting fact recall of 75.83%, while the Phase II-only model achieves 84.13%. Combining both phases further increases the supporting fact recall to 91.09%. These results confirm that while Phase I focuses primarily on sentences containing the direct answers, Phase II significantly helps the model retrieve critical "bridge" sentences, thereby improving its effectiveness in multi-hop reasoning.

RQ4: How does the hybrid RL-SL finetuning approach perform compared to other approaches? We conduct an ablation study on the HQA dataset to evaluate four finetuning strategies: (1) reinforcement learning (RL) only, (2) supervised learning (SL) only using the best sampled trials, (3) the hybrid approach that performs SL then RL in each iteration (SL+RL) and (4) the hybrid approach of EFFCOMP that performs RL then SL in each iteration (RL+SL). Figure 3 shows that, overall, the RL only and SL + RL strategies primarily optimize compression, maintaining competitive accuracy even at high compression rates, but require more training epochs to converge. Meanwhile, the SL only strategy generally attains higher validation accuracy than the RL only approach, but exhibits a lower and more stable compression rate across epochs, suggesting a more limited search space. Our hybrid (RL+SL) strategy, in contrast, achieves notably higher validation accuracy during the early epochs, while exhibiting a relatively lower compression rate. Still, the compression rate is notably higher than several extractive compression methods as seen in Table 1. In conclusion, our hybrid (RL+SL) approach leads to a more accurate LLM output sooner than the non-hybrid approaches, demonstrating the efficiency of our training framework.

6 Related Work

Prompt compression. Li et al. (2025) categorize prompt compression methods into two types. Soft prompt methods compress inputs into continuous vectors readable only by the target LLM (Wingate et al., 2022; Chevalier et al., 2023; Mu et al., 2023; Ge et al., 2024). In contrast, hard prompt methods use truncation, selection, or summarization to keep the compressed prompts human-readable (Li et al., 2023; Jiang et al., 2024; Xu et al., 2024; Yoon et al., 2024; Fei et al., 2025). EFFCOMP falls into the latter category as we value interpretability and applicability across different target LLMs. So far, few hard prompt methods have employed reinforcement learning. Jung and Kim (2024) formulate prompt compression as a token-level contextual bandit problem for instruction-following tasks. Shandilya et al. (2024) propose TACO-RL for QA and summarization. They employ a constant penalty to counter the overly aggressive compression often encouraged by its optimization objectives. EFFCOMP, by contrast, operates at the sentence level with a tailored reward design and training set construction. This naturally regulates compression without the fixed limits, allowing higher compression while preserving QA accuracy.

Combining reinforcement learning (RL) and supervised finetuning (SFT). Recent studies show that using SFT to train LLMs followed by RL to align them with desired behaviors effectively enhances their capabilities (DeepSeek-AI et al., 2025; Ouyang et al., 2022). Instead of using one after the other, SuperRL (Liu et al., 2025) enhances the complex reasoning ability of LLMs using RL. However, when all sampled trials of an example yield zero reward, leading to uninformative RL signals, they switch to SFT with high-quality offline data. In contrast, our framework focuses on improving the efficiency of the training process while alleviating the zero-reward issue by using a dense reward and setting an RL hyperparameter (Section B.2) to encourage the exploration of various action choices. It then uses the best-performing trials from RL (instead of offline data) for SFT before alternating between RL and SFT in subsequent epochs.

7 Conclusion

We propose EFFCOMP, a prompt compression framework for RAG-based QA. EFFCOMP performs context selection at the document level and then the sentence level, rendering minimal yet sufficient context for the target LLM. Our hybrid finetuning achieves the best overall training performance in terms of accuracy and consistently obtains the strongest C_β scores among all baselines. Experimental results show that EFFCOMP yields high compression ratios (up to 54.1x) while maintaining or even improving the QA accuracy (up to 8.1% relative to the no-compression baseline).

Limitations

We acknowledge that EFFCOMP has some limitations. First, EFFCOMP was designed to tackle RAG-based question answering, so its algorithm may not work effectively on other tasks such as text summarization. Second, EFFCOMP is constrained by the context length of the underlying BERT-based encoder, which is currently limited to 8,192 tokens in state-of-the-art implementations. While EFFCOMP uses a reranker to fit the input into a fixed-size window, the method is still fundamentally bounded by the encoder's context-length

limitation. Third, while adaptive compression improves QA accuracy by dynamically preserving more informative content, it also produces variable compressed context lengths, which may lead to unstable or unexpectedly high inference costs in practical deployments. This limitation could potentially be mitigated by incorporating fixed-rate compression as we did in RQ1.

References

Aisha Alansari and Hamzah Luqman. 2025. Large language models hallucination: A comprehensive survey. *Preprint*, arXiv:2510.06265.

Anthropic. The claude 3 model family: Opus, sonnet, haiku.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pages 194–199.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846, Singapore. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. *Preprint*, arXiv:1911.02116.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. 2023. On the computational complexity of self-attention. In *Proceedings of The 34th International Conference on Algorithmic Learning Theory*, volume 201 of *Proceedings of Machine Learning Research*, pages 597–619.

Weizhi Fei, Xueyan Niu, XIE GUOQING, Yingqing Liu, Bo Bai, and Wei Han. 2025. Efficient prompt compression with evaluator heads for long-context transformer inference. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

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

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, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

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

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

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, arXiv:1705.03551.

Dongwon Jung, Qin Liu, Tenghao Huang, Ben Zhou, and Muhao Chen. 2024. Familiarity-aware evidence compression for retrieval-augmented generation. *Preprint*, arXiv:2409.12468.

Hoyoun Jung and Kyung-Joong Kim. 2024. Discrete prompt compression with reinforcement learning. *IEEE Access*, 12:72578–72587.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

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

Piyawat Lertvittayakumjorn, David Kinney, Vinodkumar Prabhakaran, Donald Martin Jr., and Sunipa Dev. 2025. Towards geo-culturally grounded LLM generations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 313–330, Vienna, Austria. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

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

Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. 2025. Prompt compression for large language models: A survey. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7182–7195, Albuquerque, New Mexico. Association for Computational Linguistics.

Barys Liskavets, Maxim Ushakov, Shuvendu Roy, Mark Klibanov, Ali Etemad, and Shane K. Luke. 2025. Prompt compression with context-aware sentence encoding for fast and improved llm inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):24595–24604.

Yihao Liu, Shuocheng Li, Lang Cao, Yuhang Xie, Mengyu Zhou, Haoyu Dong, Xiaojun Ma, Shi Han, and Dongmei Zhang. 2025. Superrl: Reinforcement learning with supervision to boost language model reasoning. *Preprint*, arXiv:2506.01096.

Tyler Lu, David Pal, and Martin Pal. 2010. Contextual multi-armed bandits. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 485–492, Chia Laguna Resort, Sardinia, Italy. PMLR.

Marc Marone, Orion Weller, William Fleshman, Eugene Yang, Dawn Lawrie, and Benjamin Van Durme. 2025. mmbert: A modern multilingual encoder with annealed language learning. *Preprint*, arXiv:2509.06888.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference*

of the European Chapter of the Association for Computational Linguistics, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981, Bangkok, Thailand. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

John Schulman. 2020. Approximating kl divergence. `http://joschu.net/blog/kl-approx.html`. Accessed: 2025-08-29.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Shivam Shandilya, Menglin Xia, Supriyo Ghosh, Huiqiang Jiang, Jue Zhang, Qianhui Wu, and Victor Rühle. 2024. Taco-rl: Task aware prompt compression optimization with reinforcement learning. *Preprint*, arXiv:2409.13035.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

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

Jiwei Tang, Jin Xu, Tingwei Lu, Zhicheng Zhang, YimingZhao YimingZhao, LinHai LinHai, and Hai-Tao Zheng. 2025. Perception compressor: A training-free prompt compression framework in long context scenarios. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4093–4108, Albuquerque, New Mexico. Association for Computational Linguistics.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

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

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *Preprint*, arXiv:2412.13663.

Quan Wei, Siliang Zeng, Chenliang Li, William Brown, Oana Frunza, Wei Deng, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. 2025. Reinforcing multi-turn reasoning in LLM agents via turn-level reward design and credit assignment. In *First Workshop on Multi-Turn Interactions in Large Language Models*.

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

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

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset

for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. 2024. CompAct: Compressing retrieved documents actively for question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21424–21439, Miami, Florida, USA. Association for Computational Linguistics.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, and 1 others. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1393–1412.

Ziyin Zhang, Zihan Liao, Hang Yu, Peng Di, and Rui Wang. 2026. F2llm-v2: Inclusive, performant, and efficient embeddings for a multilingual world. *Preprint*, arXiv:2603.19223.

Yi Zhao, Zuchao Li, Hai Zhao, Baoyuan Qi, and Liu Guoming. 2025a. DAC: A dynamic attention-aware approach for task-agnostic prompt compression. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19395–19407, Vienna, Austria. Association for Computational Linguistics.

Yunlong Zhao, Haoran Wu, and Bo Xu. 2025b. Leveraging attention to effectively compress prompts for long-context llms. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):26048–26056.

A Detailed Algorithm

A.1 Reinforcement learning

This section provides an in-depth description of the RL finetuning procedure. To train the policy π_θ, we modify GRPO by excluding the explicit KL penalty from the loss function. Instead, we monitor the approximate KL divergence (Schulman, 2020) between the updated policy and the reference policy during optimization and terminate updates early if it exceeds a threshold.

For each state σ, the policy produces multiple actions in parallel, denoted by $A = \langle a_1, a_2, \ldots, a_m \rangle$, where each action is independently sampled from a categorical distribution:

$$a_i \sim \text{Categorical}(\pi_\theta(\cdot \mid \sigma, i)), \quad \forall i \in \{1, \ldots, m\}. \tag{1}$$

To measure the degree of policy change, we define $\pi_\theta(A \mid \sigma)$ as the geometric mean of the probabilities across all sampled binary actions in A:

$$\pi_\theta(A \mid \sigma) = \left(\prod_{i=1}^{m} \pi_\theta(a_i \mid \sigma) \right)^{\frac{1}{m}} \tag{2}$$

Our reinforcement learning objective is then defined as

$$\mathcal{L}_{\text{RL}}(\theta) = -\mathbb{E}\left[\min\left(\frac{\pi_\theta(A \mid \sigma)}{\pi_{\theta_{\text{old}}}(A \mid \sigma)} z,\ \text{clip}\left(\frac{\pi_\theta(A \mid \sigma)}{\pi_{\theta_{\text{old}}}(A \mid \sigma)},\ 1-\epsilon,\ 1+\epsilon \right) z \right) \right], \tag{3}$$

where π_θ and $\pi_{\theta_{\text{old}}}$ denote the current and old policy models, respectively, and ϵ is a clipping hyperparameter, typically set to 0.2.

Additionally, to quantify the degree of action exploration, we compute the Shannon entropy (Shannon, 1948) of the policy π_θ at the state σ (considered only at the `[SEN]` positions) as

$$H(\pi_\theta(\cdot \mid \sigma)) = \mathbb{E}\left[-\frac{1}{m} \sum_{i=1}^{m} \sum_{a_i \in \{0,1\}} \pi_\theta(a_i \mid \sigma)\ \log \pi_\theta(a_i \mid \sigma) \right] \tag{4}$$

The final training loss is

$$\mathcal{L}_{\text{total}}(\theta) = \mathcal{L}_{\text{RL}}(\theta) \ - \ \lambda\, H(\pi_\theta(\cdot \mid \sigma)), \tag{5}$$

where λ controls the trade-off between exploitation and exploration. In practice, we set $\lambda \geq 0.1$, which encourages more exploration.

A.2 Best Trial Memory Update

Algorithm 1: EFFCOMP Update the best trial memory

Input: Buffer C (a dictionary mapping each state σ to its best trial), state σ, group actions $\mathbf{A}$, group rewards R

Output: Updated buffer C

for $(A_i, r_i) \in (\mathbf{A}, R)$ **do**

  if $r_i > 0$ *and* ($\sigma \notin C$ *or* $r_i \geq C[\sigma].\text{reward}$) **then**

    $C[\sigma] \leftarrow (A_i, r_i)$

return C

A.3 Finetuning the Sentence Selector

Algorithm 2: EFFCOMP Hybrid Reinforcement & Supervised Learning

Input: Training data Δ, TargetLLM, Intermediate trained policy π_θ
Hyperparameters: Adam optimizer (lr $= 1\times10^{-6}$), group size $g = 8$, $\alpha = 0.95$, epochs $= 5$, entropy coefficient $\lambda = 0.1$, clip value $\epsilon = 0.2$, rollout size $= 128$
Output: Optimized policy $\pi_\theta^\star$
Initialize the rollout buffer $\mathcal{B} \leftarrow \emptyset$ and the best trial memory $C \leftarrow \emptyset$;
for *example* $(q, y_q, S_q) \in \Delta$ **do**
  $\sigma \leftarrow I(q, S_q)$;
  Get actions $A^\star \leftarrow \arg\max_{A\in\{0,1\}^m} \pi_\theta(\cdot \mid \sigma)$;
  Build compressed context $\mathbb{C} \leftarrow \mathrm{Select}(S_q, A^\star)$;
  Obtain output $y \leftarrow \mathrm{TargetLLM}(q, \mathbb{C})$;
  Compute reward $r \leftarrow \mathrm{Reward}(y_q, y, S_q, \mathbb{C})$;
  UpdateMemory$(C, \sigma, \langle A^\star\rangle, \langle r\rangle)$ using Algorithm 1;
reference model $\pi_{ref} \leftarrow \pi_\theta$;
for *epoch* $= 1$ **to** epochs **do**
  reference model $\pi_{ref} \leftarrow \pi_\theta$;
  for *example* $(q, y_q, S_q) \in \Delta$ **do**
    $\sigma \leftarrow I(q, S_q)$;
    Sample actions $\mathbf{A} \leftarrow \{A_i\}_{i=1}^{g} \sim \pi_\theta(\cdot \mid \sigma)$ (Eq. 1);
    for $i = 1$ **to** g **do**
      $\mathbb{C}_i \leftarrow \mathrm{Select}(S_q, A_i)$;
      $y_i \leftarrow \mathrm{TargetLLM}(q, \mathbb{C}_i)$;
      $r_i \leftarrow \mathrm{Reward}(y_q, y_i, S_q, \mathbb{C}_i)$;
    Let $R \leftarrow \{r_i\}_{i=1}^{g}$;
    UpdateMemory$(C, \sigma, \mathbf{A}, R)$ using Algorithm 1;
    Compute relative advantages Z from R;
    Append $(\sigma, \mathbf{A}, Z)$ to the rollout collection $\mathcal{B}$;
    if $|\mathcal{B}| \geq$ *rollout size* **then**
      Update π_θ with final training loss (Eq. 5);
      reference model $\pi_{ref} \leftarrow \pi_\theta$;
      Reset $\mathcal{B} \leftarrow \emptyset$;
  for *each* $(S_b, A_b) \in C$ **do**
    Update π_θ with supervised cross-entropy on (S_b, A_b);
return $\pi_\theta^\star$;

B Additional Implementation Details

B.1 Input Representation of the Sentence Selector

For the sentence selector, we use ModernBERT-base (Warner et al., 2024) (∼149M parameters) whose input representation is illustrated in Figure 4.

B.2 Training Configuration

Intermediate finetuning of the sentence selector was conducted using a GeForce RTX 3090 GPU, while final finetuning and inference were conducted using an NVIDIA A100-SXM4-40GB GPU.

Intermediate Finetuning.

- Learning rate: 2×10^{-5}
- Number of epochs: 2
- Weight decay: 0.01
- Batch size: 4
- Precision: bfloat16
- Optimizer: AdamW
- Loss: cross-entropy with balanced class weights

Final Finetuning.

- **Model Setup**
  - Target model: Gemma-2-9B-IT
  - Quantization: BitsAndBytes 4-bit (nf4), compute dtype: fp16
  - Sampling: greedy
  - Model maximum length: 2048
- Learning rate: 1×10^{-6}
- Number of epochs: 5
- Optimizer: AdamW
- Precision: bfloat16
- Rollout size: 128
- Group size: 8
- Backward batch size: 8
- Updates per iteration: 4
- Entropy coefficient λ : 0.1
- Max gradient norm: 0.5
- Clip value ϵ : 0.2
- Target KL: 0.02
- Reward weight (α): 0.95
- Supervised loss: cross-entropy with balanced class weights

Note that we selected the best intermediate finetuning model checkpoint and the best finetuned model checkpoint based on the validation loss and the validation reward observed, respectively.

B.3 Dataset Information

We trained and evaluated EFFCOMP on four English open-domain QA datasets, including two multi-document benchmarks (HQA and 2WikiQA) and two single-document benchmarks (NQ and TQA). These datasets were used in both intermediate finetuning and final finetuning. Dataset statistics are summarized in Table 4.

For preprocessing, we used the NLTK 3.9.1 (Bird and Loper, 2004) sentence tokenizer to segment documents into sentences.

During intermediate finetuning, we excluded two types of examples: (1) examples with $|S| > 5120$ tokens, which rarely occur due to the small k used in document reranking, and (2) samples with ground-truth answers such as "yes", "no", or "no-answer", since they do not correspond to concrete evidence in the retrieved text and may introduce misleading supervision.

During final finetuning, we excluded samples exceeding 2048 tokens due to hardware memory constraints.

B.4 LLM Prompt Template

For both Gemma-2-9B-IT and Llama-3.1-8B-Instruct, we adopted the prompt template by Jung et al. (2024) to query the LLMs:

```
You are an expert in Question Answering.
Your job is to answer questions in 1 to 5
words based on the given context.
Question: {question}
Context: {context}
Answer:
```

B.5 Inference Configuration for Target LLMs

All the results are reported from a single run. The configurations of the target LLMs are shown below.

Gemma-2-9B-IT.

- Sampling: greedy
- Quantization: BitsAndBytes 4-bit (nf4), dtype: float16
- Max new tokens: 10
- Model maximum length: 6144

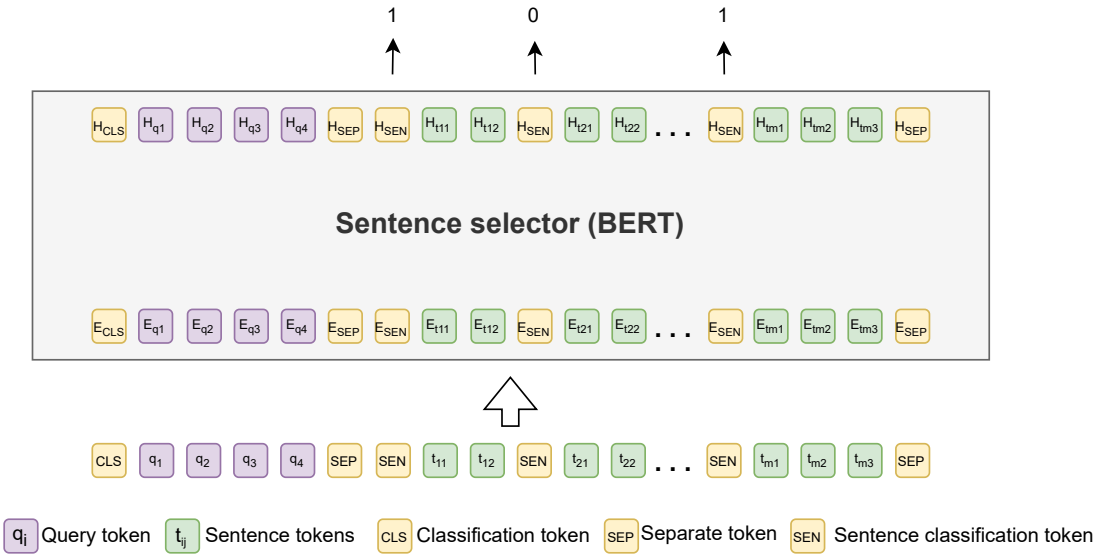


Figure 4: Input and output of the sentence selector. The input begins with the [CLS] token and the query q, then followed by the candidate sentences. Each sentence is prepended with a special [SEN] token. The [SEP] tokens were inserted after the query q and after the last token of the last candidate sentence. The sentence selector produces token-level outputs, but only predictions at the [SEN] positions are used to determine whether each corresponding sentence is retained or discarded.

Llama-3.1-8B-Instruct.

- Sampling: greedy
- Quantization: BitsAndBytes 4-bit (fp4), dtype: float32
- Max new tokens: 10
- Model maximum length: 5120

B.6 Baseline Details

This section provides additional details about the baselines in our experiments.

- **Raw Documents**: The top-30 retrieved documents are used directly without any compression.
- **Reranker-only methods**: The top-30 retrieved documents are reordered using a document reranker and the top-10 documents are selected as the compressed context. We evaluate three rerankers in this setting: BGE (∼278M parameters), Jina (∼278M parameters), and GTE (∼149M parameters).
- **LongLLMLingua** (Jiang et al., 2024): A task-aware method built on Llama-2-7B[10] that ranks documents and selects tokens using contrastive perplexity scoring. We set the dynamic compression rate fixed at 0.3, with the rate set to 0.45 for the ∼5× compression.
- **LLMLingua2** (Pan et al., 2024): A task-agnostic compression method trained via label distillation from GPT-4 on the MeetingBank dataset (Hu et al., 2023). We use the XLM-RoBERTa-large model[11] (∼550M parameters) (Conneau et al., 2020), which follows the LLMLingua2 training recipe. The compression rate is set to 0.2.
- **Recomp-Extractive** (Xu et al., 2024): An extractive approach that assigns scores at the sentence level using a dual-encoder model. We adopt the Contriever models (∼110M parameters) released by the authors for each dataset. For 2WikiQA, where no finetuned models are available, we use the Contriever finetuned on HQA. Following the previous research (Yoon et al., 2024) and similar compression ratio with our method, the top four sentences are selected as context.
- **Recomp-Abstractive** (Xu et al., 2024):

[10] https://huggingface.co/NousResearch/Llama-2-7b-hf

[11] https://huggingface.co/microsoft/llmlingua-2-xlm-roberta-large-meetingbank

Dataset	Train	Validation	Test
Base datasets			
HotpotQA	90,447	–	7,405
2WikiMultiHopQA	167,454	–	12,576
Natural Questions	152,148	–	5,499
TriviaQA	138,384	–	17,944
Intermediate finetuning			
HotpotQA	75,059	8,340	–
2WikiMultiHopQA	92,595	10,289	–
Natural Questions	133,060	14,785	–
TriviaQA	123,670	13,742	–
Final finetuning			
HotpotQA	19,207	1,000	–
2WikiMultiHopQA	15,522	1,000	–
Natural Questions	18,687	1,000	–
TriviaQA	21,136	1,000	–

Table 4: Number of examples in each dataset before and after filtering. The base dataset refers to the raw data, while the intermediate finetuning and final finetuning stages indicate the dataset sizes used in each phase.

An abstractive approach based on T5-large (∼770M) trained with summarization distillation over NQ, TQA, and HQA. The model compresses the retrieved evidence into shorter summaries. For 2WikiQA and HQA, we use the -trained version, while for TQA and NQ, dataset-specific models are applied.

- **CompAct** (Yoon et al., 2024): An iterative abstractive compression method implemented with Mistral-7B-Instruct[12] (Jiang et al., 2023). The model repeatedly summarizes the input until the target model signals completion. The number of documents per segment is set to 5, following the original implementation.

- **Context-aware Prompt Compression (CPC)** (Liskavets et al., 2025): A context-aware compression approach that assigns relative importance scores to sentences conditioned on the given question. The model is trained on a synthetic dataset using a contrastive learning objective and implemented an adaptor with Llama-3.2-1B-Instruct[13] (Jiang et al., 2023). We follow the original paper and set the target compressed length to 900 tokens, which corresponds to approximately a 5x compression when using 30 documents from the retriever.

- **DAC** (Zhao et al., 2025a): A prompt-agnostic compression method that integrates entropy and attention signals to compute token-level importance scores. We use the Qwen2-0.5B-Instruct [14] model (Yang et al., 2024) with the default configuration, and compression ratio set to 0.825 (around 5x).

[12]https://huggingface.co/cwyoon99/CompAct-7b

[13]deadcode99/cpc-1.0-llama-1b-ds-v5-iter66-lora-bidirectional-attn

C Additional Results and Analyses

C.1 Examples

To illustrate, we show a successful case of EFFCOMP in Table 5. In this case, Recomp-Abstractive and CompAct generated compressed contexts that are factually incorrect or irrelevant to the question, leading to wrong LLM predictions. By contrast, LongLLMLingua and EFFCOMP remained correctly grounded in the retrieved contexts, but EFFCOMP used 8.6 times fewer tokens than LongLLMLingua (88 vs. 758 tokens).

C.2 Performance under the In-Distribution Training Setting

We report the results of in-distribution and out-of-distribution testing for the other rerankers and the other target LLM (Llama-3.1-8B-Instruct) in Table 6. Training on the target dataset yielded both higher compression ratios and stronger QA metrics, while applying the model out of distribution resulted in certain reductions in compression ratios but only minor reductions in the QA metric.

C.3 Generalization to Another Target Model

We evaluated our method, trained on the HQA dataset with Gemma-2-9B-IT as the target model

[14]https://huggingface.co/Qwen/Qwen2-0.5B-Instruct

Question (from HQA): Stein had a part in the 1994 superhero fantasy film directed by whom
Ground-truth answer: Charles Russell

Method	$1/\tau \uparrow$	$\lvert\mathbb{C}\rvert \downarrow$	Compressed Context	LLM Prediction
Long-LLMLingua	5.01x	758	The Mask (film) The Mask is a 1994 American fantasy superhero comedy film directed by Charles Russell, produced by Bob Engelman [...] Ben Stein Benjamin Jeremy (born November 25, 1944) is an American writer, [...] He is well- screen econom teacher "Ferris Bueller's Day Off" (1986) and as Dr. Arthur Neuman in "The Mask" (1994)Son [...]	Charles Russell ✓
Recomp-Abstractive	189.75x	20	Ben Stein had a part in the 1994 superhero fantasy film "The Shadow" directed by Russell Mulcahy.	Russell Mulcahy ✗
CompAct	49.93x	76	The Shadow is a 1994 American superhero film [...] directed by Russell Mulcahy, that stars Alec Baldwin. The film co-stars John Lone, Penelope Ann Miller, [...] It was released to theaters on July 1, 1994 and received generally mixed reviews.	Russell Mulcahy ✗
EFFCOMP (BGE)	43.13x	88	He is most well-known on screen as the economics teacher in "Ferris Bueller's Day Off" (1986) and as Dr. Arthur Neuman in "The Mask" (1994) and "Son of The Mask (film) The Mask is a 1994 American fantasy superhero comedy film directed by Charles Russell, produced by Bob Engelman [...]	Charles Russell ✓

Table 5: Examples of compressed contexts and predictions from LongLLMLingua, Recomp-Abstrctive, CompAct, and EFFCOMP (with BGE reranker). $\lvert\mathbb{C}\rvert$ denotes the number of tokens of the compressed context. ✓ indicates a correct prediction, while ✗ indicates an incorrect prediction.

during training, against baseline approaches under the setting where compressed contexts were provided to another target model which is Llama-3.1-8B-Instruct. Table 7 reports the results on HQA, 2WikiQA, NQ, and TQA. Overall, the performance trends remain consistent across datasets, with our method achieving the best QA results. Furthermore, EFFCOMP with a reranker generally results in a higher compression rate, showing the importance of the reranker to our method.

C.4 Generalization to Long Generation Tasks

EFFCOMP is designed primarily for RAG-based QA tasks, as reflected by both the supervision used in Phase I and the reward function employed during RL training in Phase II. Consequently, the training objectives are not directly designed for long-form generation tasks such as document summarization. To examine whether EFFCOMP can nevertheless generalize to such tasks, we apply the EFFCOMP model trained on HQA directly to three summarization datasets from LongBench: GovReport, MultiNews, and QMSum. The results are reported in Table 8.

As expected, EFFCOMP does not consistently outperform the baselines on these long-generation tasks. Although EFFCOMP with a fixed compression ratio achieves compression ratios comparable to several existing compression methods, its ROUGE-L scores are generally lower than those of LongLLMLingua. The adaptive variant exhibits different behavior across datasets, achieving a substantially higher compression ratio of 12.3x on GovReport, while selecting considerably lower compression ratios of 1.4x and 1.9x on Multi-News and QMSum, respectively.

Among the evaluated methods, LongLLMLingua performs relatively well in this setting, maintaining strong ROUGE-L scores while reducing the input context. This is reasonable because LongLLMLingua is designed for general generation tasks, whereas EFFCOMP is specifically optimized for RAG-based QA. Overall, these results suggest that although EFFCOMP can be directly applied to long-generation tasks without additional training, its advantages are less pronounced outside the QA setting for which it was designed.

C.5 Additional Experiments with Recent Retrievers

To investigate whether EFFCOMP remains beneficial when combined with stronger retrievers and target LLMs, we conduct additional experiments using recent retrieval methods and a more capable target LLM. In particular, we examine whether EFFCOMP can further improve QA performance even when the underlying retrieval or generation component is improved.

Method	2WikiQA				NQ				TQA			
	$1/\tau$	Acc	$C_{0.5}$	C_1	$1/\tau$	Acc	$C_{0.5}$	C_1	$1/\tau$	Acc	$C_{0.5}$	C_1
Gemma-2-9B-IT												
No reranker (in-distribution)	**16.1x**	**29.5**	**0.34**	**0.45**	**23.6x**	**31.7**	**0.37**	**0.48**	**48.4x**	**78.9**	**0.82**	**0.87**
No reranker (out-of-distribution)	11.7x	28.9	**0.34**	0.44	9.8x	31.1	0.36	0.46	32.8x	**78.9**	**0.82**	**0.87**
BGE reranker (in-distribution)	**30.3x**	**25.4**	**0.30**	**0.40**	**36.4x**	**29.5**	**0.34**	**0.45**	**59.5x**	**80.1**	**0.83**	**0.88**
BGE reranker (out-of-distribution)	19.4x	25.0	0.29	**0.40**	14.1x	29.0	**0.34**	0.44	49.6x	79.5	**0.83**	**0.88**
Jina reranker (in-distribution)	**29.4x**	25.5	**0.30**	0.40	**42.0x**	**30.0**	**0.35**	**0.46**	**78.4x**	**80.1**	**0.83**	**0.88**
Jina reranker (out-of-distribution)	19.3x	**25.8**	**0.30**	**0.41**	18.8x	29.7	0.34	0.45	50.0x	79.6	**0.83**	**0.88**
GTE reranker (in-distribution)	**29.4x**	**25.4**	**0.30**	**0.40**	**42.3x**	**30.4**	**0.35**	**0.46**	**64.3x**	**80.5**	**0.84**	**0.89**
GTE reranker (out-of-distribution)	20.4x	25.0	0.29	**0.40**	19.2x	29.7	0.34	0.45	54.1x	79.8	0.83	0.88
Llama-3.1-8B-Instruct												
No reranker (in-distribution)	**16.1x**	**21.3**	**0.25**	**0.35**	**23.6x**	**27.9**	**0.33**	**0.43**	**48.4x**	73.5	0.77	**0.84**
No reranker (out-of-distribution)	11.7x	20.8	**0.25**	0.34	9.8x	26.8	0.31	0.41	32.8x	**74.3**	**0.78**	**0.84**
BGE reranker (in-distribution)	**30.3x**	**21.5**	**0.26**	**0.35**	**36.4x**	**28.1**	**0.33**	**0.44**	**59.5x**	**76.6**	**0.80**	**0.86**
BGE reranker (out-of-distribution)	19.4x	20.3	0.24	0.33	14.1x	27.5	0.32	0.42	49.6x	76.1	**0.80**	**0.86**
Jina reranker (in-distribution)	**29.4x**	**21.8**	**0.26**	**0.36**	**42.0x**	**28.9**	**0.34**	**0.45**	**78.4x**	**77.1**	**0.81**	**0.87**
Jina reranker (out-of-distribution)	19.3x	21.3	0.25	0.35	18.8x	28.2	0.33	0.44	50.0x	76.2	0.80	0.86
GTE reranker (in-distribution)	**29.4x**	**21.6**	**0.26**	**0.35**	**42.3x**	**29.6**	**0.34**	**0.45**	**64.3x**	**77.3**	**0.81**	**0.87**
GTE reranker (out-of-distribution)	20.4x	20.8	0.25	0.34	19.2x	28.3	0.33	0.44	54.1x	77.2	**0.81**	0.86

Table 6: Cross-dataset generalization of EFFCOMP using Gemma-2-9B-IT and Llama-3.1-8B-Instruct as target models, evaluated both with and without the document reranker. We report the mean compression ratio ($1/\tau$), accuracy (Acc), and C_β. The in-distribution rows correspond to training on each dataset individually, while the out-of-distribution rows correspond to training only on HQA and directly applying the model to other datasets. For all of these metrics, higher values are better. Bold numbers indicate the best performance in each setting.

We consider five retrieval approaches from two directions:

- **Improve embeddings for retrieval.** We replace Contriever-MS MARCO, used in our paper, with:
  - **Voyage 4 Lite**[15], a commercial embedding model.
  - **F2LLM-v2-8B** (Zhang et al., 2026), strongest open-source retriever according to the 2026 model on MTEB benchmark (Muennighoff et al., 2023).
- **Improve retrieval pipelines.** We consider three approaches that use more sophisticated retrieval pipelines:
  - **LLMChainExtractor** (Verma, 2024) with GPT-4.1 Nano, which uses an LLM to extract query-relevant passages or sentences from each retrieved document.
  - **EmbeddingsRedundantFilter** (Verma, 2024) with text-embedding-3-large and a similarity threshold of 0.7, which removes semantically redundant documents based on their embeddings.
  - **MultiQueryRetriever**[16], which generates multiple query reformulations to improve document recall.

We evaluate these methods on 1,000 HQA examples using two target LLMs, Gemma-2-9B-IT (Team et al., 2024) and Claude Sonnet 4.5[17]. The results are shown in Table 9. The setting marked with * corresponds to the retrieval configuration mainly used throughout our paper. For LLMChainExtractor, the retrieved documents are already highly compressed; therefore, we do not apply EFFCOMP to avoid further compression.

The results in Table 9 show that EFFCOMP remains effective with recent retrieval methods. First, EFFCOMP with the standard retriever used in our paper outperforms all advanced retrievers without EFFCOMP. For example, with Gemma-2-9B-IT, the standard setting (Contriever-MS MARCO) achieves 35.7 accuracy, compared with 32.6 for F2LLM-v2-8B and 29.3 for Voyage 4 Lite without EFFCOMP. With Claude Sonnet 4.5, the corresponding accuracy is 43.6, compared with 43.4 and

[15]https://huggingface.co/voyageai/voyage-4-lite

[16]https://reference.langchain.com/python/langchain-classic/retrievers/multi_query/MultiQueryRetriever

[17]https://platform.claude.com/docs/en/models/sonnet-4-5/overview

Method	HQA				2WikiQA				NQ				TQA			
	$1/\tau$	Acc	$C_{0.5}$	C_1	$1/\tau$	Acc	$C_{0.5}$	C_1	$1/\tau$	Acc	$C_{0.5}$	C_1	$1/\tau$	Acc	$C_{0.5}$	C_1
Raw Documents	1.0x	24.9	0.00	0.00	1.0x	16.7	0.00	0.00	1.0x	25.3	0.00	0.00	1.0x	69.7	0.00	0.00
Reranker-only methods																
BGE	3.0x	29.6	0.33	0.41	3.0x	19.2	0.22	0.30	3.0x	26.6	0.30	0.38	3.0x	75.3	0.73	0.71
Jina	3.0x	29.2	0.33	0.41	3.0x	19.3	0.22	0.30	3.0x	28.1	0.32	0.40	3.0x	75.6	0.74	0.71
GTE	3.0x	29.0	0.33	0.40	3.0x	19.2	0.22	0.30	3.0x	28.4	0.32	0.40	3.0x	75.9	0.74	0.71
Compression frameworks																
LLMLingua2	5.4x	21.4	0.25	0.34	5.5x	17.3	0.21	0.29	5.3x	22.1	0.26	0.35	5.4x	68.8	0.71	0.75
DAC	5.1x	19.9	0.23	0.32	5.0x	16.6	0.20	0.27	5.2x	19.4	0.23	0.31	5.2x	67.5	0.70	0.74
CPC	5.6x	26.7	0.31	0.40	5.9x	17.7	0.21	0.29	5.2x	25.8	0.30	0.39	5.4x	74.2	0.76	0.78
LongLLMLingua	5.1x	27.7	0.32	0.41	5.1x	18.8	0.22	0.30	5.1x	24.4	0.28	0.37	5.1x	73.6	0.75	0.77
$\text{Recomp}_{\text{Extractive}}$	29.2x	24.0	0.28	0.38	29.3x	16.6	0.20	0.28	29.8x	22.3	0.26	0.36	32.0x	69.7	0.74	0.81
$\text{Recomp}_{\text{Abstractive}}$	**132.1x**	27.8	0.32	0.43	**126.9x**	21.3	0.25	0.35	**48.3x**	26.3	0.31	0.41	**139x**	74.3	0.78	0.85
CompAct	48.7x	30.5	0.35	**0.47**	54.7x	18.8	0.22	0.32	44.2x	27.9	0.33	0.43	50.2x	75.9	0.79	0.86
Our work: EFFCOMP																
No reranker	26.8x	29.6	0.34	0.45	11.7x	20.8	**0.25**	0.34	9.8x	26.8	0.31	0.41	32.8x	74.3	0.78	0.84
BGE reranker	39.5x	**31.3**	**0.36**	**0.47**	30.3x	21.5	0.25	0.35	36.4x	28.1	0.33	0.44	59.5x	76.6	0.80	0.86
Jina reranker	38.5x	31.0	**0.36**	**0.47**	29.4x	**21.8**	**0.26**	**0.36**	42.0x	28.9	**0.34**	**0.45**	78.4x	77.1	**0.81**	**0.87**
GTE reranker	40.4x	31.0	**0.36**	**0.47**	29.4x	21.6	**0.26**	0.35	42.3x	**29.6**	**0.34**	**0.45**	64.3x	**77.3**	**0.81**	**0.87**

Table 7: Results on four QA benchmarks. We report the mean of compression ratio ($1/\tau$), accuracy (Acc), C_β for Llama-3.1-8B-Instruct as the target model. For all of these metrics, higher values are better. The best results in each column are marked in bold.

Method	GovReport		Multi-News		QMSum	
	$1/\tau$	ROUGE-L	$1/\tau$	ROUGE-L	$1/\tau$	ROUGE-L
Raw documents	1.0x	18.8	1.0x	**18.2**	1.0x	**19.2**
LLMLingua2	5.2x	16.8	**5.7x**	16.8	5.5x	17.6
DAC	6.0x	17.4	4.9x	17.1	**6.3x**	17.2
CPC	5.1x	17.2	4.7x	16.3	5.2x	17.7
LongLLMLingua	4.6x	**19.0**	3.1x	18.1	3.8x	19.0
EFFCOMP fixed	5.0x	17.4	5.3x	16.9	4.9x	18.1
EFFCOMP adaptive	**12.3x**	16.9	1.4x	17.8	1.9x	18.8

Table 8: Generalization of EFFCOMP to long-generation tasks from LongBench. The EFFCOMP model is trained on HQA and directly applied to GovReport, Multi-News, and QMSum without task-specific training. We report the mean compression ratio ($1/\tau$) and ROUGE-L score. Higher values are better for both metrics. Bold numbers indicate the best performance in each setting.

40.8, respectively. Second, EFFCOMP provides substantial gains even when combined with stronger retrieval methods. For example, F2LLM-v2-8B improves from 32.6 to 36.9 with EFFCOMP for Gemma-2-9B-IT, and from 43.4 to 47.0 for Claude Sonnet 4.5. Similarly, Voyage 4 Lite with MultiQueryRetriever improves from 28.6 to 36.9 for Gemma-2-9B-IT and from 35.1 to 44.3 for Claude Sonnet 4.5 when combined with the full EFFCOMP pipeline. These results demonstrate that the benefits of EFFCOMP are complementary to improvements in the retrieval pipeline rather than being limited to a particular retriever.

C.6 Hallucinations in Compressed Contexts

As discussed in the Introduction, we hypothesize that EFFCOMP produces fewer hallucinations than abstractive methods due to its extractive nature. To evaluate this, we use natural language inference (NLI) on the HQA test set. For each example, the original context is treated as the premise and the compressed context as the hypothesis. We then obtain NLI predictions from three commercial LLMs (Gemini 2.5 Flash Lite[18], GPT-4.1 nano[19], and Claude 3 Haiku (Anthropic)). Table 10 re-

[18]https://ai.google.dev/gemini-api/docs/models/gemini-2.5-flash-lite

[19]https://developers.openai.com/api/docs/models/gpt-4.1-nano

Retriever	Step 1 (BGE)	Step 2	$1/\tau$	Acc	$C_{0.5}$	C_1
Gemma-2-9B-IT						
Contriever-MS MARCO (Top 30)*	Top 10	✓	41.1x	35.7	0.41	0.52
Voyage 4 Lite (Top 6)	–	–	5.0x	29.3	0.34	0.43
Voyage 4 Lite (Top 6)	–	✓	37.3x	30.2	0.35	0.46
Voyage 4 Lite (Top 30)	Top 6	✓	51.6x	35.3	0.40	0.52
F2LLM-v2-8B (Top 6)	–	–	5.0x	32.6	0.37	0.46
F2LLM-v2-8B (Top 6)	–	✓	40.6x	32.9	0.38	0.49
F2LLM-v2-8B (Top 30)	Top 6	✓	50.4x	**36.9**	**0.42**	**0.54**
Contriever-MS MARCO (Top 30) + LLMChainExtractor	–	–	14.5x	31.5	0.36	0.47
Contriever-MS MARCO (Top 30) + EmbeddingsRedundantFilter	–	–	1.6x	28.9	0.30	0.33
Contriever-MS MARCO (Top 30) + EmbeddingsRedundantFilter	Top 10	✓	34.0x	29.5	0.34	0.45
Voyage 4 Lite + MultiQueryRetriever (Top 10)	–	–	6.1x	28.6	0.33	0.43
Voyage 4 Lite + MultiQueryRetriever (Top 10)	–	✓	50.9x	30.0	0.35	0.46
Voyage 4 Lite + MultiQueryRetriever (Dynamic)	Top 10	✓	**90.7x**	**36.9**	**0.42**	**0.54**
Claude Sonnet 4.5						
Contriever-MS MARCO (Top 30)*	Top 10	✓	41.1x	43.6	0.49	0.60
Voyage 4 Lite (Top 6)	–	–	5.0x	40.8	0.45	0.54
Voyage 4 Lite (Top 6)	–	✓	37.3x	39.3	0.45	0.56
Voyage 4 Lite (Top 30)	Top 6	✓	51.6x	44.0	0.49	0.61
F2LLM-v2-8B (Top 6)	–	–	5.0x	43.4	0.48	0.56
F2LLM-v2-8B (Top 6)	–	✓	40.6x	43.0	0.48	0.60
F2LLM-v2-8B (Top 30)	Top 6	✓	50.4x	**47.0**	**0.52**	**0.64**
Contriever-MS MARCO (Top 30) + LLMChainExtractor	–	–	14.5x	43.0	0.48	0.59
Contriever-MS MARCO (Top 30) + EmbeddingsRedundantFilter	–	–	1.6x	24.4	0.26	0.30
Contriever-MS MARCO (Top 30) + EmbeddingsRedundantFilter	Top 10	✓	34.0x	38.4	0.44	0.55
Voyage 4 Lite + MultiQueryRetriever (Top 10)	–	–	6.1x	35.1	0.40	0.49
Voyage 4 Lite + MultiQueryRetriever (Top 10)	–	✓	50.9x	37.8	0.43	0.55
Voyage 4 Lite + MultiQueryRetriever (Dynamic)	Top 10	✓	**90.7x**	44.3	0.50	0.61

Table 9: Performance of recent retrieval methods with and without EFFCOMP on 1,000 HQA examples. Step 1 denotes BGE reranking, where "Top k" indicates the number of documents retained by the reranker. Step 2 denotes the sentence selector, where ✓indicates that sentence selection is applied. * denotes the retrieval configuration mainly used throughout our paper.

ports the number of entailment, neutral, and contradiction predictions where we treat both neutral and contradiction predictions as hallucinations, since neither category is fully supported by the reference context. Across all models, EFFCOMP (BGE reranker) consistently yields more entailment cases and fewer contradiction cases than both CompAct and Recomp-Abstractive, indicating improved faithfulness. EFFCOMP achieves an average hallucination rate of 25%, compared to 34% for CompAct and 53% for Recomp-Abstractive. These results confirm that EFFCOMP introduces substantially fewer hallucinations than abstractive approaches.

C.7 Inference Latency And Computational Cost

Table 11 reports the compression latency, the target LLM reading latency, and the total latency, all averaged from 500 HQA examples with a batch size of 1. EFFCOMP substantially reduces LLM reading time (up to 55.4%), while introducing only a small compression overhead compared to no compression. As a result, it reduces total latency up to a 53.3% when using the Jina reranker. EFFCOMP is among the fastest methods, together with Recomp-Extractive

To further evaluate computational efficiency, we measure the computational cost of compression in terms of Teraflops (TFLOPS) under the same evaluation setting (HQA, 500 samples, batch size = 1), as shown in Table 12. Since EFFCOMP employs a lightweight sentence-level compressor, it incurs substantially lower computational cost than most existing compression frameworks. In particular, EFFCOMP with the Jina reranker requires only 1.2 TFLOPs, consisting of 0.9 TFLOPs for reranking and 0.3 TFLOPs for sentence-level compression, which is the lowest compression cost among the compared methods. These results demonstrate that EFFCOMP achieves both low inference latency and low computational cost, making it suitable for prac-

Model	Method	Entailment	Neutral	Contradiction
Gemini 2.5 Flash Lite	CompAct	81.2%	15.8%	3.0%
	Recomp-Abstractive	45.7%	35.4%	18.9%
	EFFCOMP (BGE)	**86.0%**	**11.4%**	**2.6%**
GPT-4.1 nano	CompAct	50.6%	1.3%	48.1%
	Recomp-Abstractive	43.7%	0.6%	55.8%
	EFFCOMP (BGE)	**61.8%**	**0.2%**	**38.0%**
Claude 3 Haiku	CompAct	66.7%	24.8%	8.5%
	Recomp-Abstractive	50.3%	21.0%	28.7%
	EFFCOMP (BGE)	**77.1%**	**18.7%**	**4.2%**

Table 10: NLI label counts on HQA (7,405 examples). Higher entailment and lower neutral/contradiction indicate better faithfulness and fewer hallucinations.

tical, user-facing applications.

Method	Comp (s)	Read (s)	Total (s)
Raw documents	–	1.84	1.84
LLMLingua2	0.28	0.84	1.12
DAC	1.46	0.79	2.25
CPC	0.26	0.81	1.07
LongLLMLingua	2.31	0.90	3.21
Recomp-Extractive	0.19	0.74	0.93
Recomp-Abstractive	2.84	**0.71**	3.55
CompAct	16.30	0.72	17.02
EFFCOMP (BGE)	0.08	0.82	0.90
EFFCOMP (Jina)	**0.04**	0.82	**0.86**
EFFCOMP (GTE)	0.12	0.82	0.94

Table 11: Compression latency, LLM read latency, and total latency per example averaged over 500 HQA examples using Gemma-2-9B-IT. Bold values denote the lowest metric in each column.

Method	Compression TFLOPs
LLMLingua2	3.3
DAC	35.2
CPC	8.4
LongLLMLingua	252.2
$\text{Recomp}_{\text{Extractive}}$	2.7
CompAct	65.3
EFFCOMP (BGE)	2.1(Reranker) + 0.3(Sentence) = 2.5
EFFCOMP (Jina)	0.9(Reranker) + 0.3(Sentence) = **1.2**
EFFCOMP (GTE)	1.1(Reranker) + 0.3(Sentence) = 1.4

Table 12: Compression computational cost in TFLOPs averaged over 500 HQA examples with a batch size of 1. Lower values indicate lower computational cost.

C.8 Memory Usage

We evaluate the memory footprint of the compressor and the downstream LLM during inference. Memory usage is reported as the average peak GPU memory during the forward pass. For Recomp-Abstractive, per-example measurement is not supported, so we report the peak memory across 500 examples. Table 13 reports the memory consumption of all methods. Our method achieves the second-lowest compressor memory usage among all frameworks while maintaining competitive memory consumption during downstream LLM inference.

C.9 Different Architectures for Sentence Selector

We further evaluate our method with different encoder architectures to study whether compression performance depends heavily on the backbone model. In addition to ModernBERT (149M), we experiment with mGTE-base[20] (305M) and mmBERT-mini[21] (Marone et al., 2025) (140M). For each architecture, we train the model for three epochs and select the checkpoint with the best reward score for evaluation. We evaluate all models on the HQA and NQ benchmarks using the same training and evaluation setup. Table 14 shows that all architectures achieve similar QA performance in terms of accuracy score, while producing substantially different compression ratios. Interestingly, the larger model does not consistently provide the best trade-off between compression and QA quality in our setting. For example, GTE achieves the highest compression ratio at 31.8x while maintaining performance comparable to ModernBERT. Meanwhile, the lightweight mmBERT-mini model also performs competitively in terms of C_β despite its smaller size. These results suggest that architecture choice mainly affects compression efficiency rather than downstream QA performance.

[20] https://huggingface.co/Alibaba-NLP/gte-multilingual-base

[21] https://huggingface.co/jhu-clsp/mmBERT-small

Method	Compressor Memory (MB)	LLM Inference Memory (MB)
Raw documents	–	11,874.37
LLMLingua2	2,588.44	7,158.68
DAC	23,161.87	6,890.60
CPC	2,969.61	6855.91
LongLLMLingua	16,129.06	6,935.18
Recomp-Extractive	**945.46**	6,619.45
Recomp-Abstractive	5,934.80	8,208.77
Compact	14,090.37	**6,580.15**
EFFCOMP (BGE)	1,252.01	7,202.59
EFFCOMP (Jina)	1,242.26	7,141.47
EFFCOMP (GTE)	1,323.48	7,178.40

Table 13: Peak GPU memory usage (in MB) for each compressor and the downstream LLM (Gemma-2-9B-IT). Memory is measured during the model forward pass. For Recomp-Abstractive, the peak across 500 examples is reported due to the lack of per-sample measurement.

C.10 Effect of Improved Phase I Labels

We further investigate whether the quality of the supervision used in Phase I can be improved by leveraging the gold supporting sentences provided in a multi-hop dataset. Instead of assigning positive labels based on whether a sentence contains the ground-truth answer, we assign a positive label when its character-level overlap with a gold supporting sentence exceeds 50%. We denote this alternative intermediate finetuning strategy as Phase I*. We evaluate three training configurations on the validation set: Phase I* only, Phase I* followed by Phase II, and our original Phase I followed by Phase II. The results are reported in Table 15.

The results show that using supporting-sentence supervision in Phase I leads to more aggressive compression. Phase I* alone achieves the highest compression rate of 86.4%, although its accuracy is substantially lower than the two configurations that include Phase II. Adding Phase II to Phase I* increases the accuracy from 82.3% to 88.6%, while maintaining a relatively high compression rate of 83.3%. This demonstrates the benefit of Phase II in improving QA performance even when stronger supporting-sentence supervision is used during Phase I. Comparing Phase I* + Phase II with our original Phase I + Phase II, the two labeling strategies exhibit different trade-offs. Phase I* + Phase II achieves a higher compression rate (83.3% vs. 75.6%) and a higher C_1 score (0.86 vs. 0.83), whereas our original approach achieves higher QA accuracy (91.6% vs. 88.6%) and a slightly higher $C_{0.5}$ score (0.88 vs. 0.87). These results suggest that supporting-sentence annotations can provide useful supervision when greater compression is desired, but do not necessarily improve QA accuracy.

Furthermore, Phase I* relies on gold supporting-sentence annotations, which may not be available in other QA datasets. In contrast, our original Phase I constructs labels using only the ground-truth answers, making the training procedure applicable to a broader range of QA datasets without requiring additional sentence-level annotations.

C.11 Effect of Sentence Representation Tokens

Our sentence selector uses a dedicated `[SEN]` token to represent each sentence and make sentence-level retention decisions. Although the standard `[CLS]` token can serve a similar role, we use `[SEN]` based on the intuition that `[CLS]` is generally used to represent the entire input sequence, whereas `[SEN]` provides an explicit representation for each individual sentence. To examine this design choice, we replace each `[SEN]` token with `[CLS]` and evaluate both variants on the validation set. The results are reported in Table 16.

The results show that both tokens can effectively support sentence-level selection, but they exhibit different trade-offs between QA accuracy and compression. Using `[CLS]` achieves a higher accuracy of 93.5%, compared with 91.6% for `[SEN]`. However, `[SEN]` achieves a substantially higher compression rate of 75.6%, compared with 67.4% for `[CLS]`. When considering both objectives jointly, `[SEN]` achieves higher $C_{0.5}$ and C_1 scores of 0.88 and 0.83, respectively, compared with 0.87 and 0.78 for `[CLS]`. These results indicate that although `[CLS]` favors QA accuracy, the dedicated `[SEN]` token provides a better overall balance between preserving QA performance and achieving effective compression.

Method	HQA				NQ			
	$1/\tau$	Acc	$C_{0.5}$	C_1	$1/\tau$	Acc	$C_{0.5}$	C_1
Raw Documents	1.0x	34.6	0.00	0.00	1.0x	30.1	0.00	0.00
Reranker-only methods								
BGE reranker	3.0x	35.2	0.39	0.46	3.0x	29.6	0.33	0.41
Jina reranker	3.0x	34.6	0.38	0.46	3.0x	30.6	0.34	0.42
GTE reranker	3.0x	34.7	0.38	0.46	3.0x	30.6	0.34	0.42
Compression frameworks								
LLMLingua2	5.4x	25.0	0.29	0.38	5.3x	23.8	0.28	0.37
DAC	5.1x	24.7	0.29	0.38	5.2x	22.7	0.27	0.35
CPC	5.6x	30.5	0.35	0.45	5.2x	28.0	0.32	0.42
LongLLMLingua	5.1x	31.8	0.36	0.46	5.1x	25.7	0.30	0.39
Recomp-Extractive	29.2x	25.6	0.3	0.41	29.8x	22.8	0.27	0.37
Recomp-Abstractive	**132.1x**	27.9	0.33	0.44	48.3x	27.6	0.32	0.43
CompAct	48.7x	31.2	0.36	0.47	44.2x	27.8	0.32	0.43
Our work: EFFCOMP *(Modernbert-base)*								
No reranker	26.8x	**37.4**	**0.43**	**0.54**	23.6x	31.7	**0.37**	**0.48**
BGE reranker	39.5x	36.3	0.42	0.53	36.4x	29.5	0.34	0.45
Jina reranker	38.5x	35.8	0.41	0.52	42.0x	30.0	0.35	0.46
GTE reranker	40.4x	35.5	0.41	0.52	42.3x	30.4	0.35	0.46
Our work: EFFCOMP *(mmBert-mini)*								
No reranker	11.0x	37.1	0.42	0.53	17.9x	**32.1**	**0.37**	**0.48**
BGE reranker	27.8x	36.0	0.41	0.52	22.3x	29.6	0.34	0.45
Jina reranker	27.5x	35.6	0.41	0.52	25.2x	30.6	0.35	0.46
GTE reranker	27.9x	35.5	0.41	0.52	24.3x	30.4	0.35	0.46
Our work: EFFCOMP *(GTE-base)*								
No reranker	31.8x	36.7	0.42	0.53	27.8x	29.2	0.34	0.45
BGE reranker	49.0x	35.7	0.41	0.52	42.8x	29.0	0.34	0.45
Jina reranker	48.5x	35.4	0.41	0.52	52.0x	29.3	0.34	0.45
GTE reranker	49.7x	35.2	0.40	0.52	**52.7x**	28.8	0.34	0.45

Table 14: Results on two QA benchmarks for Gemma-2-9B-IT. We report compression ratio ($1/\tau$), accuracy, and C_β. Higher is better. Best values are in bold.

Method	Acc	$1-\tau$	$C_{0.5}$	C_1
Phase I*	82.3	**86.4** (%)	0.83	0.84
Phase I* + Phase II	88.6	83.3 (%)	0.87	**0.86**
Phase I + Phase II	**91.6**	75.6 (%)	**0.88**	0.83

Table 15: Comparison of different Phase I labeling strategies on the validation set. Phase I* uses gold supporting sentences to construct the intermediate finetuning labels. We report accuracy (Acc), compression rate, and C_β. Higher values are better, and best values are in bold.

Method	Acc	$1-\tau$	$C_{0.5}$	C_1
`[CLS]`	**93.5**	67.4 (%)	0.87	0.78
`[SEN]`	91.6	**75.6** (%)	**0.88**	**0.83**

Table 16: Comparison between `[CLS]` and `[SEN]` tokens for sentence-level representation on the validation set. We report accuracy (Acc), compression rate, and C_β. Higher values are better, and best values are in bold.

C.12 Additional Ablation Study on Different Finetuning Strategies

In addition to RQ4 in Section 5, we conducted an ablation study on 2WikiQA, NQ, and TQA, evaluating four training strategies: reinforcement learning (RL) only, supervised learning (SL) only with the best trial memory, and the hybrid approach of EFFCOMP that combines both approaches (RL+SL and SL+RL). The results are displayed in Figure 5. We observed similar patterns in HQA, NQ, and TQA, where intermediate finetuning led to higher accuracy. Specifically, the hybrid strategy (RL+SL) often achieved higher validation accuracy in early epochs, while having a relatively lower compression rate. In contrast, the RL only strategy prioritized the compression rate while sacrificing the accuracy. Meanwhile, the SL only strategy usually achieved higher validation accuracy than the RL one but with a lower and stable compression rate

across epochs. For 2WikiQA, where intermediate finetuning did not substantially improve accuracy, the trend of the hybrid (RL+SL) approach is similar to the other three datasets, whereas the trends of the RL only and SL only strategies are the opposite of those in the other three datasets.

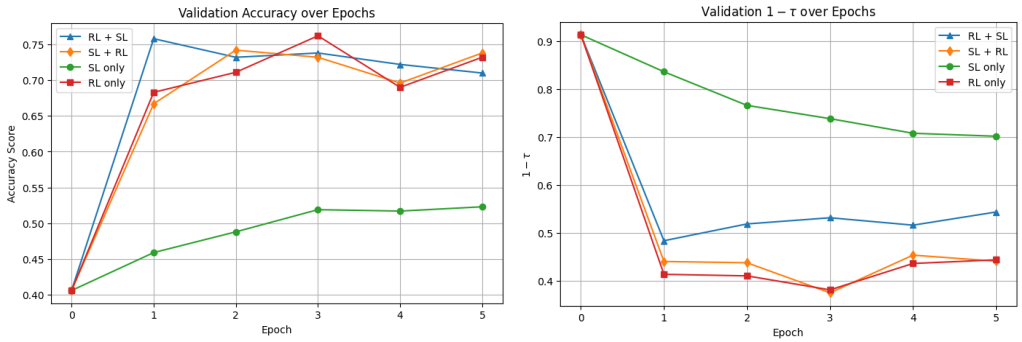
D The Use of Large Language Models

We leveraged ChatGPT and Gemini to enhance grammar, refine phrasing, and optimize word choice across the paper. We also used them to help format tables and figures in LaTeX. The typical prompts we used with the LLMs were concise instructions such as "Proofread:" or "polish this text", which direct the models to refine grammar and phrasing without altering the technical content.

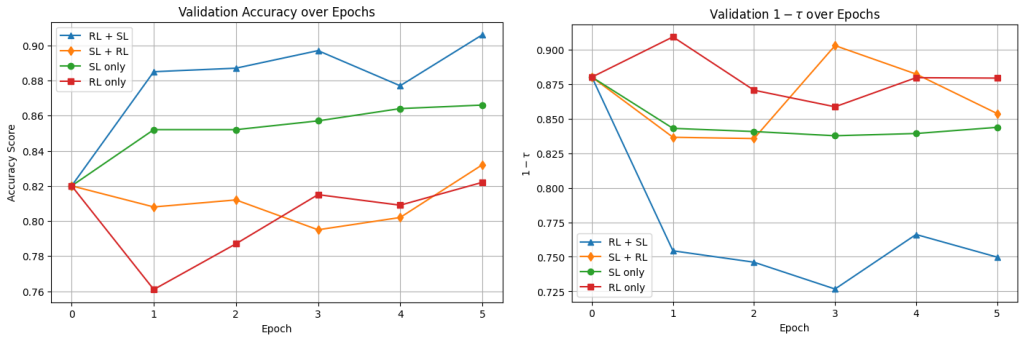
E Licenses

Our experiments use several publicly available datasets. HotpotQA is licensed under CC BY-SA 4.0. Natural Questions, 2WikiMultiHopQA, and TriviaQA are licensed under the Apache License 2.0. LongBench is licensed under the MIT License. All intermediate finetuning models used in our experiments, including Gemma-2-9B-IT and LLaMA-3.1-8B-Instruct, are used in accordance with their respective licenses for research purposes.

(A) 2WikiMultiHopQA



(B) Natural Questions



(C) TriviaQA

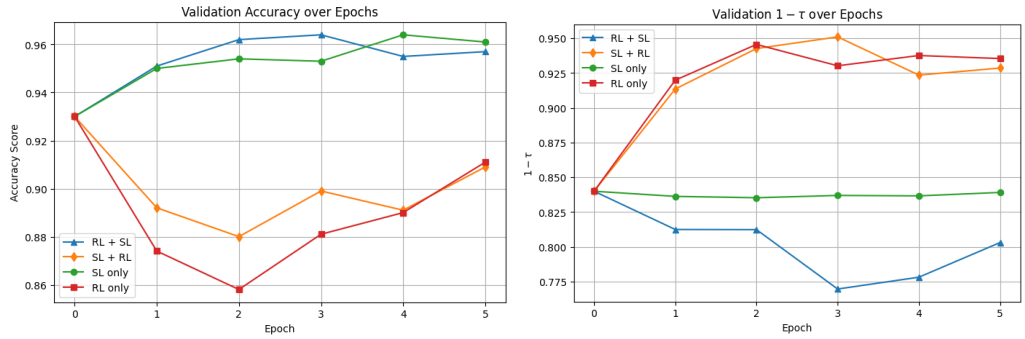


Figure 5: Ablation study of different training strategies for EFFCOMP on three QA datasets. The plots compare (left) validation accuracy and (right) compression rate $1 - \tau = 1 - |\mathbb{C}|/|\mathbb{D}|$ during validation across five epochs. For both metrics, the higher values are better.