xRAG: Extreme Context Compression for Retrieval-augmented Generation with One Token

Xin Cheng[1]   **Xun Wang**[2]   **Xingxing Zhang**[2]   **Tao Ge**[2]
Si-Qing Chen[2]   **Furu Wei**[2]   **Huishuai Zhang**[1]   **Dongyan Zhao**[1]

[1] Peking University   [2] Microsoft
chengxin1998@stu.pku.edu.cn

Abstract

This paper introduces xRAG, an innovative context compression method tailored for retrieval-augmented generation. xRAG reinterprets document embeddings in dense retrieval—traditionally used solely for retrieval—as features from the retrieval modality. By employing a modality fusion methodology, xRAG seamlessly integrates these embeddings into the language model representation space, effectively eliminating the need for their textual counterparts and achieving an extreme compression rate. In xRAG, the only trainable component is the modality bridge, while both the retriever and the language model remain frozen. This design choice allows for the reuse of offline-constructed document embeddings and preserves the plug-and-play nature of retrieval augmentation. Experimental results demonstrate that xRAG achieves an average improvement of over 10% across six knowledge-intensive tasks, adaptable to various language model backbones, ranging from a dense 7B model to an 8x7B Mixture of Experts configuration. xRAG not only significantly outperforms previous context compression methods but also matches the performance of uncompressed models on several datasets, while reducing overall FLOPs by a factor of 3.53. Our work pioneers new directions in retrieval-augmented generation from the perspective of multimodality fusion, and we hope it lays the foundation for future efficient and scalable retrieval-augmented systems[1].

1 Introduction

Retrieval-Augmented Language Models (RALMs) [41, 21, 8, 13, 66] have demonstrated remarkable performance in various knowledge-intensive tasks. By retrieving domain-specific, long-tailed, and up-to-date knowledge from a non-parametric datastore, RALMs significantly extend the boundaries of parametric Large Language Models (LLMs). However, integrating entire documents into prompts can substantially increase inference costs and potentially exceed the context limits of LLMs [28, 75]. As shown in Figure 1, with the help of relevant document, LLM could generate the correct answer, albeit at the cost of processing a document that extends the original query by more than tenfold.

How can we mitigate the costs associated with extended context while maintaining the benefits of retrieval augmentation? Recent research interest has converged on a promising direction: Context Compression. This explores two main approaches: soft-prompting methods, such as Gist [56], AutoCompressor [14], and ICAE [19], which compress the context into dense memory slots, and hard-prompting methods, such as LLMLingua [28] and RECOMP [75], where compression is applied on the surface form. These approaches, however, either require significant memory for storing LLM

---

[1]Code available at: `https://github.com/Hannibal046/xRAG`.

RAG

Refer to the background document and answer the question:

Background: Motel 6 | " Beginning in 1986, Motel 6 has advertised through radio commercials featuring the voice of writer and National Public Radio commentator Tom Bodett, with the tagline "We'll leave the light on for you." The ads were created by Dallas advertising agency The Richards Group. They feature a tune composed by Tom Faulkner, performed by him on guitar and Milo Deering on fiddle. The first spots were conceived and written by David Fowler. In 1996, the ads won a Clio Award. The campaign itself has won numerous national and international awards and was selected by Advertising Age magazine as one of the Top 100 Advertising Campaigns of the Twentieth Century." ✓

Question: What company advertised itself with the slogan "We'll leave a light on for you"?

Output: Motel 6. ✓

w/o Retrieval

Answer the question:

Question: What company advertised itself with the slogan "We'll leave a light on for you"?

Output: Marriott International. ✗

xRAG

Refer to the background document and answer the question:

Background: [X]

Question: What company advertised itself with the slogan "We'll leave a light on for you"?

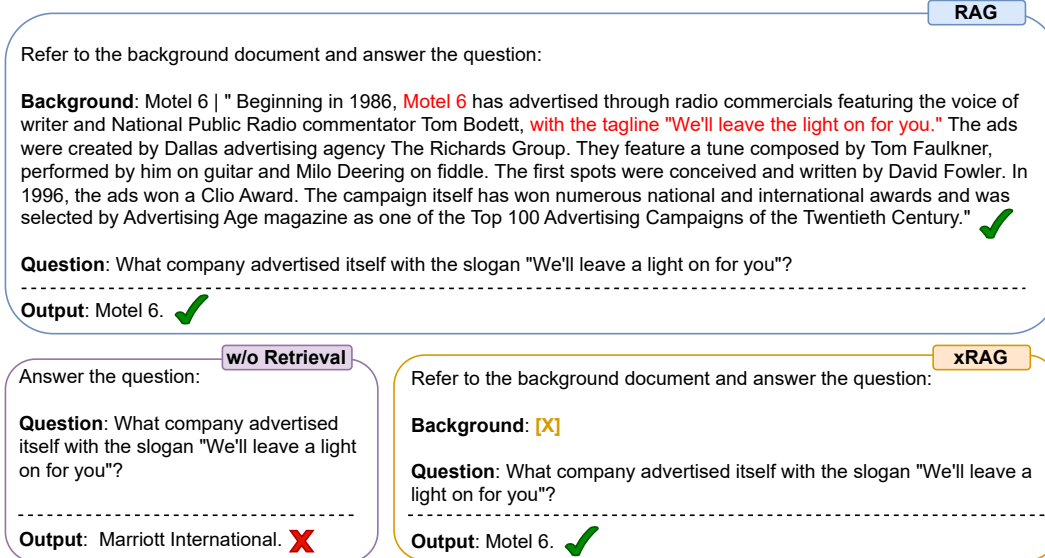
Output: Motel 6. ✓

Figure 1: xRAG enables efficient retrieval augmentation by adding one document token [X].

activations (e.g., 1.05 MB per token as reported by [56]) or suffer from relatively low compression rates. More critically, these methods often fail to address a key aspect that sets RALMs apart from other NLP tasks: through large-scale contrastive learning with question-document pairs, modern dense retrieval systems already condense document information into a single high-dimensional embedding, and *this embedding reveals (almost) as much information as the text* [55, 43].

In this paper, we explore an interesting and novel perspective on retrieval augmentation and context compression—modality fusion. In multimodal literature where text-only language models can learn to "perceive" and "listen", a pretrained modality encoder, such as CLIP [60], is often utilized to extract modality features. These features are then integrated into language models through a modality fusion bridge [44, 50, 33]. Building on this conceptual similarity between retriever encoder and modality encoder, we propose xRAG, which reinterprets document embeddings in dense retrieval—traditionally used solely for retrieval—as features from the retrieval modality. By employing a modality fusion methodology, xRAG seamlessly integrates these embeddings into the language model representation space, eliminating the need for their textual counterparts and achieving an extreme compression rate. In xRAG, the only trainable component is the modality bridge, while both the retriever and the LLM remain frozen. This design choice allows for the reuse of offline-constructed document embeddings and preserves the plug-and-play nature of retrieval augmentation—two essential factors for a practical RAG system.

To verify the effectiveness and versality of our framework, we conducted comprehensive experiments with different LLM backbones, ranging from a dense 7B model to an 8x7B Mixture of Experts model. Our results reveal that adding just one document token could lead to over a 10% improvement across six knowledge-intensive tasks, significantly surpassing previous compression methods. xRAG also delivers results comparable to uncompressed models on several datasets. This is remarkable considering that the only trainable component constitutes less than 0.1% of the LLM's parameters. In terms of efficiency, xRAG reduces total FLOPs by a factor of 3.53 compared to the uncompressed RAG model. We further provide detailed analyses of xRAG, examining various training strategies, data blends, and component selections for the retrieval system. We hope this research lays the foundation for future efficient and scalable retrieval-augmented systems.

2 Related Work

Retrieval-augmented Generation Equipping a parametric language model with a non-parametric datastore has proven effective for various NLP tasks, such as language modeling [36, 54, 81], open-domain question answering [24, 41, 66, 49], machine translation [35, 11] and others. Given the vast
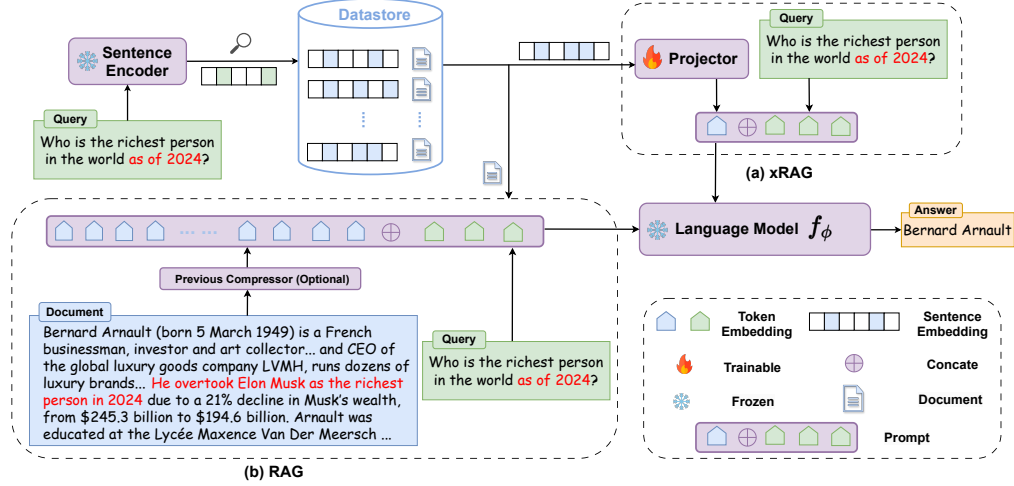
Figure 2: Overview of xRAG (a) and RAG (b). For a given query, RAG typically concatenates the retrieved document with the query, significantly extending the context length. In contrast, xRAG addresses this issue through modality fusion by directly projecting the document embedding into the LLM's representation space. This allows for efficient retrieval-augmentation with the addition of only one token.

design space of this generation paradigm, numerous approaches with different focuses have been proposed. For instance, RETRO [8] and PlugLM [12] feature architectural innovations for better integration with the non-parametric datastore. REALM [21] introduces an end-to-end methodology for joint optimization of the language model and the retriever. REPLUG [66] and RA-DIT [48] enhance the retriever using LLM feedback to better align these two components. DSP [37] and InteR [17] explore sophisticated interactions between the retriever and the language model. Selfmem [13] employs a reward model to iteratively refine both retrieval and generation processes. Self-RAG [4] introduces a self-reflection mechanism that significantly enhances the quality and factuality of language models. For a more comprehensive review, refer to [18, 5, 3]. Our work, xRAG, distinguishes itself from previous studies by adopting a modality fusion approach to the problem of retrieval augmentation, resulting in a both effective and efficient RAG system.

Context Compression Context compression, which aims to reduce the input length of LLMs while preserving the salient information of the original context, has gained significant attention recently. Gist [56] achieves up to a 26x compression rate of instruction by modifying the attention mask and caching the activation of soft gist tokens. ICAE [19] and AutoCompressor [14] compresses long contexts into short, compact memory slots that can be directly conditioned on by the LLM for various purposes. LLMLingua [28, 29] proposes a coarse-to-fine prompt compression method based on the perplexity score. These are general compressors, while others propose methods specifically designed for RAG, such as FilCo [72] and RECOMP [75]. A more detailed comparison between these compression methods in terms of memory efficiency, compression rate, and adaptability can be found in Appendix A.

3 Methods

Problem Formulation In retrieval-augmented generation, a non-parametric datastore $\mathbb{D} = \{(\mathrm{E}_i, \mathrm{D}_i)\}_{i=1}^{|\mathbb{D}|}$ consists of pairs where each D_i represents a document chunk as a sequence of L_i tokens $\mathrm{D}_i = \{d_1^i, \ldots, d_{L_i}^i\}$. Correspondingly, E_i is the dense representation derived from a sentence embedding model $\mathbf{SE}_\theta(\cdot)$ with input D_i. For an input query q, its dense representation $\mathbf{SE}_\theta(q)$ is used to find the relevant documents by matching against the collection $\{\mathrm{E}_i\}_{i=1}^{|\mathbb{D}|}$ with certain similarity search algorithm such as MIPS. After retrieval, the system selects a relevant pair (E, D) from $\mathbb{D}$, concatenates the chosen document D with q, and processes the combined input with a language model
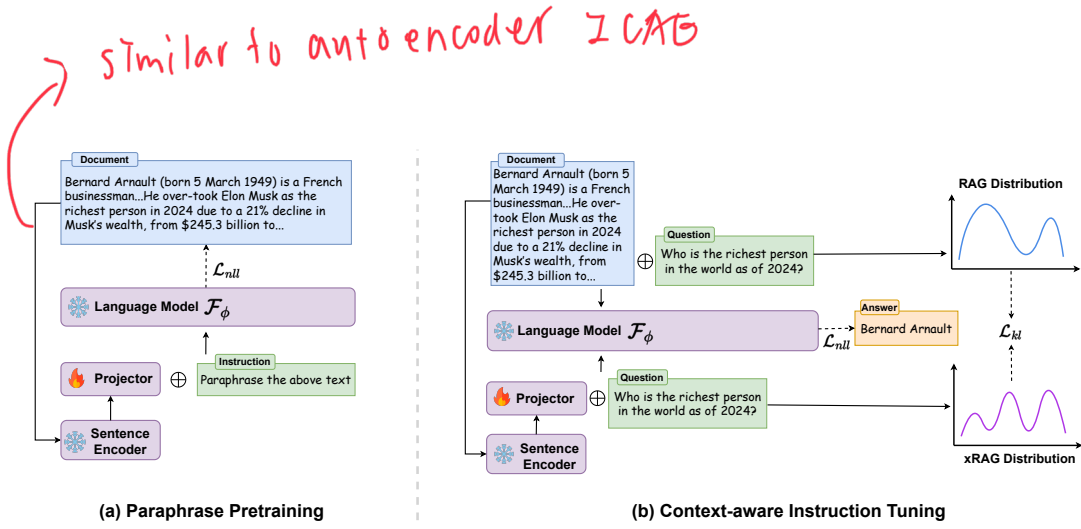
similar to auto encoder ICAG

Figure 3: Two-stage training strategy of xRAG including (a) Paraphrase Pre-training on unlabeled corpus and (b) Context-aware Instruction Tuning optimized with labeled data and self-distillation.

$\mathcal{F}_\phi(D \oplus q)$. Optionally, a context compression module $\mathcal{C}$ can be integrated to reduce the length of D from L to a more concise l, achieving a compression ratio of $\frac{L}{l}$.

3.1 xRAG Architecture

Traditional methods for document compression typically focus on surface form of the document [28, 72, 75]. In contrast xRAG tackle the problem from a modality fusion view. Concretely, we introduce a modality projector $\mathbf{W}$, which is trained to directly project the retrieval features E into the LLM representation space. Our proposed framework is visually contrasted with the traditional RAG system in Figure 2. In the standard RAG, the input to the LLM comprises the embeddings $\texttt{Emb}(D \oplus q)$ of length $|D| + |q|$, where $\texttt{Emb}$ signifies the embedding layer of the LLM. Conversely, with xRAG, the modified input is represented as $\mathbf{W}(E) \oplus \texttt{Emb}(q)$, which yields a substantially reduced length of $1 + |q|$. In this framework, the challenges come from the modality fusion: How can a text-only language model understand features from retrieval modality? To achieve this, we explore a two-stage training strategy: Paraphrase Pretraining followed by Context-aware Instruction Tuning.

3.2 Paraphrase Pretraining

Similar to the pretraining strategies employed in vision-language models that use image-captioning data to align two modalities [50, 15, 51], the primary objective of our paraphrase pretraining is to build a compatible representation between the extracted retrieval feature and the corresponding document. Illustrated in Figure 3(a), for each pair (E, D) in a retrieval corpus $\mathbb{D}$, we employ a natural language instruction $\mathbf{X}_{\texttt{instruct}}$ to prompt the LLM to undertake a paraphrasing task (e.g. "[X] The above text could be paraphrased as: [D]", where [X] and [D] are placeholders for $\mathbf{W}(E)$ and document D)[2]. In this setup, the model learns to connect $\mathbf{W}(E)$ and D by recovering D on the condition of $\mathbf{W}(E)$ and the model is optimized by:

$$\mathcal{L}_{\text{nll}} = -\sum_{i=1} \log p_\phi(d_i | \mathbf{W}(E), \mathbf{X}_{\texttt{instruct}}, d_{<i}) \tag{1}$$

where p_ϕ is given by the softmax distribution of LLM $\mathcal{F}_\phi$, and $d_{<i}$ denotes the document token before current prediction token d_i, achieved by casual attention mask in auto-regressive LMs.

3.3 Context-aware Instruction Tuning

After the pretraining phase, although the language model $\mathcal{F}_\phi$ has developed an internally compatible representation, it has never been explicitly trained to utilize these features for downstream tasks. To address this gap, we proceed to instruct the model in harnessing the fused feature $\mathbf{W}(E)$ by continually training the model on data where the answer is closely associated with the given context, including reading comprehension, summarization, and open domain question answering data. We

---

[2]To maintain diversity, we sample from an instruction pool, which could be found in Appendix B.

constructed an mixed dataset, containing approximately 1 million entries from open-source datasets, as detailed in Appendix C. For each triplet in the dataset, $(\mathbf{X}_{\mathrm{context}}, \mathbf{X}_{\mathrm{question}}, \mathbf{X}_{\mathrm{answer}})$, we initially obtain the sentence representation for $\mathbf{X}_{\mathrm{context}}$ via the embedding model $\mathrm{E}_{\mathrm{context}} = \mathbf{SE}_\theta(\mathbf{X}_{\mathrm{context}})$. Subsequently, we refine the optimization of on two directions:

Optimization I: Language Modeling. Aligned with established instruction tuning methodologies [71, 23, 48, 42], our objective is to finetune the model so that it generates the correct output when provided with a specific instruction, conditioned upon the given context information. Unlike traditional models that utilize the textual context $\mathbf{X}_{\mathrm{context}}$, our method employs a dense feature $\mathrm{E}_{\mathrm{context}}$ to encapsulate the context information:

$$\mathcal{L}_{\mathrm{nll}} = - \sum_{i=1} \log p_\phi(\mathbf{X}_{\mathrm{answer},i} | \mathbf{W}(\mathrm{E}_{\mathrm{context}}), \mathbf{X}_{\mathrm{question}}, \mathbf{X}_{\mathrm{answer},<i}) \tag{2}$$

Optimization II: Self-Distillation. The second trajectory of optimization aims to guide the xRAG in the effective utilization of contextual information, drawing from the principles of self-distillation [2, 67] and imitation learning [58, 22]. By considering the RAG model as a "teacher" and xRAG as a "student", we endeavor to distill the knowledge from RAG, thereby enabling xRAG to emulate the RAG model's proficiency in handling the full, uncompressed documents. This approach enhances xRAG's resilience in scenarios where it encounters noisy or irrelevant context that may not directly lead to the correct answer, detailedly discussed in § 6.1. Concretely, for a language model $\mathcal{F}_\phi$ using either $\mathbf{X}_{\mathrm{context}}$ or $\mathrm{E}_{\mathrm{context}}$ as the source of context, our objective is to minimize the divergence between the two resulting output distributions. This discrepancy is measured using the Kullback-Leibler (KL) divergence:

$$\mathcal{L}_{\mathrm{kl}} = D_{\mathrm{KL}}(p_\phi(\mathbf{X}_{\mathrm{answer}} | \mathbf{X}_{\mathrm{context}}, \cdot) \, || \, p_\phi(\mathbf{X}_{\mathrm{answer}} | \mathbf{W}(\mathrm{E}_{\mathrm{context}}), \cdot)) \tag{3}$$

Here $\mathbf{X}_{\mathrm{question}}$ is omitted for brevity and the final loss is the linear combination controlled by a hyperparameter: $\mathcal{L}_{nll} + \alpha\mathcal{L}_{kl}$.

back translation [handwritten annotation]

3.4 Design Principle

In designing the projector $\mathbf{W}$, our primary objective is to maintain the simplicity of the framework. We therefore opted for a two-layer MLP while other more sophisticated module such as Q-Former [44] could also be considered. Notice that the projector is the only trainable component, accounting for only 0.46% of the total parameters in the Mistral-7b model and 0.07% in the Mixtral-8x7b model. Such a design choice departs from previous studies that necessitated full-parameter tuning to adapt LLMs for compressed contexts [56, 72, 14]. We believe this approach will likely be more accessible and practical because, fundamentally, the RAG itself functions as a plug-and-play module for LLMs, and so should its compressed version. This design also avoid the risk of compromising other core capabilities of LLM during full-parameter tuning, as observed in [52, 53].

Moreover, in contrast to other compression methods that necessitate storing LLM activations for each compressed token [56, 19, 14]—an impractical strategy in the RAG setting, given the millions of documents involved—our method introduces no additional memory overhead. Instead, it leverages offline-constructed document embeddings, originally designed for retrieval. To summarize, xRAG not only simplifies the integration process but also avoids unnecessary computational or memory expenses.

4 Experimental Setup

4.1 Evaluation Dataset

We evaluated the performance of xRAG primarily on knowledge-intensive tasks which encompass a range of challenges: (1) three Open Domain Question Answering datasets that address questions on a wide array of topics: Natural Questions [40], TriviaQA [32], and Web Questions [7]. (2) one Multihop Question Answering dataset, HotpotQA [77], which necessitates multi-step reasoning to generate answers. (3) one Long-form Question Answering dataset, TruthfulQA [47], that requires the generation of long-form and truthful responses. (4) one fact-checking dataset, FactKG [38], which challenges the model to use complex reasoning to determine the factual accuracy of given claims.

In line with the KILT [59] and GenRead [79], we assessed three ODQA datasets and HotpotQA using the Exact Match (EM) metric, FactKG with Accuracy, and for the long-form QA, we used both the F1 score and Rouge-L (R-L) score. These tasks demand a broad spectrum of world knowledge and have been extensively explored in the retrieval-augmentation literature [41, 21, 34, 9, 24, 66].

4.2 Implementation Details

To demonstrate the versatility of our framework, we choose two backbones, differing in scale and architecture: Mistral-7b [26] and Mixtral-8x7b [27]. For the retrieval corpus, we utilized the Wikipedia dump from December 2021, which was pre-processed into passages following the methodology described in [25]. This resulted in approximately 37 million passages, each averaging 180 tokens in length. Our default retrieval model is the SFR [1], which, at the time of writing this paper, holds the leading position on the MTEB leaderboard [57]. We use *top-1* ranked document for inclusion in our instruction-tuning dataset and for the evaluation of downstream tasks. More details are provided in Appendix D.

4.3 Baselines

In determining appropriate baselines for comparison, we adhered to a fundamental principle: the selected compression methods must support the general plug-and-play capability of retrieval augmentation. This entails that they should function effectively without the need for dataset-specific tuning [75, 72], or any alteration to the parameters of LLMs [72, 56]. Furthermore, given the extensive volume of the retrieval corpus, it is essential that these compression methods demonstrate memory efficiency, specifically by not requiring the storage of LLM activations for each individual token [56, 19, 14]. With these criteria in mind, our evaluation compares xRAG to the following baselines: **(I)** Primarily, we consider two variants of LLMs: one that operates without retrieval augmentation and another that includes it. These serve as the lower and upper performance bounds for our study of compression techniques, respectively. **(II)** Additionally, our comparisons extend to LLMLingua [28], a plug-and-play approach for context compression. **(III)** Taking inspiration from [56], we incorporate a method of discrete compression using TF-IDF. This approach yields compression rates comparable to those achieved by xRAG and serves as the lower bound for discrete compression.

| Task Type | NQ | TriviaQA | WebQA | HotpotQA | TrutufulQA | | FactKG | Average | # Doc Length |
	Open-Domain QA (EM)			Multihop QA (EM)	Long-form QA (F1/R-L)		Fact Checking (Acc)		
Mistral-7b									
w/o retrieval	30.25	57.08	34.89	27.02	26.23	25.51	54.78	36.54 (0.0%)	0
w retrieval	**42.71**	65.88	37.84	**38.79**	26.50	25.92	**67.76**	**43.63** (19.4%)	175.1
***with Compression**									
LLMLingua [†]	30.64	57.94	32.63	29.91	25.70	25.10	64.17	38.01 (4.0%)	98.6
LLMLingua [‡]	28.81	57.09	32.33	29.13	26.10	25.39	63.57	37.48 (2.5%)	61.1
TF-IDF	30.25	58.49	35.43	26.62	26.33	25.83	59.56	37.49 (2.6%)	1
xRAG	39.10	65.77	**39.40**	34.05	**28.10**	27.71	63.08	42.46 (16.2%)	1
Mixtral-8x7b									
w/o retrieval	41.99	71.10	40.31	32.87	25.60	24.90	62.64	42.76 (0.0%)	0
w retrieval	45.15	70.34	41.26	**43.46**	27.10	25.80	**70.42**	46.22 (8.0%)	175.1
***with Compression**									
LLMLingua[†]	37.65	67.70	36.02	35.66	25.99	25.39	67.98	42.32 (-1.0%)	96.6
LLMLingua[‡]	37.81	67.81	35.78	35.27	25.68	25.00	68.03	44.17 (-1.3%)	61.1
TF-IDF	41.19	69.94	41.63	32.05	26.80	26.00	66.17	43.41 (1.4%)	1
xRAG	**47.28**	**74.14**	**44.50**	39.66	**27.80**	26.64	68.20	**46.91** (9.7%)	1

Table 1: Experimental results on six downstream tasks. The best results are in **bold** and the second best are with underscore. Percentage in the brackets denotes the relative improvement over non-retrieval setting. LLMs are frozen during the experiments and retrieved documents are set the same for different compression methods. [‡] and [†] denotes different compression ratio.

5 Experimental Results

5.1 Knowledge Intensive Tasks

In Table 1, we present our main results. Across both Mistral-7b and Mixtral-8x7b configurations, we observe a consistent and significant improvement when retrieval augmentation is applied (p-value

< 0.05), although the gains are more modest for the larger model configurations. This trend aligns with observations reported by [66, 48]. Further analysis on the efficacy of various compression techniques reveals that xRAG outperforms other approaches by a large margin. Remarkably, xRAG not only reduces the token count drastically—from 175.1 to a single token—but also maintains robust performance levels. In some instances, xRAG's performance is comparable to, or even exceeds, that of the uncompressed models. Specifically, in the Mistral-7b configuration, xRAG achieves nearly the same performance improvement as the uncompressed model (16.6% compared to 19.4%), and in the Mixtral-8x7b configuration, it surpasses the uncompressed model (9.7% compared to 8.0%). One possible reason lies in the vulnerability of current RAG system when the irrelevant or misleading documents are presented, a topic detailed discussed in §6.1. We also observe that xRAG performs well in tasks that require document understanding, such as TriviaQA. However, in tasks that demand reasoning over document, like HotpotQA and FactKG, xRAG lags behind by a considerable gap.

| | CUDA Time (ms) | | | GFLOPs | | |
	RAG	xRAG	Improvement	RAG	xRAG	Improvement
FactKG	431.5	215.6	x2.01	4683.8	1289.5	x3.63
NQ	918.7	611.3	x1.51	1338.6	384.0	x3.48
TriviaQA	807.1	512.1	x1.57	1667.2	492.3	x3.38
WebQA	872.6	577.3	x1.51	1405.1	386.8	x3.63
Average			**x1.64**			**x3.53**

Table 2: Comparison of RAG and xRAG performance in CUDA Time and GFLOPS.

5.2 Computational Efficiency

In this section, we conduct a thorough assessment of our framework's computational efficiency and memory management. To rigorously evaluate our model, we employed `Torch Profiler`[3] to measure the CUDA Time (milliseconds) and Giga FLOPs of both the RAG and xRAG models across four real-world datasets. In these evaluations, the Mistral-7b, operating in bfloat16 inference mode, served as the base LLM. CUDA Time and GFLOPs were calculated on an average per batch basis with a fixed batch size, and GFLOPs were normalized by the number of generated tokens. These experiments were performed on the same computational hardware, specifically an Nvidia A100 and an AMD EPYC 7V12 64-Core Processor. As depicted in Table 2, despite variations in prompt and generation lengths across the datasets, xRAG significantly outpaced the RAG model, achieving a x1.64 increase in CUDA Time efficiency and a x3.53 reduction in GFLOPs.

6 Analysis

6.1 Evaluation Beyond the Overall Score

Although retrieval augmentation generally boosts performance as shown by aggregate metrics, it may not uniformly benefit all instances. In certain cases, the retrieval system might provide irrelevant or even misleading information, leading to incorrect answers that were previously correct [78, 72, 5]. To enable a more fine-grained evaluation, we introduce two novel metrics: the Resilience Rate and the Boost Rate. The resilience rate quantifies the percentage of instances in which the system's responses remain correct both before and after retrieval augmentation, highlighting the system's stability and robustness. Conversely, the boost rate measures the percentage of instances that were initially answered incorrectly but were rectified following the introduction of a retrieved document, thereby assessing the efficacy of retrieval augmentation. An ideal RAG system should have both high resilience rate and boost rate.

In Figure 4, we display these metrics for the uncompressed RAG and two compression methods: LLMLingua and xRAG. Surprisingly, although retrieval augmentation generally enhances performance, the resilience rate for RAG averages only 75.2%, indicating that retrieval can adversely affect about one-quarter of previously correct answers. In contrast, xRAG demonstrates considerable

---

[3]`https://pytorch.org/docs/stable/profiler.html#module-torch.profiler`

robustness across all evaluated datasets. This robustness largely stems from xRAG's ability to maintain an unbiased stance toward the internal knowledge representation of the LLM, especially when confronted with noisy retrieval content. Similar trends are noted in [48, 52], where search-augmented instruction learning is shown to bolster the robustness of language models. However, xRAG still lags behind RAG in boost rate, particularly in multi-hop reasoning tasks. It is crucial to note that a high resilience rate does not necessarily mean that the LLM disregards the provided information, which could potentially lead to a reduced boost rate. A comparative analysis with LLMLingua indicates that xRAG is not only more robust but also more effective.
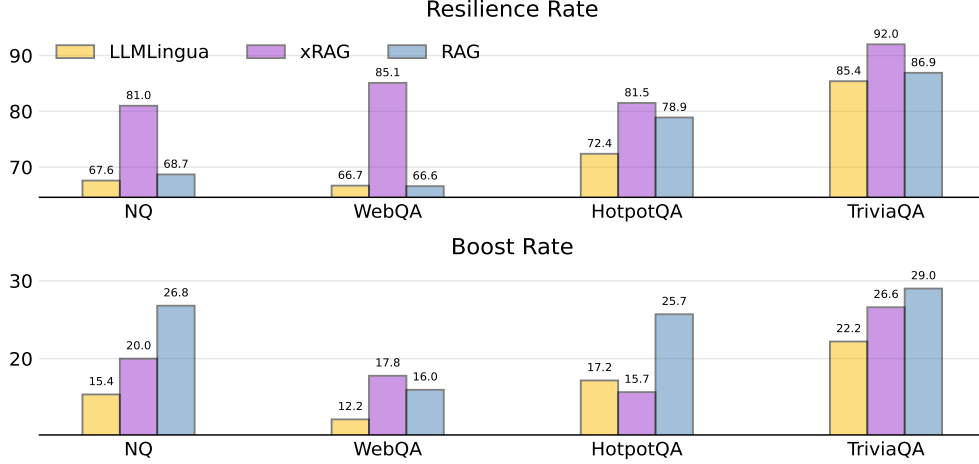

Figure 4: Resilience rate and boost rate of three augmentation methods: LLMLinuga, xRAG and RAG over a Mixtral-8x7b baseline without retrieval augmentation.

6.2 What makes xRAG effective?

| | NQ | TriviaQA | WebQA | HotpotQA | Averaged | | |
					Performance	Resilience	Boost
Mistral-7b							
xRAG	39.10	65.77	39.40	34.05	**44.58**	82.3%	**22.2%**
w/o finetune	30.14	59.48	35.19	26.70	37.87	66.6%	20.8%
w/o pretrain	31.25	59.07	41.19	24.32	38.95	79.8%	14.1%
w/o nll	35.46	65.27	39.57	31.80	43.02	**83.7%**	19.4%
w/o self-kd	34.99	64.33	39.22	27.45	41.49	76.2%	20.8%
w LoRA	35.71	60.14	40.45	22.91	39.80	76.0%	18.0%
Mixtral-8x7b							
xRAG	47.48	74.14	44.50	39.66	**51.45**	84.9%	20.0%
w/o finetune	34.46	64.08	34.89	30.43	40.96	65.9%	17.8%
w/o pretrain	42.54	71.17	47.44	31.23	48.09	85.0%	14.2%
w/o nll	45.10	72.85	45.03	37.11	50.02	84.8%	18.9%
w/o self-kd	42.38	72.26	44.73	32.41	47.94	79.8%	18.9%

Table 3: Ablation on different training strategy for xRAG.

This section delves into a thorough evaluation of various elements that contribute to xRAG's overall performance, focusing on its training strategy, the blend of datasets used and the effect of different embedding models. Due to the space limit, we present the last factor in Appendix E.

1. Training Strategy We carefully ablate four optimization choices: pretraining, instruction tuning, and two optimization objectives—language modeling (nll) and self-distillation (self-kd). We also train a Mistral-7b with LoRA on our instruction tuning dataset to rule out the possibility that our improvement simply comes from tuning on more data. The outcomes are presented in Table 3. Our analysis reveals that the interplay of different training strategies significantly contributes to the
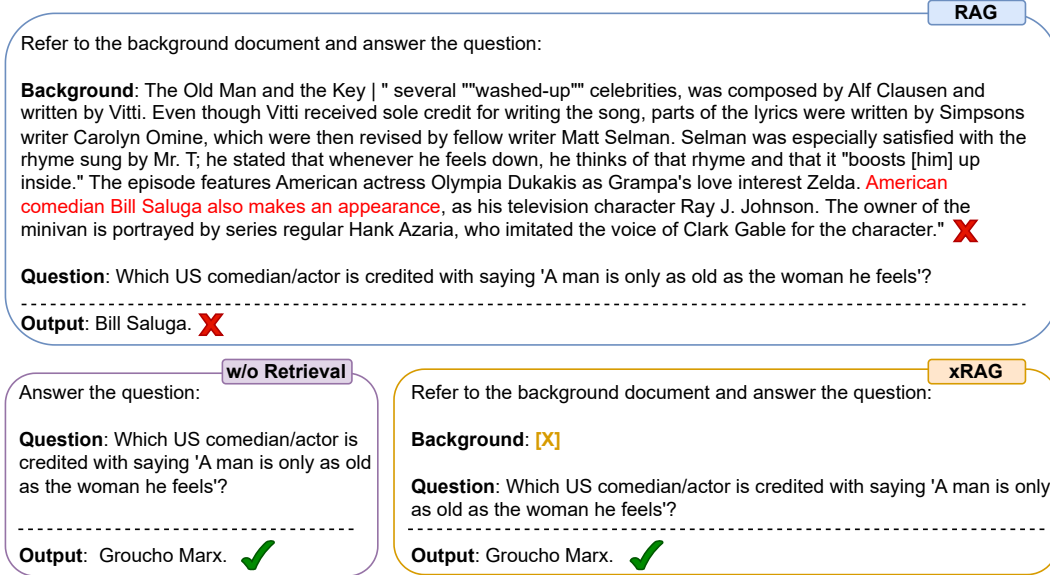
Figure 5: Given the misleading document, RAG model tend to generate a wrong answer based on the document, while xRAG demonstrate its robustness by leveraging the internal knowledge of LLM.

performance of our framework. In the case of Mistral-7b, pretraining and finetuning phases are of equal significance to the end results. However, for Mixtral-8x7b, the impact of pretraining is notably diminished, likely due to the larger model's enhanced capability to incorporate multi-modality information. Furthermore, we find that during finetuning, self-distillation is more important than language modeling. The primary advantage of self-distillation lies in bolstering the resilience rate of the xRAG system. Optimization with nll loss tends to cause an overreliance on context information, rendering the system more vulnerable when the retriever fails to fetch relevant documents.

	# Train	NQ	TriviaQA	WebQA	HotpotQA	Average		
						Performance	Resilience	Boost
xRAG (Mistral-7b)	955k	39.10	65.77	39.40	34.05	**44.58**	82.3%	22.2%
w RC only	488k	36.98	65.77	41.39	32.82	<u>44.24</u>	81.9%	22.4%
w QA only	385k	36.45	65.57	41.14	31.80	43.74	80.5%	22.1%
w Summ only	81k	36.37	64.95	40.40	31.98	43.42	78.8%	**22.8%**

Table 4: Abaltion results on different data selection strategy.

II. Instruction-tuning Dataset Blend As discussed in §3.3, our instruction-tuning dataset primarily comprises three categories: reading comprehension, open-domain QA, and text summarization. To explore the effects of different data blends, we instruction-tune three xRAG model variants, each using data from these distinct categories. The results are shown in Table4. Our analysis reveals that among the dataset blends, reading comprehension data most significantly enhances the xRAG model's performance, as evidenced by both high resilience and boost rates. Intriguingly, when tuned solely with summarization data, xRAG still manages to deliver strong performance on QA datasets it has never been exposed to. This finding underscores that the advantages of instruction tuning for xRAG are not rooted in task-specific knowledge. Instead, they derive from the model's improved ability to utilize projected context information effectively.

6.3 Case Study

In Figure 5, we show one interesting case about the robustness of xRAG. When retrieval system provide misleading content, standard RAG would overly rely on the document and generate answer that are faithful to the document while not factually true. Our xRAG model opt to rely on the internal

knowledge of LLM and being robust to the misleading content. In Appendix H, we include more cases about xRAG including several error analysis.

7 Conclusion

In this work, we present xRAG, an innovative context compression method tailored for retrieval-augmented generation. For knowledge-intensive tasks, xRAG can be significantly faster than RAG while maintaining comparable performance. We are excited about the future of this modality-based retrieval-augmented system and plan to further improve its performance in the areas of reasoning over embedding, handling multiple documents, and combining with multi-vector retrieval.

References

[1] Sfr-embedding-mistral: Enhance text retrieval with transfer learning. `https://blog.salesforceairesearch.com/sfr-embedded-mistral/`.

[2] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

[3] Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. Retrieval-based language models and applications. In Yun-Nung (Vivian) Chen, Margot Margot, and Siva Reddy, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 41–46, Toronto, Canada, July 2023. Association for Computational Linguistics.

[4] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023.

[5] Akari Asai, Zexuan Zhong, Danqi Chen, Pang Wei Koh, Luke Zettlemoyer, Hannaneh Hajishirzi, and Wen tau Yih. Reliable, adaptable, and attributable language models with retrieval, 2024.

[6] 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. Ms marco: A human generated machine reading comprehension dataset, 2018.

[7] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

[8] 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. Improving language models by retrieving from trillions of tokens, 2022.

[9] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions, 2017.

[10] Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. Dialogsum: A real-life scenario dialogue summarization dataset, 2021.

[11] Xin Cheng, Shen Gao, Lemao Liu, Dongyan Zhao, and Rui Yan. Neural machine translation with contrastive translation memories, 2022.

[12] Xin Cheng, Yankai Lin, Xiuying Chen, Dongyan Zhao, and Rui Yan. Decouple knowledge from parameters for plug-and-play language modeling, 2023.

[13] Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. Lift yourself up: Retrieval-augmented text generation with self memory, 2023.

[14] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts, 2023.

[15] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023.

[16] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs, 2019.

[17] Jiazhan Feng, Chongyang Tao, Xiubo Geng, Tao Shen, Can Xu, Guodong Long, Dongyan Zhao, and Daxin Jiang. Synergistic interplay between search and large language models for information retrieval, 2023.

[18] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[19] Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model, 2023.

[20] Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. Association for Computational Linguistics, 2019.

[21] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training, 2020.

[22] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[23] Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. Camels in a changing climate: Enhancing lm adaptation with tulu 2, 2023.

[24] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering, 2021.

[25] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models, 2022.

[26] 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. Mistral 7b, 2023.

[27] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

[28] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models, 2023.

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

[30] Kelvin Jiang, Dekun Wu, and Hui Jiang. FreebaseQA: A new factoid QA data set matching trivia-style question-answer pairs with Freebase. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 318–323, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[31] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China, November 2019. Association for Computational Linguistics.

[32] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017.

[33] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models, 2024.

[34] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020.

[35] Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation, 2021.

[36] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models, 2020.

[37] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp, 2023.

[38] Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. Factkg: Fact verification via reasoning on knowledge graphs. *arXiv preprint arXiv:2305.06590*, 2023.

[39] Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge, 2017.

[40] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur 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. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7, 2019.

[41] 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. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[42] Haoran Li, Qingxiu Dong, Zhengyang Tang, Chaojun Wang, Xingxing Zhang, Haoyang Huang, Shaohan Huang, Xiaolong Huang, Zeqiang Huang, Dongdong Zhang, Yuxian Gu, Xin Cheng, Xun Wang, Si-Qing Chen, Li Dong, Wei Lu, Zhifang Sui, Benyou Wang, Wai Lam, and Furu Wei. Synthetic data (almost) from scratch: Generalized instruction tuning for language models, 2024.

[43] Haoran Li, Mingshi Xu, and Yangqiu Song. Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence, 2023.

[44] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.

[45] Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, Singapore, December 2023. Association for Computational Linguistics.

[46] Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen tau Yih, and Xilun Chen. How to train your dragon: Diverse augmentation towards generalizable dense retrieval, 2023.

[47] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022.

[48] Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. Ra-dit: Retrieval-augmented dual instruction tuning, 2023.

[49] Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. Ra-dit: Retrieval-augmented dual instruction tuning, 2024.

[50] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[51] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, Yaofeng Sun, Chengqi Deng, Hanwei Xu, Zhenda Xie, and Chong Ruan. Deepseek-vl: Towards real-world vision-language understanding, 2024.

[52] Hongyin Luo, Yung-Sung Chuang, Yuan Gong, Tianhua Zhang, Yoon Kim, Xixin Wu, Danny Fox, Helen Meng, and James Glass. Sail: Search-augmented instruction learning, 2023.

[53] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning, 2024.

[54] Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. Nonparametric masked language modeling, 2023.

[55] John X. Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M. Rush. Text embeddings reveal (almost) as much as text, 2023.

[56] Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens, 2024.

[57] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark, 2023.

[58] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pages 3878–3887. PMLR, 2018.

[59] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. Kilt: a benchmark for knowledge intensive language tasks, 2021.

[60] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[61] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.

[62] Siva Reddy, Danqi Chen, and Christopher D. Manning. Coqa: A conversational question answering challenge, 2019.

[63] Anna Rogers, Olga Kovaleva, Matthew Downey, and Anna Rumshisky. Getting closer to ai complete question answering: A set of prerequisite real tasks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8722–8731, 2020.

[64] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction, 2022.

[65] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks, 2017.

[66] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. Replug: Retrieval-augmented black-box language models, 2023.

[67] Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022.

[68] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge, 2019.

[69] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training, 2024.

[70] Shuohang Wang, Yichong Xu, Yuwei Fang, Yang Liu, Siqi Sun, Ruochen Xu, Chenguang Zhu, and Michael Zeng. Training data is more valuable than you think: A simple and effective method by retrieving from training data, 2022.

[71] Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. How far can camels go? exploring the state of instruction tuning on open resources, 2023.

[72] Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. Learning to filter context for retrieval-augmented generation, 2023.

[73] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022.

[74] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

[75] Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with compression and selective augmentation, 2023.

[76] Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

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

[78] Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making retrieval-augmented language models robust to irrelevant context, 2023.

[79] Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. Generate rather than retrieve: Large language models are strong context generators, 2023.

[80] Jungmin Yun, Mihyeon Kim, and Youngbin Kim. Focus on the core: Efficient attention via pruned token compression for document classification. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13617–13628, Singapore, December 2023. Association for Computational Linguistics.

[81] Zexuan Zhong, Tao Lei, and Danqi Chen. Training language models with memory augmentation, 2022.

A   Comparison between different Context Compression Models

In Table 5, we present a detailed comparison of various context compression models, emphasizing their real-world applicability. This comparison focuses on two key aspects: (1) Plug-and-play capability, which assesses whether dataset-specific tuning is necessary for new, unseen data; (2) Memory efficiency, which evaluates if additional memory space is required to store the compressed information, such as high-dimensional vectors typically used in soft prompting methods.

Model	Specifically designed for RAG	Maximun Compression Rate	Approach	Plug-and-Play	Memory Efficient
AutoCompressor [14]	✗	x15	Soft Prompting	✗	✗
Gist [56]	✗	x26	Soft Prompting	✗	✗
ICAE [19]	✗	x8	Soft Prompting	✓	✗
LLMLingua [28]	✗	x20	Prompt Editing	✓	✓
Selective Context [45]	✗	x5	Prompt Editing	✓	✓
Token Elimination [80]	✓	x10	Attention Filtering	✓	✓
FilCo [70]	✓	x2	Prompt Editing	✗	✓
RECOMP [75]	✓	x16.6	Prompt Editing	✓	✓
xRAG	✓	x178	Modality Fusion	✓	✓

Table 5: Comparison between different compression methods from their setting to design principle.

B   Instructions for Paraphrase Pretraining

The list of instructions for Paraphrase Pretraining is shown in Table 6. They present the same meaning with natural language variance.

- "Background: [X] means the same as [D] "
- "Background: [X] Can you put the above sentences in your own terms? [D] "
- "[X] Please provide a reinterpretation of the preceding background text. [D] "
- "These two expressions are equivalent in essence:(1) [X] (2) [D] "
- "Background: [X] is a paraphrase of what? [D] "
- "[X] Could you give me a different version of the background sentences above? [D] "
- "In other words, background: [X] is just another way of saying: [D] "
- "You're getting across the same point whether you say background: [X] or [D] "
- "[X] After uppacking the ideas in the background information above, we got: [D] "
- "[X] Please offer a restatement of the background sentences I've just read. [D] "
- "Background: [X] , which also means: [D] "
- "Strip away the mystery, and you'll find [X] is simply another rendition of: [D] "
- "The essence of background: [X] is captured again in the following statement: [D] "

Table 6: Instructions used for Paraphrase Pretraining where [X] and [D] are placeholders for projected retrieval feature $\mathbf{W}(E)$ and document D.

C Details for Instruction Tuning Dataset

After collecting the raw data from different categories, we use templates[4] from FLAN [73] to construct instruction tuning dataset. In Table 7 we list an overview and in Table 8 we list the detailed information for each subtask of our dataset. For QA tasks that lack an explicit context, we perform a retrieval operation within the corpus $\mathbb{D}$ to identify the most relevant document to serve as context. This approach is akin to the retrieval-augmented instruction tuning depicted in [52, 48].

Task Type	# Involved datasets	# Train	# Prompt	# Label
Reading Comprehension	7	488,344	447.62	30.34
Summarization	3	81,821	483.49	53.29
Open Domain QA	7	385,173	203.55	20.09

Table 7: Overall statistics of Instruction Tuning dataset.

Task Type	Dataset	# Train	# Prompt Len	# Label Len
Reading Comprehension	CoQA [62]	7101	617.98	77.75
	DROP [16]	76098	356.06	3.86
	NarrativeQA [39]	32747	702.39	7.86
	PubMedQA [31]	1000	397.91	65.4
	QuAIL [63]	10246	512.9	2.0
	SQuAD v2 [61]	130319	214.54	6.87
	PwC [19]	241564	571.35	53.07
Open Domain QA	NQ [40]	87925	203.62	5.976
	TriviaQA [32]	78785	216.1	6.49
	CommonsenseQA [68]	9741	223.64	2.0
	WikiQA [76]	1040	192.89	40.79
	YahooQA[5]	87358	196.56	56.7
	FreebaseQA [30]	20353	218.49	4.87
	MSMarco [6]	99994	194.82	15.91
Summarization	CNN/DM [65]	100000	616.99	63.37
	SamSum [20]	14731	187.87	29.12
	DialogSum [10]	12460	247	37.61

Table 8: Detailed data statistics for our Context-aware Instruction Tuning Dataset.

---

[4] https://github.com/google-research/FLAN/blob/main/flan/templates.py

D Implementation Details

For the language models we use, Mixtral-8x7b is approximately 6.5 times larger in scale compared to Mistral-7b and features a divergent architectural approach—specifically, a dense versus mixture-of-experts design. For our assessments, we employed the instruction-tuned variants of these models.

Owing to efficiency constraints, we opted not to perform on-the-fly retrieval. Instead, we pre-constructed a retrieval index using the efficient and robust multi-vector retriever, ColBERT-v2 [64], from which we retrieved the *top-1* ranked document for inclusion in our instruction-tuning dataset and for the evaluation of downstream tasks. Subsequently, we re-encoded these documents using the embedding model of interest (e.g., SFR). This strategy allows us to iterate data-centric experiments quickly. All experiments are conducted on the a setup of 8xNvidia A100 GPUs.

In Table 9 and Table 10, we list the hyperparameters for Paraphrase Pretraining and Context-aware Instruction Tuning.

Hyperparameter	Assignment
optimizer	AdamW
learning rate	6e-3
lr scheduler type	linear
warmup ratio	0.03
weight dacay	0.0
epochs	1
flash attention	True
batch size	12
gradient accumulation steps	4
num GPUs	8
max sequence length	336
max train samples	2,000,000

Table 9: Hyperparameters for Paraphrase Pretraining.

Hyperparameter	Assignment
optimizer	AdamW
learning rate	2e-5
lr scheduler type	linear
warmup ratio	0.03
weight dacay	0.0
epochs	1
KL α	2.0
KL temperature	1.0
flash attention	True
batch size	4
gradient accumulation steps	2
num GPUs	8
max sequence length	1024
max train samples	955,338

Table 10: Hyperparameters for Context-aware Instruction Tuning.

E   About different Embedding Models

In our primary experiments, we use the SFR model as our default sentence embedding model. This section delves into the effects of different embedding models. We examine four universal text embedding models: E5-Mistral and E5-Large [69] alongside BGE-Large and BGE-Base [74]. Additionally, we assess two retrieval-specific models: Dragon [46] and DPR [34]. The configurations of the different retrievers and their MTEB scores[6]—a metric indicating their general sentence representation capability—are listed in Table 11. To isolate the impact of potentially different retrieved documents, we ensure that all models utilize the same *top-1* document. The performance is averaged over four question answering datasets. A general pattern is that embedding models with stronger sentence representation capabilities tend to further enhance the downstream performance. Remarkably, the Dragon model, despite being a BERT-base-sized retrieval-specific model, outperforms general text embedding models that are twice its size (BGE-Large).

Model		Model Size	Embedding Dim	Universal Embedding	MTEB Score	Average Performance	Resilience	Boost
Mistral-7b								
	w/o retrieval					37.2	-	-
	w retrieval					**46.3**	74.1%	**29.5%**
	w SFR	7B	4096	✓	67.56	44.5	82.3%	22.2%
	w E5-Mistral	7B	4096	✓	66.63	44.0	84.0%	20.6%
	w E5-Large	335M	1024	✓	62.25	42.1	80.2%	19.6%
xRAG	w BGE-Large	335M	1024	✓	64.23	41.6	78.2%	19.8%
	w BGE-Base	109M	768	✓	63.55	41.2	78.7%	18.9%
	w Dragon	109M	768	✗	-	42.1	**84.2%**	16.9%
	w DPR	109M	768	✗	-	40.5	77.4%	18.2%

Table 11: Ablation on different sentence embedding models.

---

[6]https://huggingface.co/spaces/mteb/leaderboard

F   Analysis on Mistral-7b

In Figure 6, we list the Resilience rate and Boost Rate on Mistral-7b model, which exhibit same pattern with Mixtral-8x7b model.
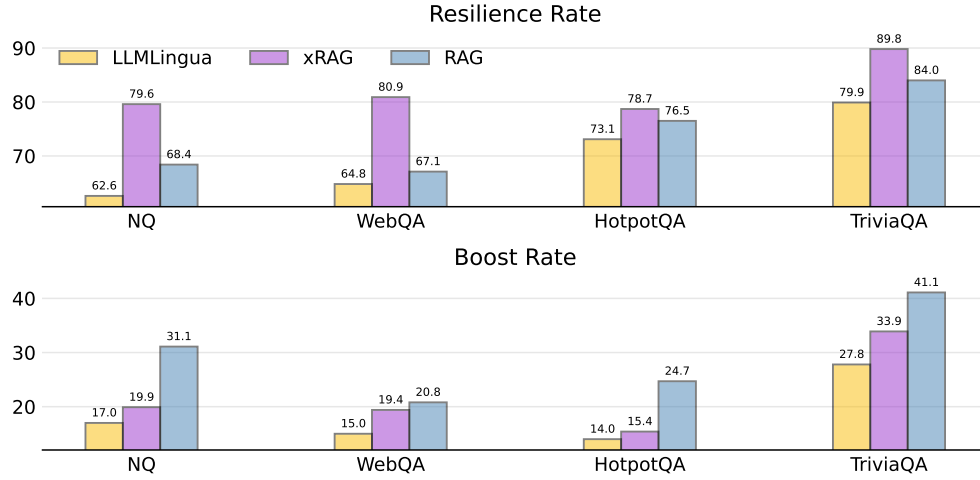
Figure 6: Robustness and effectiveness analysis on 4 QA datasets with Mistral-7b model.

G Limitations

We discuss the limitations of our framework as follows:

- In this work, we only consider the most commonly used retrieval system—single dense vector retrieval, while sparse retrieval methods such as BM25 or multi-vector retrieval methods like ColBERT are not included. We believe that combining these methods would be a promising direction for xRAG, as sparse vectors could complement dense vectors, and multi-vector retrieval would provide xRAG with more flexibility by not condensing all information into one token.

- Currently, xRAG delivers decent performance when a relevant document is fetched; however, it lags behind RAG by a considerable margin in tasks that require reasoning (such as HotpotQA and FactKG). One possible reason is that during the training phase of xRAG, reasoning-relevant data is not provided. How to make xRAG a better reasoner remains our future work.

- We only consider the *Top-1* retrieval setting, while ensembling multiple relevant documents has been shown to be effective for RAG systems due to the complementary information contained in *Top-K* documents. We believe there is potential advantage for xRAG to scale to multi-document settings, as the input length of xRAG for multi-documents scales by a factor of 1, while for RAG, it scales by the document length factor.

H   More interesting cases

- In Figure 7, we report a failure case of xRAG. In this case, retrieval alone is not enough to derive the final answer and the LLM is required to perform reasoning over retrieved document (the listed universities are all located in Switzerland).

- An interesting example is shown in Figure 8, when the retrieved document is a list of a characters in the book Discworld, the RAG model would respond with a fictional character, while xRAG generate the right answer by focusing on the relevant part of the document.

- In Figure 9, even when the retrieved document is relevant, RAG would still hallucinate while xRAG could generate the right answer based on the document.

- In Figure 10, the retriever mistakenly fetch the wrong document (Phantom of the Opera of interest is a music rather than a file) and RAG would be misled while xRAG remain robust to generate the correct answer.

Figure 7: Failure case of xRAG when reasoning is required to derive the final answer.

RAG

Refer to the background document and answer the question:

Background: List of Discworld characters | " Meaning ""Two-Fingered"" in Ephebian, Didactylos is a philosopher, based on Diogenes of Sinope, who comes into the stories in Small Gods. He lives in a barrel inside the wall of the palace of the Tyrant in Ephebe, crafting bespoke philosophies, axioms or aphorisms for scraps. Having apprenticed his nephew Urn as a philosopher, they frequently argue over the merits of natural philosophy, specifically Urn's development of steam power. Although one of the most popular philosophers of all time, Didactylos never earns the respect of his fellow philosophers, due to the fact that he thinks 'about the wrong things'. He has been pictured with a lantern though blind and is looking for an ""honest man"". ✓

Question: Which Greek philosopher was said to have lived inside a barrel?
- -
Output: Didactylos. ✗

w/o Retrieval

Answer the question:

Question: Which Greek philosopher was said to have lived inside a barrel?
- -
Output: No one. ✗

xRAG

Refer to the background document and answer the question:

Background: [X]

Question: Which Greek philosopher was said to have lived inside a barrel?
- -
Output: Diogenes of Sinope. ✓

Figure 8: xRAG correctly locates the relevant part in a long document by selecting Diogenes of Sinope as the answer rather than Didactylos, a fictional character in the book Discworld.

RAG

Refer to the background document and answer the question:

Background: List of Apollo missions | The Apollo program was the third United States human spaceflight program carried out by the National Aeronautics and Space Administration (NASA), which accomplished landing the first humans on the Moon from 1969 to 1972. During the Apollo 11 mission, astronauts Neil Armstrong and Buzz Aldrin landed their Apollo Lunar Module (LM) and walked on the lunar surface, while Michael Collins remained in lunar orbit in the command and service module (CSM), and all three landed safely on Earth on July 24, 1969. 5 subsequent Apollo missions also landed astronauts on the Moon, the last in December 1972. ✓

Question: What mission was the fifth human space flight of 'Project Apollo', the third human voyage to the Moon and the first manned mission to land on the Moon?
- -
Output: Apollo 15. ✗

w/o Retrieval

Answer the question:

Question: What mission was the fifth human space flight of 'Project Apollo', the third human voyage to the Moon and the first manned mission to land on the Moon?
- -
Output: Apollo 9. ✗

xRAG

Refer to the background document and answer the question:

Background: [X]

Question: What mission was the fifth human space flight of 'Project Apollo', the third human voyage to the Moon and the first manned mission to land on the Moon?
- -
Output: Apollo 11. ✓

Figure 9: xRAG correctly locates the relevant part in a long document while RAG would still hallucinate the wrong answer.

RAG

Refer to the background document and answer the question:

Background: Phantom of the Opera (1943 film) | The score was written by Edward Ward. For the opera sequences, Ward adapted music from Tchaikovsky's Symphony No. 4 as well as using themes by Chopin. He also composed an original theme, Lullaby of the Bells, which was heard in the film as the Phantom's piano concerto. ✗

Question: Who wrote the phantom of the opera music?

Output: Edward Ward. ✗

w/o Retrieval

Answer the question:

Question: Who wrote the phantom of the opera music?

Output: Andrew Lloyd Webber. ✓

xRAG

Refer to the background document and answer the question:

Background: [X]

Question: Who wrote the phantom of the opera music?

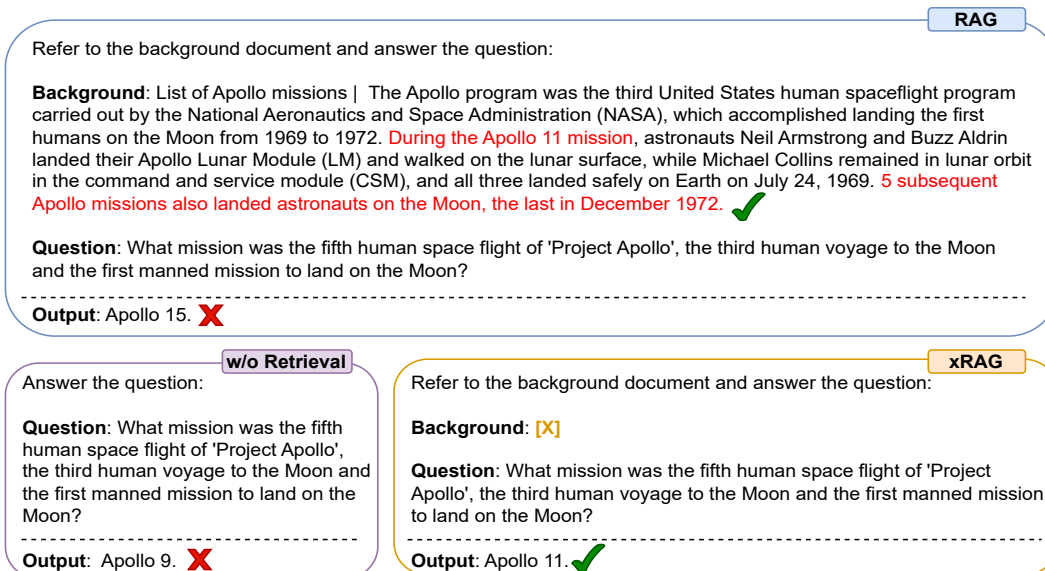--
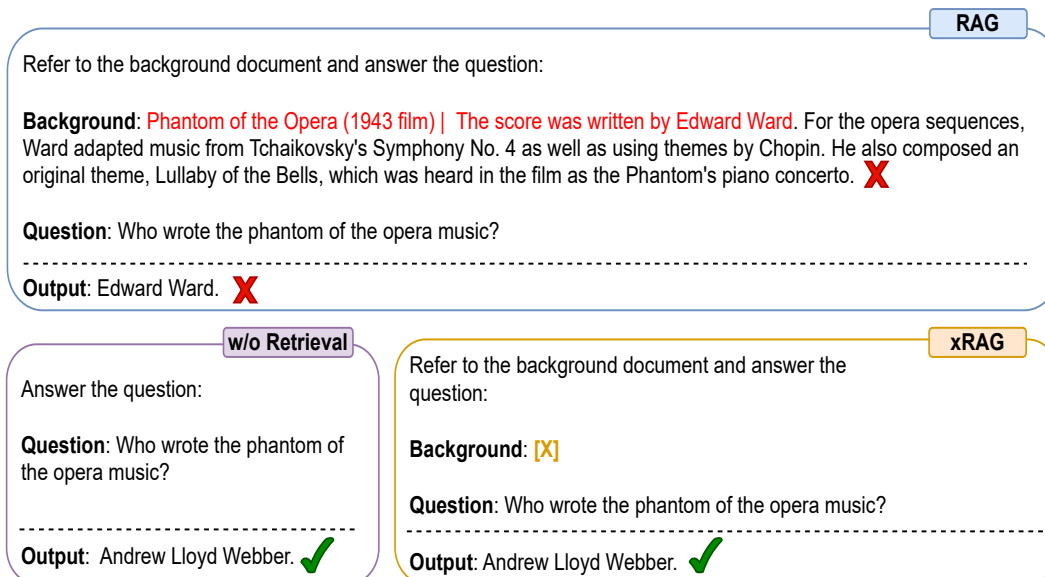
Output: Andrew Lloyd Webber. ✓

Figure 10: xRAG correctly locates the relevant part in a long document while RAG would still hallucinate the wrong answer.