

Enhancing RAG Efficiency with Adaptive Context Compression

Shuyu Guo*
Shandong University
Qingdao, China
guoshuyu225@gmail.com

Shuo Zhang
Bloomberg
London, United Kingdom
szhang611@bloomberg.net

Zhaochun Ren†
Leiden University
Leiden, The Netherlands
z.ren@liacs.leidenuniv.nl

Abstract

Retrieval-augmented generation (RAG) enhances large language models (LLMs) with external knowledge but incurs significant inference costs due to lengthy retrieved contexts. While context compression mitigates this issue, existing methods apply fixed compression rates—over-compressing simple queries or under-compressing complex ones. We propose Adaptive Context Compression for RAG (ACC-RAG), a framework that dynamically adjusts compression rates based on input complexity, optimizing inference efficiency without loss of accuracy. ACC-RAG combines a hierarchical compressor (for multi-granular embeddings) with a context selector to retain minimal sufficient information, akin to human skimming. Evaluated on Wikipedia and five QA datasets, ACC-RAG outperforms fixed-rate methods and unlocks >4× faster inference versus standard RAG while maintaining or improving accuracy.

1 Introduction

Large Language Models (LLMs) are pre-trained on massive datasets, encoding vast knowledge within billions of parameters. While they excel at many tasks, their parametric knowledge often falls short for knowledge-intensive applications. Retrieval-Augmented Generation (RAG) addresses this limitation by extending the model’s knowledge boundaries through external context retrieval (Gao et al., 2023). However, integrating lengthy retrieved contexts into prompts increases inference costs and may exceed LLMs’ context window limits (Chevalier et al., 2023).

Context compression mitigates this issue by transforming long contexts into shorter input sequences. Existing methods fall into two categories: (1) Lexical-based compression, which reduces input length by preserving key tokens (Wang

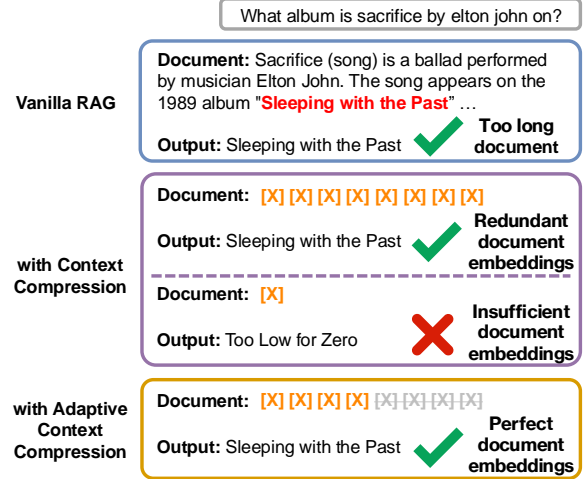


Figure 1: An example of a retrieval-augmented model with different context compression methods.

et al., 2023b) or generating summaries (Xu et al., 2024a); and (2) Embedding-based compression, which encodes text into dense embeddings (Ge et al., 2024) for inference. Embedding-based methods have been proven more efficient and effective (Cheng et al., 2024), typically employing a compressor trained in two phases: first through pre-training (e.g., via autoencoding or language modeling) to preserve contextual information (Chevalier et al., 2023), followed by fine-tuning (e.g., with instruction follow-up or self-distillation) to adapt to downstream tasks (Rau et al., 2024). Existing embedding-based approaches fix the compression rate (i.e., token-to-embedding ratio), leading to trade-offs: high rates risk losing essential information, while low rates retain redundancies (Figure 1). Additionally, inconsistent evaluation benchmarks—varying training data scales and tasks across baselines—hinder fair comparisons.

We propose an adaptive context compression framework for RAG that dynamically adjusts the number of compressed embeddings during inference. The framework decouples offline compression (fixed-rate hierarchical embeddings) from on-

*Work performed during a visit to Leiden University.

†Corresponding Author.

line selection (dynamic embedding feeding, halted once sufficient context is accumulated), mimicking human selective reading. We establish a unified benchmark (Wikipedia corpus + five QA datasets) to ensure fair evaluation. Our method outperforms other compression techniques in effectiveness and efficiency, matches or exceeds standard RAG accuracy on four datasets with $4\times$ faster inference, and demonstrates significant potential for adaptive compression via compressor analysis.

In summary, our contributions include: (1) A novel adaptive context compression framework for RAG, improving inference efficiency while maintaining retrieval augmentation benefits. (2) Training strategies for a hierarchical compressor and adaptive selector, enabling multi-granularity compressed embedding generation and automatic context selection. (3) A unified benchmark for context compression methods, resolving evaluation biases from inconsistent training data and task setups in prior works. (4) Experimental results showing our method as the best under context compression, with comparable results to regular RAG but with improved efficiency. (5) Further analysis highlighting the potential of adaptive context compression.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) improves model performance across tasks (Gao et al., 2023; Asai et al., 2023), including language modeling (Min et al., 2023; Wang et al., 2023a), question answering (Lewis et al., 2020; Shi et al., 2024; Xu et al., 2024b; Xiong et al., 2024), machine translation (Khandelwal et al., 2021; Cheng et al., 2022), code generation (Zhang et al., 2023b,a), and more. Various approaches have been proposed, including end-to-end optimization of model and retriever (Guu et al., 2020), enhanced integration with non-parametric knowledge (Borgeaud et al., 2022; Cheng et al., 2023a), improved alignment between model and retriever (Shi et al., 2024; Lin et al., 2024), and the integration of a self-reflection mechanism (Cheng et al., 2023b; Asai et al., 2024). However, most focus on improving effectiveness, neglecting efficiency degradation from incorporating external knowledge.

2.2 Context Compression

Context compression aims to shorten model input text, improving inference speed. It’s a key

solution to mitigate efficiency degradation in RAG systems (Li et al., 2024). Methods fall into two categories: lexical-based, which reduce input context length by extracting tokens (Wang et al., 2023b) or summarizing context (Xu et al., 2024a), and embedding-based, which encode the context into embeddings to replace textual input (Mu et al., 2023). Embedding-based methods currently outperform lexical-based due to their flexible information storage. Embedding-based methods have shown strong performance. Bulatov et al. (2022) segmented long context into chunks and encoding their information into memory tokens. Wingate et al. (2022) introduced context compression and applied it to toxicity reduction. Mu et al. (2023) proposed using gist tokens to generalize context compression. Chevalier et al. (2023) compressed long contexts into summary vectors. Ge et al. (2024) employed autoencoding and language modeling objectives to pretrain compressors. Cheng et al. (2024) projected retrieval embeddings into context embeddings to achieve extreme compression. Rau et al. (2024) jointly optimized the compressor and decoder, extending the compression capability to multiple documents. Notably, existing methods rely on static compression strategies that cannot adapt to varying question difficulty. Additionally, differences in training data scales and task mixtures across studies impede fair comparison. Our work resolves both limitations.

3 Adaptive Context Compression for RAG

A RAG framework typically consists of a retriever \mathcal{R} and a decoder \mathcal{D} . The retriever builds a search index I containing dense representations of all documents from a collection. For a given query Q , \mathcal{R} retrieve the top- k relevant documents $D = \{d_1, d_2, \dots, d_k\}$ by matching Q ’s embedding against I via similarity search. The query Q and documents D are concatenated and fed into the decoder \mathcal{D} to generate a response R . This increases the input size to the decoder, reducing inference efficiency as $|D| \gg |Q|$.

Existing embedding-based compression methods employ a fixed-rate compressor to convert documents into embedding sequences, which are subsequently fed into the decoder for answer generation instead of the original documents. However, this fixed-ratio approach lacks adaptability to query complexity. The ACC-RAG framework

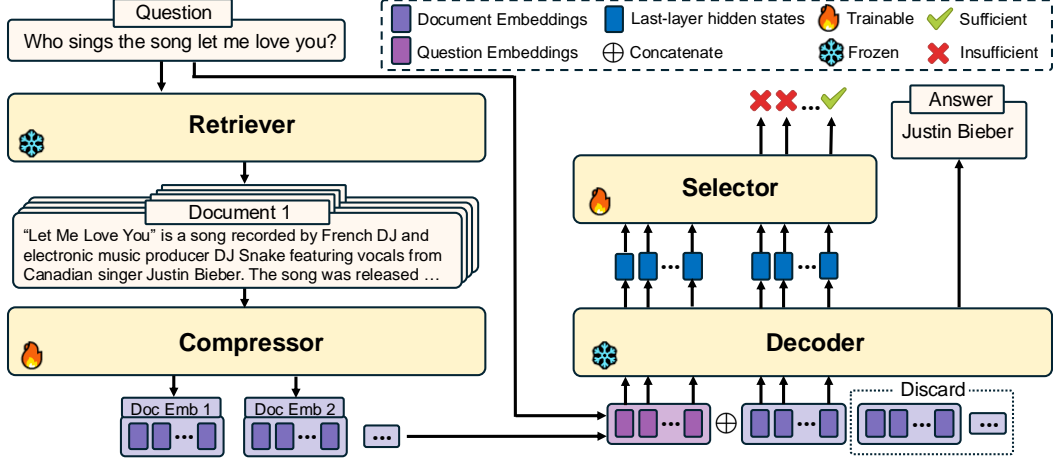


Figure 2: Overview of Adaptive Context Compression for RAG pipeline. The workflow is as follow: Context Retrieval → Hierarchical Compression → Adaptive Context Selection → Response Generation.

introduces a *hierarchical compressor* \mathcal{C} and an *adaptive selector* \mathcal{S} working together to support dynamic compression. The former encodes documents into multi-granular document embeddings, enabling variable information density across segments. The latter progressively feeds embeddings into the decoder and halts once sufficient context is reached, effectively controlling input sequence length. This mechanism allows variable-length decoder inputs, thereby realizing dynamic compression. The full workflow is illustrated in Figure 2.

3.1 Hierarchical Compressor

The compressor \mathcal{C} processes each document d_i into a multi-granular embedding sequence. Specifically, the input text is concatenated with $m_i = \lfloor L_i/\tau \rfloor$ special compression tokens, where L_i is the document length and τ is a fixed compression rate. These compression tokens are encoded, and their final-layer embeddings form the compressed sequence E_i :

$$E_i = \mathcal{C}(d_i) = [e_i^{(1)}, \dots, e_i^{(m_i)}] \quad (1)$$

3.1.1 Fix-rated Preprocessing

The compressor employs a fixed compression ratio: higher ratios yield shorter compressed sequences with reduced information retention. The entire compression process operates offline, with all documents pre-processed into their embedding sequences for instant access during retrieval.

3.1.2 Training Strategy

The compressor is trained through a widely adopted two-stage process: pretraining and fine-tuning. The

former enables the compressor to preserve maximal contextual information, while the latter facilitates adaptation to downstream tasks. During pre-training, the compressor learns to encode text into embeddings through two tasks (Ge et al., 2024): (1) auto-encoding, where the decoder reconstructs the original text from the compressed embeddings alone, and (2) language modeling, where the decoder generates text continuations conditioned on the embeddings. Formally, given a task instruction Q , compressed sequence E_c , and target response $R = \{r_1, \dots, r_n\}$ (either original text or continuation), both tasks optimize the negative log-likelihood (NLL) loss:

$$\mathcal{L}(\theta_{\mathcal{C}}) = - \sum_{t=1}^n \log P_{\theta_{\mathcal{D}}}(r_t | Q, E_c, r_{<t}) \quad (2)$$

Only the compressor parameters $\theta_{\mathcal{C}}$ are updated, while the decoder parameters $\theta_{\mathcal{D}}$ remain frozen. During fine-tuning, we employ a self-distillation task (Allen-Zhu and Li, 2023), where the original RAG model acts as a teacher to preserve decoder capabilities. Given a query Q , context C , and target response R , the teacher distribution is $P_{\text{teacher}}(r_t | Q, C, r_{<t})$, and the student distribution is $P_{\text{student}}(r_t | Q, E_c, r_{<t})$. The training minimizes the KL divergence between these distributions:

$$\mathcal{L} = D_{\text{KL}}(P_{\text{teacher}} \parallel P_{\text{student}}) \quad (3)$$

Notably, the fine-tuning approach avoids instruction-following tasks, preventing the compressed embeddings from altering the decoder’s original generation pattern.

3.1.3 Multi-granularity Compression

In addition to encoding useful information, the compressor must also learn hierarchical information organization. We define granularity b as the starting position of an embedding sequence. The compressed sequence should exhibit varying information densities across subsequences truncated at different granularity levels. To achieve multi-granular compression, we optimize across multiple granularities during both pretraining and fine-tuning. Formally, given a training granularity sequence $\mathcal{B} = \{b_1, \dots, b_k\}$, the training objective combines reconstruction errors at all granularities:

$$\mathcal{L}(\theta_C) = - \sum_{b \in \mathcal{B}} \sum_{t=1}^n \log P_{\theta_D}(r_t \mid Q, E_c^{1:b}, r_{<t}) \quad (4)$$

where $E_c^{1:b} = [e_1, \dots, e_b]$. This method ensures \mathcal{C} allocates crucial information early and maintains complementary details in later positions.

3.2 Adaptive Selector

The adaptive selector \mathcal{S} dynamically controls decoder input by progressively accumulating and verifying context embeddings during inference. The query is encoded into a query embedding sequence $E_q = \mathcal{D}_{\text{embed}}(q)$, while the context embedding sequence is obtained by concatenating all compressed document sequences in retrieval order: $E_c = \text{Concat}(E_1, \dots, E_k)$. The selector iterates through granularity levels from the inference granularity sequence $\mathcal{B} = \{b_1, \dots, b_k\}$. At each step t :

1. Extract the subsequence $E_{1:b_t} = [e_1, \dots, e_{b_t}]$ from E_c , then concatenate with E_q to construct the current input:

$$X_t = \text{Concat}(E_q, E_{1:b_t}) \quad (5)$$

2. Feed X_t into the decoder \mathcal{D} to obtain the last layer hidden states:

$$H_t = \mathcal{D}(X_t) \quad (6)$$

3. Feed H_t into the selector \mathcal{S} to verify if the context information is sufficient:

$$\mathcal{S}(H_t) = \begin{cases} 1 & \text{sufficient} \\ 0 & \text{insufficient} \end{cases} \quad (7)$$

This process continues until $\mathcal{S}(H_t) = 1$ or $t = k$. The final input X is determined based on the termination state:

$$X = \begin{cases} X_t & \mathcal{S}(H_t) = 1 \\ \text{Concat}(E_q, E_c) & t = k \end{cases} \quad (8)$$

3.2.1 Training Data Synthesis

The selector \mathcal{S} is trained on synthesized decision tuples. Given an instruction I , context C , and gold response R , for each granularity $b \in \mathcal{B}$, the hidden states H_b and generated responses R_b are obtained through inference with the decoder, while the label y_b is derived by evaluating R_b against R :

$$\begin{aligned} H_b &= \mathcal{D}(I, E_c^{1:b}) \\ R_b &= \mathcal{D}_{\text{gen}}(I, E_c^{1:b}) \\ y_b &= \text{eval}(R, R_b) \end{aligned} \quad (9)$$

The training data x_i consists of pairs:

$$x_i = \{(H_b, y_b) \mid \forall b \in \mathcal{B}\} \quad (10)$$

A fixed decoder requires only a unified selector, as the generation mode remains the same.

3.2.2 Reinforcement Learning

The selector is trained using reinforcement learning, optimized via policy gradient with trajectory-based rewards. During inference, we stop adding context embeddings and generate the response once the selector predicts "sufficient". For each episode with granularities $b_1 < \dots < b_k$:

$$\begin{aligned} \text{Trajectory: } \tau &= (s_1, a_1), \dots, (s_T, a_T) \\ \text{State: } s_t &= \text{Concat}(H_I, H_{b_t}) \\ \text{Action: } a_t &= \begin{cases} 1 & \text{sufficient} \\ 0 & \text{continue} \end{cases} \end{aligned} \quad (11)$$

where H_I and H_{b_t} refer to the hidden states of the instruction I and the truncated embedding sequence $E_c^{1:b_t}$, respectively.

The trajectory terminates when the action is 1 or all granularities are traversed. The reward is:

$$R(\tau) = \begin{cases} +1 & \text{if } \exists t : a_t = 1 \wedge y_{b_t} = 1 \\ -1 & \text{otherwise} \end{cases} \quad (12)$$

The policy π_θ parameters are updated via the REINFORCE algorithm (Williams, 1992):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] \quad (13)$$

where π_θ shares parameters with \mathcal{S} .

4 Experimental Setup

4.1 Datasets

To ensure a fair comparison of different compression methods, we train and evaluate all methods on

unified datasets. For the retrieval corpus, we use the Wikipedia corpus from Dec. 20, 2018, standardized by Karpukhin et al. (2020) using the preprocessing from Chen et al. (2017). The corpus is split into multiple, disjoint text blocks of 128 tokens as documents, resulting in 21,015,324 documents in the end. For fine-tuning, we conduct experiments on five commonly used open-domain QA benchmarks, namely Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), WebQuestions (WQ) (Berant et al., 2013), CuratedTREC (TREC) (Baudis and Sedivý, 2015), and SQuAD v1.1 (Rajpurkar et al., 2016). All question-answer pairs are ensured to have supporting documents available in the retrieval corpus. For pretraining, we select all supporting documents from the training set of QA and randomly sample additional documents from the retrieval corpus, resulting in a total of 1 million documents for training.

4.2 Baselines

We compare our method with two categories of compression approaches: *plug-in methods* and *full-tuning methods*. *Plug-in methods* train additional compressors without modifying the decoder parameters, resulting in low training costs and preserving the original performance of the decoder. Baseline methods in this category include:

ICAE (Ge et al., 2024): Pretrains the compressor with same task as ours but finetunes through instruction-following. We implemented ICAE models with multiple compression rates for evaluation.

xRAG (Cheng et al., 2024): Maps retrieval embeddings into compression embeddings for extreme compression. Performs only auto-encoding during pretraining and combines instruction-following with self-distillation during fine-tuning.

Full-tuning methods adopt an end-to-end approach, jointly training the compressor and decoder for better alignment and performance, but at increased training costs and changes to the original decoder performance. Baseline methods in this category include:

Autocompressor (Chevalier et al., 2023): Trains the compressor only with language modeling. We use the pretraining-only COCOM model as a simplified version.

COCOM (Rau et al., 2024): Similar to ICAE but jointly optimizes the compressor and decoder.

Our method is implemented solely as a plug-in method, offering lower cost, better scalability, and strong performance. We use the same backbone

LLM for all methods and apply identical LoRA parameters for LoRA-based fine-tuning.

4.3 Evaluation Metrics

We evaluate model performance from two dimensions: effectiveness and efficiency. For effectiveness, we use the Match (M) metric, which measures whether the reference answer appears in the model’s generated output. We do not use the Exact Match (EM) metric due to verbose responses from LLMs affecting its calculation. As for Efficiency, we use First Token Inference Time (FTIT) as the metric. Unlike widely-adopted Total Inference Time (TIT) (Cheng et al., 2024; Rau et al., 2024), FTIT focuses on the time to the first token, isolating the impact of compression methods on efficiency rather than generation performance.

4.4 Implementation Details

We use Mistral-7B-Instruct (Jiang et al., 2023) as the backbone LLM for all methods, with a decoding temperature of 0 for deterministic generation. We apply LoRA (Hu et al., 2022) to the model as the compressor, disabling it during inference as the decoder. All experiments are implemented using PyTorch and Transformers. The default retrieval model is ColBERT (Santhanam et al., 2022), using the top-5 ranked documents for fine-tuning and evaluation. Pretraining is done on single documents while fine-tuning is performed on five documents. For selector training, we generate 15,000 training samples and 2,000 test samples from the QA train set. The selector employs a 4-layer/4-head transformer encoder with 256D projection, followed by a 2-layer MLP classifier, incorporating segment embeddings for instruction-context differentiation and contextual granularity in final features. More details are provided in Appendix C.

5 Experimental Results

5.1 Main Results

The main results for ACC-RAG are presented in Table 1. Our ACC-RAG method, trained and inferred with a granularity sequence of [1, 32], balances effectiveness and efficiency. Detailed results for other ACC-RAG configurations are in Section 5.4. The experimental results demonstrate that ACC-RAG significantly (Paired t-test, $p < 0.05$) outperforms all compression baselines in M score while maintaining superior efficiency. Notably, under our unified benchmark, most methods (e.g.,

Table 1: The main results between ACC-RAG and other compression methods. CR stands for Compression Rate. For each dataset, two columns represent Match (M) and First Token Inference Time (FTIT), with the highest M bolded and the second-highest underlined. * indicates statistical non-significance ($p>0.05$) with respect to ACC-RAG.

Method	CR	Datasets											
		NQ		TriviaQA		WQ		TREC		SQuAD		Average	
LLM	-	34.79	180	20.93	391	43.36*	172	34.29*	235	20.92	210	30.86	238
RAG	-	44.49	3264	23.82	3529	41.78	3334	<u>33.43</u>	3374	47.34	3356	38.17	3371
Full-Tuning Methods													
AutoCompressor	$\times 4$	24.13	1007	14.81	1242	28.15	1015	23.49	1085	15.79	1057	21.27	1081
COCOM	$\times 4$	26.73	1007	18.83	1242	23.92	1015	31.56	1085	25.60	1057	25.33	1081
Plug-in Methods													
xRAG	$\times 128$	5.93	269	8.61	485	12.60	265	17.00	324	7.77	301	10.38	328
ICAIE	$\times 128$	23.30	269	18.11	485	23.57	265	30.55	324	18.47	301	22.8	328
	$\times 16$	27.04	440	19.23	667	23.97	436	31.41	502	23.95	476	25.12	504
	$\times 4$	27.53	1007	19.63	1242	24.51	1015	31.41	1085	27.58	1057	26.13	1081
ACC-RAG(ours)	adaptive	<u>41.11</u>	630	<u>23.33</u>	878	<u>42.91</u>	620	<u>33.43</u>	689	<u>35.37</u>	666	<u>35.23</u>	697

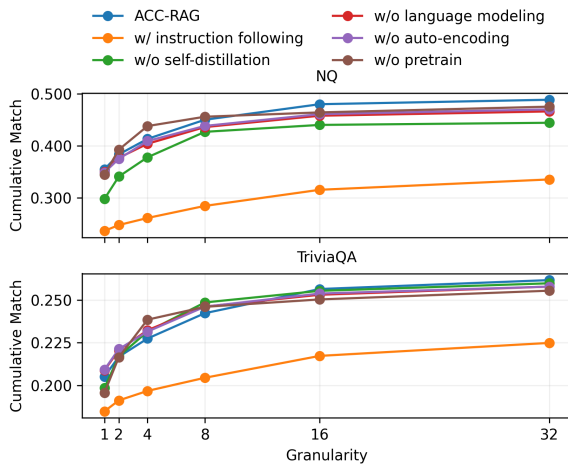


Figure 3: Ablation on different training strategies.

xRAG, COCOM) exhibit lower effectiveness compared to their original reports, primarily due to their reliance on larger-scale training data and task-specific tuning. In contrast, ACC-RAG achieves a balanced trade-off between effectiveness and efficiency: compared to direct generation, it significantly improves M scores on NQ, TriviaQA, and SQuAD ($p<0.05$) with minimal drop on WQ and TREC ($p>0.05$). Meanwhile, it matches or exceeds vanilla RAG’s accuracy on four datasets while reducing FTIT by over $4\times$.

5.2 Compressor Analysis

In this section, we investigate factors influencing compressor performance. Experiments are conducted under the top-1 retrieval setting for analysis. We adopt cumulative M scores across different granularities as the evaluation metric rather than M scores at a single granularity, as the compressor compresses hierarchical information into position-aware embedding sequences for dynamic aggrega-

tion. Notably, cumulative M scores assume a perfect selector capable of accurately choosing the optimal granularity for each instruction. While this does not exist, this section focuses solely on evaluating the compression capacity of the compressor.

5.2.1 Compression Rate Analysis

The result of different compression rates for compressor is shown in Appendix D. It can be observed that compressors with smaller compression rates achieve higher final cumulative M scores, demonstrating stronger information encoding capabilities. However, compressors with higher compression rates exhibit accelerated M score accumulation, where the information density becomes more concentrated at the smaller granularity despite total information loss.

5.2.2 Training Strategy Ablation

The result of ablation of different training strategies for the compressor is shown in Figure 3. Our training strategy achieves the highest cumulative M scores on two datasets, demonstrating its effectiveness. In ablation experiments, replacing the self-distillation task with instruction following significantly degraded compression performance. This is likely because instruction following guides the compressor to encode information into embeddings that do not align with the decoder’s original generation distribution, impairing the decoder’s generation ability. This could explain the suboptimal performance of previous works. Without pretraining, the compressor exhibits faster M-score accumulation at smaller granularities but slower progression at larger granularities. Both language modeling and auto-encoding tasks contribute to performance gains, as their individual removal degrades perfor-

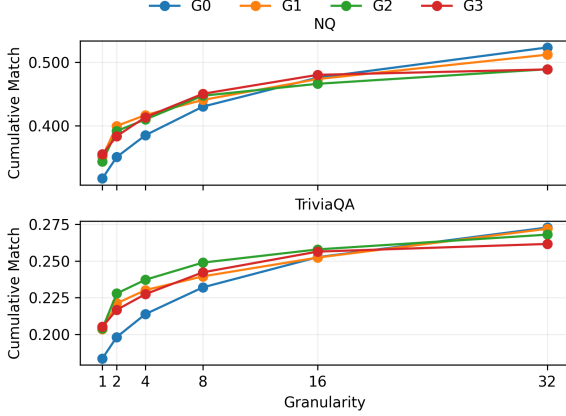


Figure 4: Results of different granularity sequences for compressor. In the legend, G0-G3 denote [32], [1, 32], [1, 2, 8, 32], [1, 2, 4, 8, 16, 32] respectively.

mance. The self-distillation proves critical for NQ performance but negligible for TrivialQA, suggesting its crucial role in certain scenarios.

5.2.3 Granularity Sequence Selection

We investigate the impact of different training granularity sequences on compressor performance during training. We present results for the inference granularity sequence [1, 2, 4, 8, 16, 32], with additional results in Appendix E. As shown in Figure 4, with the granularity sequence becoming increasingly dense from G0 to G3, the overall trend indicates a slight decrease in the final accumulated M scores but accelerated M score accumulation at smaller granularities. This suggests that even with a fixed compression rate training approach (i.e., the single granularity sequence G0) as in previous work, the compressor retains hierarchical encoding capabilities. Training with additional smaller granularities effectively guides the compressor to encode information in the compressed embeddings at earlier positions. This allows the decoder to access sufficient information with fewer embeddings, thereby improving inference efficiency. We also observe that the granularity-based inference method significantly outperforms the single granularity approach under ideal conditions. The final accumulated M scores on both datasets notably surpass vanilla RAG, demonstrating the immense potential of our ACC-RAG framework.

5.3 Selector Analysis

5.3.1 Architecture Ablation

Our ablation study (Appendix F) reveals that RL serves as the critical component for sequence-level performance, while supervised learning exhibits

Table 2: The results of different compressors w/ or w/o a selector during inference. For each dataset, two columns represent Match (M) and First Token Inference Time (FTIT). G0-G3 denote different granularity sequences.

Compressors	Selector	Datasets			
		NQ		TriviaQA	
ACC-G0	✗	40.19	1007	23.21	1242
ACC-G0	✓	41.33	786	23.38	984
ACC-G1	✗	40.86	1007	23.62	1242
ACC-G1	✓	41.11	630	23.33	878
ACC-G2	✗	39.94	1007	23.20	1242
ACC-G2	✓	39.39	622	23.07	895
ACC-G3	✗	41.27	1007	23.32	1242
ACC-G3	✓	40.17	631	22.58	878

error propagation leading to sequence-level failure due to overemphasis on individual classification. In addition, the attention mechanism also plays a key role in enhancing the selector, while the introduction of segment embeddings and granularity slightly further improves performance.

5.3.2 Inference Verification

To validate the effectiveness of the selector during actual inference, we compare different compressors with and without the selector using the inference granularity sequence [1, 32]. As shown in Table 2, ACC-G0 and ACC-G1 with the selector outperformed the versions without the selector on nearly all datasets with superior efficiency (with FTIT reduced by over 20%). ACC-G2 and ACC-G3 with the selector also maintain comparable performance to their counterparts without the selector, while achieving over 30% FTIT reduction.

5.4 Framework Evaluation

The results of combinations of compressors trained with different granularity sequences and inference selection strategies are shown in Table 3. We observe that, for the same compressor, sparser inference granularity sequences yield higher M scores but require more context embeddings. This is due to sparser sequences reducing classification errors by the selector, while potentially missing more precise termination points. Among all compressors, ACC-G0+G1 achieves the highest average M score, while ACC-G1+G1 reaches a similar M score with less than 60% of the context embeddings, demonstrating the effectiveness of the multi-granularity training strategy in hierarchical information encoding. Additionally, training the compressor and selection strategy with denser granularity sequences further enhances efficiency, as exemplified by ACC-

Table 3: The results of combinations of different compressors and selection strategies. G0-G3 denote different granularity sequences. For each dataset, two columns represent Match (M) and Average Context Embedding Counts, with the best metrics bolded and the second-best underlined.

Compressors	Selection Strategies	Datasets											
		NQ		TriviaQA		WQ		TREC		SQuAD		Average	
ACC-G0	G3	41.00	102	23.34	105	42.32	97	33.86	102	35.06	109	35.12	103
	G2	41.11	104	23.42	107	42.27	99	<u>34.15</u>	103	35.09	111	35.21	105
	G1	<u>41.33</u>	107	<u>23.38</u>	111	42.52	104	34.01	108	35.13	114	35.27	109
ACC-G1	G3	41.41	<u>56</u>	23.33	<u>58</u>	42.22	<u>56</u>	33.14	<u>57</u>	35.16	<u>57</u>	35.05	<u>57</u>
	G2	<u>41.33</u>	62	23.24	63	<u>42.67</u>	62	33.14	63	<u>35.26</u>	62	35.13	62
	G1	41.11	64	23.33	64	42.91	64	33.43	64	35.37	64	<u>35.23</u>	64
ACC-G2	G3	39.42	47	22.69	50	42.37	48	32.71	48	35.22	49	34.48	48
	G2	39.20	<u>56</u>	23.11	60	42.57	58	33.29	58	34.97	58	34.63	58
	G1	39.39	63	23.07	65	42.57	64	33.72	64	35.15	65	34.78	64
ACC-G3	G3	40.06	63	22.67	62	42.27	64	<u>34.15</u>	64	31.51	63	34.13	63
	G2	40.17	64	22.58	64	42.32	64	34.44	64	31.52	64	34.21	64
	G1	40.17	64	22.58	64	42.32	64	34.44	64	31.52	64	34.21	64

Table 4: The results of RAG performance between vanilla RAG and ACC-RAG. For each dataset, two columns represent resilience and boost rate, with the highest score bolded and the second-highest underlined.

Method	Datasets			
	NQ		TriviaQA	
RAG	72.98	28.78	82.06	8.97
ACC-RAG-G0	76.18	<u>22.94</u>	83.22	<u>7.91</u>
ACC-RAG-G1	77.95	21.67	<u>83.56</u>	7.75
ACC-RAG-G2	74.82	20.69	83.22	7.52
ACC-RAG-G3	<u>77.79</u>	20.31	83.95	6.72

G2+G3, which optimized inference efficiency at the expense of a slight M score decrease.

5.5 Case Study

We present a representative case in Appendix G, which demonstrates the outstanding performance of ACC-RAG in both efficiency and effectiveness.

5.6 Scalability Evaluation

We observe an excellent trade-off between effectiveness and efficiency with Llama3-3B-Instruct and Llama3-8B-Instruct (results in Appendix H). ACC-RAG achieves results comparable to RAG on both models and datasets (with <10% M score difference), while improving speed by 4 to 5 times (with FTIT reduced by 76% to 83%), fully showcasing the scalability of the method.

5.7 RAG Integration Analysis

We evaluate the RAG performance using the resilience rate and the boost rate proposed in (Cheng et al., 2024). The resilience rate quantifies the proportion of correct decoder responses retained

post-retrieval, while the boost rate measures the percentage of initially incorrect responses corrected through retrieval augmentation. As shown in Table 4, RAG exhibits a higher boost rate, while ACC-RAG demonstrates a higher resilience rate. As the training granularity sequence for compressor becomes denser, the overall trend is an increase in the boost rate and a decrease in the resilience rate.

5.8 OOD Analysis

We evaluate the Out of Distribution(OOD) performance of our method(results in Appendix I), demonstrating its exceptional generalization ability to unseen supporting documents and unseen queries from different domains.

5.9 Additional Computation Analysis

Our approach incurs additional computational overhead during inference due to the introduced selector. Analysis in Appendix J demonstrates the favorable trade-off between controlled computational cost and substantial performance gains.

6 Conclusion

In this paper, we propose Adaptive Context Compression for RAG, a framework that dynamically adjusts compressed embeddings required during inference. The framework effectively improves inference efficiency while maintaining the quality benefits of retrieval. Experimental results under standardized benchmark show that our method achieves the best results under context compression and comparable or superior accuracy with over 4× inference efficiency compared to RAG.

Limitations

Despite the promising results demonstrated in this paper, there are several limitations to our framework.

- We have only conducted a preliminary exploration of the selector and its performance is currently the largest bottleneck in the entire framework. Improving the prediction accuracy of the selector is key to further unlocking the potential of adaptive text compression.
- Our work focuses on adaptive context selection, lacking experimental analysis under controlled conditions, which remains a promising direction for future research.
- We have trained the compressor and selector separately, without further exploring the potential of joint training. This may hinder the alignment between the two modules, potentially reducing the overall performance of the framework.
- Due to computational constraints, we have not explored the performance of ACC-RAG on larger models and longer texts, which may limit its applicability in certain scenarios.

Ethics Statement

The research conducted in this paper centers around the improvement of efficiency in retrieval augmented generation. Our framework accelerates the inference of retrieval augmented models by compressing long context into compact embeddings and dynamically selecting them.

We acknowledge the importance of the ACM Code of Ethics and affirm our adherence to it. We ensure that this work is compatible with the provided code. All data used in this study is obtained from existing benchmarks, while all models used are publicly available, thus ensuring a high level of transparency and reproducibility in our experimental procedure.

We have made every effort to ensure that our research does not harm individuals or groups, nor does it involve any form of deception or potential misuse of information.

References

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. [Towards understanding ensemble, knowledge distillation and](#)

[self-distillation in deep learning](#). In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. [Retrieval-based language models and applications](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 41–46. Association for Computational Linguistics.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. [Self-rag: Learning to retrieve, generate, and critique through self-reflection](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Petr Baudis and Jan Sedivý. 2015. [Modeling of the question answering task in the yodaqa system](#). In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 6th International Conference of the CLEF Association, CLEF 2015, Toulouse, France, September 8-11, 2015, Proceedings*, volume 9283 of *Lecture Notes in Computer Science*, pages 222–228. Springer.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. [Semantic parsing on freebase from question-answer pairs](#). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. [Improving language models by retrieving from trillions of tokens](#). In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. 2022. [Recurrent memory transformer](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. [Reading Wikipedia to answer open-domain questions](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879,

- Vancouver, Canada. Association for Computational Linguistics.
- Xin Cheng, Shen Gao, Lemaou Liu, Dongyan Zhao, and Rui Yan. 2022. [Neural machine translation with contrastive translation memories](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 3591–3601. Association for Computational Linguistics.
- Xin Cheng, Yankai Lin, Xiuying Chen, Dongyan Zhao, and Rui Yan. 2023a. [Decouple knowledge from parameters for plug-and-play language modeling](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14288–14308, Toronto, Canada. Association for Computational Linguistics.
- Xin Cheng, Di Luo, Xiuying Chen, Lemaou Liu, Dongyan Zhao, and Rui Yan. 2023b. [Lift yourself up: Retrieval-augmented text generation with self-memory](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. [xRAG: Extreme context compression for retrieval-augmented generation with one token](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- 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, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3829–3846. Association for Computational Linguistics.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. [Retrieval-augmented generation for large language models: A survey](#). *CoRR*, abs/2312.10997.
- Tao Ge, Jing Hu, 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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. [Retrieval augmented language model pre-training](#). In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [Lora: Low-rank adaptation of large language models](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. [Mistral 7b](#). *CoRR*, abs/2310.06825.
- Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. [Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. 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 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.
- Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. [Nearest neighbor machine translation](#). In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: a benchmark for question answering research](#). *Trans. Assoc. Comput. Linguistics*, 7:452–466.
- Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich K  ttler, Mike Lewis, Wen-tau Yih, Tim Rockt  schel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. 2024. [Prompt compression for large language models: A survey](#). *CoRR*, abs/2410.12388.
- Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. [RA-DIT](#):

- retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2023. [Nonparametric masked language modeling](#). In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2097–2118. Association for Computational Linguistics.
- Jesse Mu, Xiang Li, and Noah D. Goodman. 2023. [Learning to compress prompts with gist tokens](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [Squad: 100, 000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.
- David Rau, Shuai Wang, Hervé Déjean, and Stéphane Clinchant. 2024. [Context embeddings for efficient answer generation in RAG](#). *CoRR*, abs/2407.09252.
- Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. [Colbertv2: Effective and efficient retrieval via lightweight late interaction](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3715–3734. Association for Computational Linguistics.
- Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. [REPLUG: retrieval-augmented black-box language models](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 8371–8384. Association for Computational Linguistics.
- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. [FEVER: a large-scale dataset for fact extraction and verification](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.
- Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023a. [Augmenting language models with long-term memory](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md. Rizwan Parvez, and Graham Neubig. 2023b. [Learning to filter context for retrieval-augmented generation](#). *CoRR*, abs/2311.08377.
- Ronald J. Williams. 1992. [Simple statistical gradient-following algorithms for connectionist reinforcement learning](#). *Mach. Learn.*, 8:229–256.
- 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, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5621–5634. Association for Computational Linguistics.
- Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. [Benchmarking retrieval-augmented generation for medicine](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 6233–6251. Association for Computational Linguistics.
- Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024a. [RECOMP: improving retrieval-augmented lms with context compression and selective augmentation](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Zhentao Xu, Mark Jerome Cruz, Matthew Guevara, Tie Wang, Manasi Deshpande, Xiaofeng Wang, and Zheng Li. 2024b. [Retrieval-augmented generation with knowledge graphs for customer service question answering](#). In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 2905–2909. ACM.
- 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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.
- Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. 2023a. [Repocoder: Repository-level code completion through iterative retrieval and generation](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*.

pages 2471–2484. Association for Computational Linguistics.

Xiangyu Zhang, Yu Zhou, Guang Yang, and Taolue Chen. 2023b. [Syntax-aware retrieval augmented code generation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 1291–1302. Association for Computational Linguistics.

A Datasets

We use the dataset version processed by DPR¹ including retrieval corpus and five QA datasets, which is licensed by *CC-BY-NC 4.0*. All datasets are English-language resources within the question answering (QA) domain. The datasets we used inherit from them while maintaining consistency in QA intent. We ensure that the dataset is free from harmful content or information leakage through a sampling approach. The detailed statistics of the dataset are presented in Table 5. Notably, based on whether the supporting documents are present in the pretraining dataset, the development set of the fine-tuning data is further divided into seen and unseen subsets.

Datasets	Train	Dev	Test
Natural Questions	58880	3195	3320
TriviaQA	60413	2102	4658
WebQuestions	2474	77	201
CuratedTREC	1125	32	84
SQuAD v1.1	70096	6985	936

Table 5: Number of QA pairs in each dataset across Train, Dev, and Test splits. The two columns of Dev denote the seen and the unseen subsets respectively.

B Models

We use Mistral-7B-Instruct-v0.2², Llama3.2-3B-Instruct³ and Llama3.1-8B-Instruct⁴ as our backbone LLM. The Mistral model is licensed by *Apache-2.0* and two Llama models are licensed by *Meta Llama3 Community License*. The retriever we used is ColBERT-v2⁵ with *MIT License*.

C Implementation Details

In Table 6 and Table 7, we list the hyperparameters for compressor pretraining and fine-tuning. These parameters are primarily inherited from prior work (e.g., learning rate, LoRA configuration), as the same backbone LLM and similar training tasks are used. Additionally, all methods are trained under

¹<https://github.com/facebookresearch/DPR>

²<https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2>

³<https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct>

⁴<https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct>

⁵<https://github.com/stanford-futuredata/ColBERT>

Hyperparameter	Assignment
optimizer	AdamW
learning rate	$1e^{-4}$
lr scheduler type	linear
warmup ratio	0.03
weight decay	0.0
epochs	1
flash attention	True
batch size	8
gradient accumulation steps	4
LoRa r	128
LoRa alpha	32
LoRa dropout	0.05
LoRa bias	None
gpu	$1 \times \text{A100 80G}$
max sequence length	320

Table 6: Hyperparameters for compressor pretraining.

the same configuration for a fair comparison, ensuring no intentional optimization bias towards our approach. For efficient training, at each training step, we randomly sample only one granularity from granularity sequence and one task to optimize during pretraining. In our configuration, we pretrain the compressor with roughly 20 hours and finetune with roughly 50 hours. For selector training, we list the hyperparameters in Table 8. We primarily search the hyperparameters of learning rates and find that a range between $2e^{-4}$ and $1e^{-4}$ yields the best results. Additionally, we adjusted the number of attention heads and layers in the transformer encoder. We observed that increasing both parameters beyond 4 provided no improvement, while reducing them below 4 resulted in a slight decline in performance. The entire training process took approximately 1 hour, and we ultimately selected the model with the highest sequence accuracy on the test set. All experiments are implemented using PyTorch⁶ and Hugging Face Transformers⁷.

D Analysis on Compression Rate of Compressor

We investigate the impact of different compression rates on compressor performance during training. The results are shown in Figure 5. We train the model with a single granularity to facilitate analysis, where the granularity is the count of compressed embeddings obtained by compression at a

⁶<https://pytorch.org/>

⁷<https://github.com/huggingface/transformers>

Hyperparameter	Assignment
optimizer	AdamW
learning rate	$1e^{-5}$
lr scheduler type	linear
warmup ratio	0.03
weight decay	0.0
epochs	5
flash attention	True
batch size	2
gradient accumulation steps	4
LoRa r	128
LoRa alpha	32
LoRa dropout	0.05
LoRa bias	None
gpu	1×A6000 48G
max sequence length	1024

Table 7: Hyperparameters for compressor fine-tuning.

Hyperparameter	Assignment
optimizer	AdamW
learning rate	$1e^{-4}$
lr scheduler type	linear
warmup ratio	0.0
weight decay	0.0
epochs	50
batch size	128
gradient accumulation steps	1
gpu	1×A6000 48G

Table 8: Hyperparameters for selector training.

specific compression rate.

E Analysis on Granularity Sequences of Compressor

The results of different granularity sequences for compressor under inference granularity sequence [1, 2, 8, 32] and [1, 32] are shown in Figure 6 and Figure 7. The conclusions are consistent with Section 5.2.3. We observe that the final accumulated M scores exceeds the vanilla RAG even with only two inference granularities.

F Ablation Study of Selector

We investigate the impact of different components of selector for classification performance. The default setting of ACC-Selector is described in Section 4.4. We adopt the sequence classification accuracy as evaluation metric, verifying whether the selector’s first prediction of "sufficient" is correct.

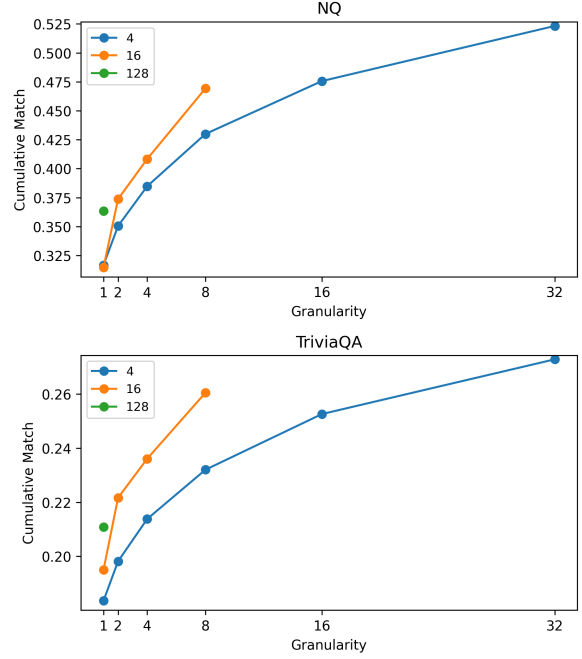


Figure 5: Ablation on different compression rates for compressor.

We report the results in test set in Table 10.

Methods	Sequence Accuracy
ACC-Selector	70.75
w/o RL	64.25
w/o attention	66.30
w/o segment embeddings	69.35
w/o granularity	69.50

Table 10: Ablation on selector.

G Representative Case

To provide a more intuitive understanding of the effectiveness of ACC-RAG, we present a representative case in Table 13. In this case, direct generation and ACC-RAG-G2 with a fixed compression rate of 16 fail to answer the question. However, our method ACC-RAG-G2 with inference granularity sequence [1, 32] not only provides the correct answer but also uses the fewest context tokens compared to ACC-RAG-G2 with a fixed compression rate of 4 and RAG.

H Scalability Analysis

To investigate whether ACC-RAG is applicable to other models, we apply our approach to both Llama3-3B-Instruct and Llama3-8B-Instruct models and report the results shown in Table 11.

Table 9: The OOD performance of unseen supporting documents. The highest Match(M) and lowest First Token Inference Time(FTIT) is bolded.

Method	Datasets						
	NQ	TriviaQA	WQ	TREC	SQuAD	Average	Average FTIT
RAG	69.97	27.44	49.75	39.29	59.62	49.21	3268
ICAE($\times 4$)	61.66	26.26	50.25	35.71	47.97	44.37	1052
ACC-RAG(ours)	63.34	26.56	50.75	39.29	51.60	46.31	673

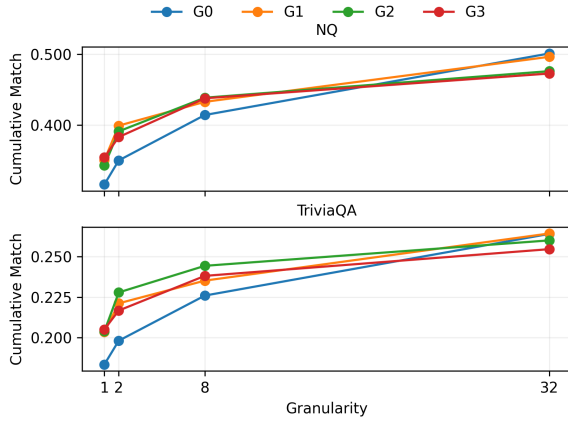


Figure 6: Results of different granularity sequences for compressor under inference granularity sequence [1, 2, 8, 32]. In the legend, G0-G3 denote [32], [1, 32], [1, 2, 8, 32], [1, 2, 4, 8, 16, 32] respectively.

Table 11: The results of ACC-RAG in Llama3 for scalability study. For each dataset, two columns represent Match (M) and First Token Inference Time (FTIT).

Model	Method	Datasets			
		NQ		TriviaQA	
Llama3-3B	-	36.37	65	15.33	101
	RAG	40.72	1037	22.26	966
	ACC-RAG	38.76	180	18.62	225
Llama3-8B	-	39.72	139	20.56	235
	RAG	44.43	2169	23.64	2089
	ACC-RAG	41.32	380	21.19	489

I OOD Analysis

To validate the OOD capability of our method, we conduct evaluations from two perspectives: (1) unseen supporting documents and (2) unseen queries from different domains. The former is evaluated on the unseen Devset where each case utilizes a ground-truth supporting document absent from the pretraining dataset. Results in Table 9 demonstrate that our method outperforms the strongest baseline in both effectiveness and efficiency. Compared to Vanilla RAG, our approach achieves over 4 \times faster inference speed while maintaining comparable accuracy across three datasets, with superior performance observed on two datasets. The

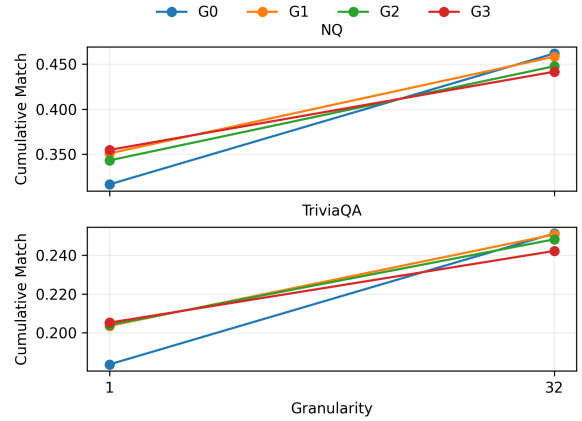


Figure 7: Results of different granularity sequences for compressor under inference granularity sequence [1, 32]. In the legend, G0-G3 denote [32], [1, 32], [1, 2, 8, 32], [1, 2, 4, 8, 16, 32] respectively.

Table 12: The OOD performance on unseen queries from different domains. For each dataset, two columns represent Match (M) and First Token Inference Time (FTIT).

Method	Datasets			
	HotpotQA		FEVER	
LLM	33.70	290	74.44	293
RAG	44.96	3313	83.60	3325
ICAE($\times 4$)	40.96	1114	79.62	1120
ACC-RAG(ours)	41.98	689	81.55	693

latter evaluation employs additional dataset from two distinct tasks (HotpotQA (Yang et al., 2018) and FEVER (Thorne et al., 2018)). As shown in Table 12, our method again demonstrates superior performance compared to strongest baseline while maintaining the 4 \times speed improvement over Vanilla RAG without compromising accuracy on both datasets.

J Additional Computation Analysis

On the NQ dataset, our method achieves an average FTIT of 630, with the selector accounting for 47 (~7% of total time). This selective mechanism enables our method to outperform RAG (3264 FTIT) and fixed-ratio compression approaches with same

compression rate (1007 FTIT).

K Potential Risks

Our method reduces RAG input length while preserving contextual integrity, yet inherits inherent potential risks of RAG such as retrieval-related risks (data bias/information leakage), misleading outputs from outdated/incorrect knowledge, adversarial database manipulation leading to harmful responses and so on. In addition, although not observed at present, compressing text into embeddings may potentially distort the original information, leading to harmful outputs generated by the model.

Model Input
<p>Question: what's the official symbol of the carnival of quebec?</p> <p>Context:</p> <ol style="list-style-type: none"> 1. It is adorned by an arrowed pattern and was worn around the winter coats of the time. It is also a symbol of the Lower Canada Rebellion and the Quebec Winter Carnival, as it is worn by the festival mascot, Bonhomme Carnaval. Imitations are sold and seen throughout the carnival. The belt is represented in a number of artistic creations, such as the illustration "Le Vieux de '37" by Henri Julien, the painting "L'Assemblée des six-comtés" by Charles Alexander Smith and the song "Mon Pays, suivi du Reel des Aristocrates" from néo-trad musical band Les Cowboys Fringants. 2. Carifesta Carifesta () is an annual Caribbean Carnival held in Montreal, Quebec, Canada. It was established in 1974, and is held in July. The event is coordinated by the "Caribbean Cultural Festivities Association", a nonprofit organization. Carifesta was established prior to some Carnivals that take place in the Caribbean, Cayman Carnival Batabano for example. 3. Quebec Winter Carnival The Quebec Winter Carnival (), commonly known in both English and French as Carnaval, is a pre-Lenten festival held in Quebec City. After being held intermittently since 1894, the "Carnaval de Québec" has been celebrated annually since 1955. That year "Bonhomme Carnaval", the mascot of the festival, made his first appearance. 4. Canadian rule (maple leaves) and with Quebec's motto below "Je me souviens". Je me souviens ("I remember") was first carved under the coat of arms of Quebec's Parliament Building façade in 1883. It is an official part of the coat of arms and has been the official licence plate motto since 1978, replacing "La belle province" (the beautiful province). The expression "La belle province" is still used mostly in tourism as a nickname for the province. 5. Montreal Carnivale. The carnivale is named after its mascot Bon Soo, created by Ken MacDougall. Bon Soo Winter Carnival The Bon Soo Winter Carnival is an annual winter carnival in Sault Ste. Marie, Ontario, Canada. The carnival, held every February, began in , and has grown to become one of Ontario's top 50 festivals as well as Canada's second largest winter carnival next to the Montreal Carnivale in Quebec.
Model Outputs
<p>Direct Generation: The official symbol of the Carnival of Quebec, also known as Carnaval de Québec, is a snowgoose. Generated with 0 context tokens (incorrect answer).</p> <p>ACC-RAG-G2 with fixed compression rate 16: The official symbol of the Carnival of Quebec is the "Fleur-de-lis" or "leur-de-lis carnival" as it is commonly known. Generated with 40 context tokens, wrong answer.</p> <p>ACC-RAG-G2(ours): The official symbol of the Carnival of Quebec is "Bonhomme Carnaval," a jovial snowman-like figure wearing a red hat and scarf, who is the mascot of the Quebec Winter Carnival. Generated with 64 context tokens, right answer.</p> <p>ACC-RAG-G2 with fixed compression rate 4: The official symbol of the Carnival of Quebec is "Bonhomme Carnaval," a mascot character with a red nose, a red hat, and a red scarf, who is depicted as a snowman. Generated with 160 context tokens, right answer.</p> <p>Vanilla RAG: The official symbol of the Quebec Winter Carnival is "Bonhomme Carnaval" and the maple leaf with the motto "Je me souviens" (I remember). Generated with 640 context tokens, right answer.</p>

Table 13: A Case of generated responses using different methods.