ConvX: A Lightweight Converter to Bridge Indexed Dense Representations and Large Language Models for Retrieval-Augmented Generation

Bonggeun Choi, Keunha Kim, Junho Han, Youngjoong Ko*

Sungkyunkwan University, Repulic of Korea

bonggeun.choi818@gmail.com, keunhakim98@gmail.com,
hjunho2357@gmail.com, yjko@skku.edu

Abstract

Retrieval-Augmented Generation (RAG) has significantly advanced open-domain question answering systems by incorporating external knowledge into large language models. Despite its effectiveness, existing RAG pipelines suffer from critical efficiency limitations. In particular, modern transformer-based generators exhibit quadratic or higher computational complexity with respect to input sequence length and hidden dimensionality, leading to substantial inference latency as model scales and contextual inputs increase. This issue is exacerbated in RAG settings, where retrieved contexts substantially expand the input prompt. To alleviate this challenge, we propose an effective compression-based RAG framework, ConvX[1], that directly leverages indexed dense representations produced by a retriever, entirely substituting to long text contexts. Our approach expands a single dense representation into a fixed number of memory slots using a lightweight converter to provide rich lexical information. This design enables efficient knowledge integration while significantly reducing input length and computational overhead. Empirical evaluations demonstrate that the proposed model achieves competitive performances compared to the existing state-of-the-art model that uses a large ad-hoc context compressor, while offering substantially improved inference efficiency.

1 Introduction

Large language models (LLMs) have recently driven substantial progress in open-domain question answering systems, owing to their ability to generate contextually coherent responses (Minaee et al., 2025; Zhao et al., 2025). Despite these advances, LLMs remain fundamentally constrained by the static nature of their training data, often struggling with queries that require specialized knowledge, fine-grained factual details, or up-to-date information. These limitations hinder their reliability in real-world applications where factual grounding and adaptability are critical (Huang et al., 2025).

Retrieval-Augmented Generation (RAG) addresses this challenge by incorporating external knowledge through retrieval, significantly improving factual accuracy and reducing hallucinations (Lewis et al., 2020). As LLMs continue to scale and support longer context windows, RAG systems can integrate richer evidence, enabling more sophisticated reasoning (Gao et al., 2024). However, this increased contextual capacity comes with a substantial computational cost. Transformer-based architectures incur quadratic complexity with respect to input length, leading to severe inference latency when large volumes of retrieved contexts are appended to the prompt (Zhu et al., 2024).

To mitigate this issue, recent studies have explored compressing contexts before passing them to the generator. Recent embedding-based compression methods represent a long context using a small set of dense vectors (a.k.a. memory slots) through an ad-hoc compressor, offering higher compression rates. However, this approach encounters critical limitations. First, they often overfit to their training datasets, implicitly enforcing generator behavior such that memory slots function as soft prompts rather than faithful knowledge carriers. Second, they introduce a double-encoding problem, as passages are encoded both by the retriever and again by the compressor (Chevalier et al., 2023; Ge et al., 2024). Although precomputing and indexing memory slots can reduce online computation, this incurs substantial storage overhead due to the increased number and dimensionality of stored vectors (Rau et al., 2025). Another line of work (Cheng et al., 2024) partially addresses these inefficiencies by directly projecting retriever embeddings into the LLM embedding space and using

*Corresponding author

[1] https://github.com/NLPlab-skku/ConvX

Findings of the Association for Computational Linguistics: ACL 2026, pages 36587–36599

the projected representations as alternatives to textual inputs, thereby eliminating double encoding. However, representing an entire passage with a single dense vector is often insufficient to capture fine-grained lexical and compositional information, especially for long or information-rich contents.

In this paper, we propose a simple yet effective compression-based RAG framework called **ConvX** that alleviates the above limitations. Like prior index-based methods, our approach reuses dense representations produced by the retriever to avoid double encoding, but instead of relying on a single projected embedding, we expand each retriever representation into a fixed number of memory slots using a lightweight converter. These memory slots are designed to retain diverse and complementary lexical signals from the original passage, enabling the generator to access richer and explicit information while maintaining a compact input representation. Extensive experiments demonstrate that our method achieves competitive or sometimes better performance compared to existing ad-hoc compressor-based approaches, while offering significantly improved efficiency in the RAG environment.

2 Related Work

Text-Based Compression Text-based compression aims to reduce the length of input prompts by eliminating redundant words or extracting the salient spans directly from the raw text. For instance, LLMLingua (Jiang et al., 2023) utilizes a small language model to prune redundant tokens based on their perplexity. In contrast, RECOMP (Xu et al., 2023) introduces learnable compressors that represent the context into a dense summary or select salient sentences via end-to-end training. However, these approaches often face limitations in compression ratios, failing to fully address the long-context problems.

Embedding-Based Compression Embedding-based compression methods achieve higher compression ratios than text-based approaches by condensing contextual information into continuous vector representations. For example, AutoCompressor (Chevalier et al., 2023) progressively constructs dense summary vectors by recursively conditioning on previously generated summaries, ultimately using the collection of these vectors as a compressed representation. In contrast, ICAE (Ge et al., 2024), PISCO (Louis et al., 2025), and PCC (Dai et al., 2025) compress long contexts into a fixed set of compact memory slots in a single pass. Despite their effectiveness in achieving high compression ratios, these approaches typically rely on an auxiliary LLM as an ad-hoc compressor, which introduces additional computational overhead and reduces overall efficiency.

Meanwhile, xRAG (Cheng et al., 2024) directly injects precomputed retriever embeddings into a target LLM by projecting them into the model's embedding space via a lightweight adapter. While this design effectively alleviates the double-encoding problem, it remains limited in representational capacity, as each document is represented by a single vector and typically only the top-1 retrieved result is incorporated, which can lead to information loss. Inspired by the direct embedding injection paradigm of Cheng et al. (2024), **ConvX** addresses this limitation by expanding each retrieved embedding into multiple representations through a lightweight converter, thereby capturing richer lexical and semantic information originally contained in the source documents.

3 Methodology

3.1 Motivation

Recent dense retrievers pretrained on large-scale corpora have been shown to encode rich lexical information within their dense representations (Wang et al., 2023; Xiao et al., 2022; Ma et al., 2024). Motivated by this property, we hypothesize that if such latent lexical cues can be effectively recovered or aligned when mapping retriever embeddings into a generator's embedding space, the generator can better interpret and utilize retrieved knowledge. To this end, we propose transforming each retriever embedding into multiple memory slots via a lightweight converter, explicitly designed to preserve passage-level lexical information and convey it to the target LLM in a more expressive form.

Another key consideration in our architectural design is minimizing dependency on a specific retriever. Since our framework directly consumes retriever embeddings, naively replacing the retriever would necessitate retraining the entire model. To address this, we decouple the converter from the target LLM, such that only the lightweight converter needs to be retrained when the retriever is changed. Accordingly, our training pipeline consists of three stages: the converter is trained in the first stage and subsequently frozen, enabling efficient adaptation

to different retrievers without modifying the LLM. Nevertheless, ensuring robust generalization across diverse retrievers remains an open challenge, which we leave for future work.

3.2 Stage 1: Pretraining Converter

The converter serves two purposes: (i) bridging the dimensional mismatch between the retriever and the target LLM, and (ii) preserving lexical information from the original passage. Rather than collapsing all lexical signals into a single dense vector, we expand each retriever embedding into multiple memory slots. These memory slots are explicitly supervised to encode passage-level vocabularies, such that the vocabularies are distributed and preserved across the slots. This design allows the converted representations to retain richer lexical coverage while remaining compact and suitable for efficient generation.

Given a retriever-produced passage embedding, we first partition it into M chunks:

$$\mathrm{H} \in \mathbb{R}^{M \times D_r'}, \quad D_r' = D_r / M, \tag{1}$$

where D_r is the retriever embedding dimension. This exposes multiple semantic subspaces of the passage instead of collapsing all information into a single vector. The chunked embeddings are projected into the LLM embedding space via the converter:

$$\tilde{\mathrm{Z}} = \mathrm{Converter}(\mathrm{H}) \in \mathbb{R}^{M \times D_g}, \tag{2}$$

where D_g is the LLM hidden dimension. The converter is a three-layer network whose intermediate layer applies multi-head self-attention over the M memory slots, enabling interaction among slots and preventing gradient collapse.

Vocabulary Reconstruction To encourage the converted embeddings to recover lexical information from the original passage, we compute token-level probability distributions for each memory slot. Specifically, the probabilities $\mathrm{P}^{\text{conv}} \in \mathbb{R}^{M \times V}$ are obtained by projecting the converted embeddings through the LLM's language modeling head, followed by a softmax over the vocabulary of size V. We then aggregate these slot-level distributions into a single passage-level vocabulary distribution using max pooling across the M memory slots:

$$p^{\text{conv}} = \max_{m \in [1,M]} (\mathrm{softmax}(\mathrm{LMHead}(\tilde{\mathrm{Z}})))_m. \tag{3}$$

Then, let $\mathrm{y} \in \{0,1\}^V$ indicate token presence in the original passage. The vocabulary reconstruction loss is defined as

$$\mathcal{L}_{\text{nll}} = -\frac{1}{\|\mathrm{y}\|_1} \sum_{i=1}^{V} y_i \cdot \log p_i^{\text{conv}}. \tag{4}$$

where normalization by $\|\mathrm{y}\|_1$ prevents the loss magnitude from diminishing due to the large vocabulary size.

Auxiliary Alignment We introduce two auxiliary losses to align the converted embeddings with the LLM's internal representations. First, we compute the LLM hidden states for the original passage X and calculate mean squared error (MSE) loss:

$$\mathrm{Z} = \theta_{\text{LLM}}(X) \in \mathbb{R}^{|X| \times D_g}, \tag{5}$$

$$\mathcal{L}_{\text{mse}} = \|\bar{\tilde{\mathrm{z}}} - \bar{\mathrm{z}}\|_2^2, \tag{6}$$

where $\bar{\tilde{\mathrm{z}}}$ and $\bar{\mathrm{z}}$ are averaging vectors of $\tilde{\mathrm{Z}}$ and Z over M and $|X|$, respectively.

Second, we align vocabulary distributions using KL divergence. The LLM-induced distribution is

$$p^{\text{llm}} = \max_{m \in [1,M]} (\mathrm{softmax}(\mathrm{LMHead}(\mathrm{Z})))_m, \tag{7}$$

and the KL loss is defined as

$$\mathcal{L}_{\text{kl}} = \mathrm{KL}(p^{\text{conv}} \parallel p^{\text{llm}}). \tag{8}$$

The two objectives encourage to align the global semantic representations of the converted memory slots and to produce vocabulary activations consistent with those induced by the LLM. The overall converter pretraining objective is

$$\mathcal{L} = \mathcal{L}_{\text{nll}} + \mathcal{L}_{\text{mse}} + \mathcal{L}_{\text{kl}}. \tag{9}$$

All loss terms are equally weighted, which we found to work best empirically. After Stage 1, the converter is frozen in subsequent stages.

3.3 Stage 2: Pretraining the LLM

In Stage 2, we pretrain the target LLM to effectively consume memory slots $\tilde{\mathrm{Z}}$, which differ from native token embeddings. We design four language modeling tasks that expose the LLM to both single- and multi-passage memory inputs. In the following subsections, we denote the LLM log-likelihood as

$$f_\theta(\cdot) = \log P_{\theta_{\text{LLM}}}(\cdot). \tag{10}$$

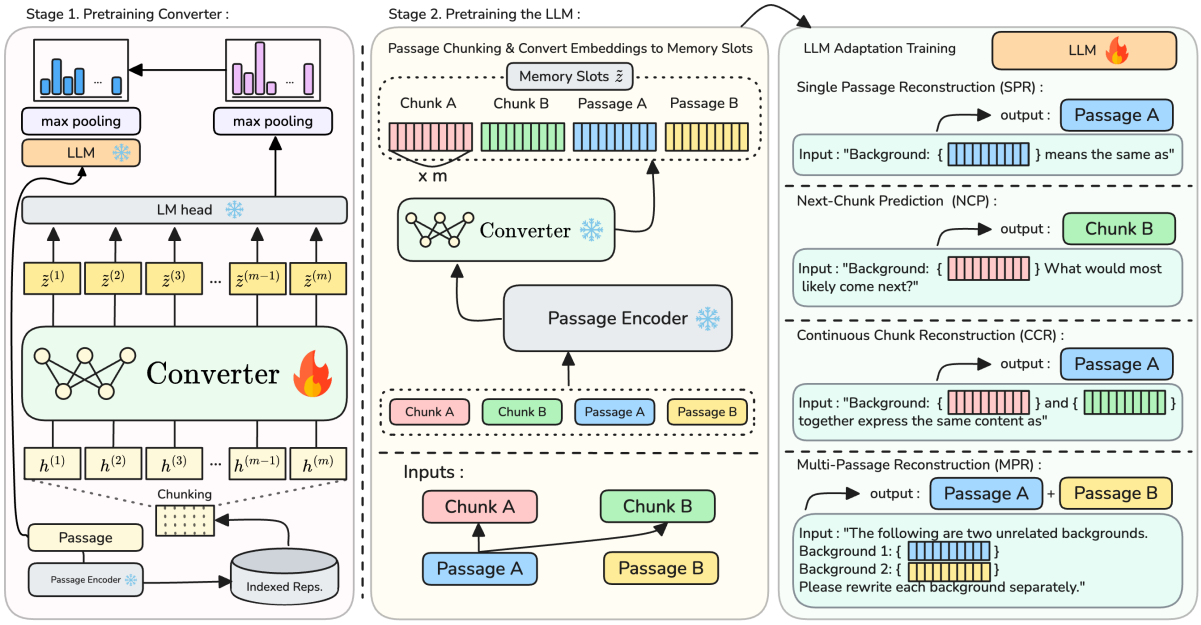

Figure 1: The illustration of pretraining the converter and decoder. Note that the converter is frozen after the pretraining so that the converted representations sustain the lexical information while pretraining and finetuning the decoder.

Single-Passage Processing Following prior works (Cheng et al., 2024; Ge et al., 2024; Rau et al., 2025; Dai et al., 2025), we first construct two tasks based on a single passage.

Single Passage Reconstruction (SPR). Given a passage $X = x_1, \ldots, x_{|X|}$, we encode it with the dense retriever and convert the resulting embedding using the pretrained converter. The LLM is trained to reconstruct the original passage conditioned on $\tilde{\mathrm{Z}}$:

$$\mathcal{L}_{\text{spr}} = -\sum_{t=1}^{|X|} f_\theta(x_t|\tilde{\mathrm{Z}}, x_{<t}). \quad (11)$$

Next-Chunk Prediction (NCP). We split the passage into two contiguous chunks, $X_{\mathrm{A}} = \{x_1, ..., x_j\}$ and $X_{\mathrm{B}} = \{x_{j+1}, ..., x_{|X|}\}$. Memory slots $\tilde{\mathrm{Z}}_{\mathrm{A}}$ are constructed only from the preceding chunk. The LLM then continues the content in X_{B}:

$$\mathcal{L}_{\text{ncp}} = -\sum_{t=j+1}^{|X|} f_\theta(x_t|\tilde{\mathrm{Z}}_{\mathrm{A}}, x_{<t}). \quad (12)$$

Multi-Passage Processing To further improve robustness when handling multiple retrieved passages, we introduce two additional tasks involving multiple sets of memory slots.

Continuous Chunk Reconstruction (CCR). We encode the both chunks X_{A} and X_{B} independently and convert them into memory slots: $\tilde{\mathrm{Z}}_{\mathrm{A}}$ and $\tilde{\mathrm{Z}}_{\mathrm{B}}$. The LLM reconstructs the full passage using both memory sets:

$$\mathcal{L}_{\text{ccr}} = -\sum_{t=1}^{|X|} f_\theta(x_t|\tilde{\mathrm{Z}}_{\mathrm{A}}, \tilde{\mathrm{Z}}_{\mathrm{B}}, x_{<t}). \quad (13)$$

Multi-Passage Reconstruction (MPR). To pre-simulate retrieval scenarios where passages are semantically unrelated, we sample two distinct passages X_1 and X_2 and obtain their corresponding memory slots $\tilde{\mathrm{Z}}_1$ and $\tilde{\mathrm{Z}}_2$. The LLM is trained to reconstruct the concatenated passage $X_{\text{cat}} = [X_1; X_2]$ conditioned on both memory inputs:

$$\mathcal{L}_{\text{mpr}} = -\sum_{t=1}^{|X_{\text{cat}}|} f_\theta(x_t|\tilde{\mathrm{Z}}_1, \tilde{\mathrm{Z}}_2, x_{<t}). \quad (14)$$

The overall pretraining objective for the LLM is the sum of the four task losses:

$$\mathcal{L} = \mathcal{L}_{\text{spr}} + \mathcal{L}_{\text{ncp}} + \mathcal{L}_{\text{ccr}} + \mathcal{L}_{\text{mpr}}. \quad (15)$$

During pretraining, tasks are sampled within each batch according to predefined ratios. For stability, single-passage tasks dominate the early training phase, while the proportion of multi-passage tasks is gradually increased as training progresses. The overall process of pretraining the converter and LLM is illustrated in Figure 1.

3.4 Stage 3: Self-Distillation

Finally, we adapt the pretrained LLM to downstream RAG tasks. Each question is paired with five retrieved passages, which may include irrelevant or noisy contexts. Training under this setting exposes the target LLM to realistic RAG conditions, encouraging robustness and improved generalization in the presence of imperfect retrieval.

Since the target LLM's original generation capability may be partially degraded during pretraining with memory-slot inputs and the language modeling objective, we restore its generative behavior through self-distillation (Yang et al., 2024). Specifically, we generate pseudo answers using the vanilla RAG conditioned on the original textual passages and use these outputs as supervision, instead of relying on conventional supervised fine-tuning (SFT) that depends on short, gold-standard answers. This self-distillation strategy enables the target LLM to recover fluent and faithful generation while maintaining robustness even when textual inputs are replaced by memory slots.

4 Experiments

4.1 Experimental Setup

Implementation Details We initialize the target LLM from Mistral-7B-Instruct-v0.2[2] and apply LoRA (Hu et al., 2022). Specific hyperparameters for training full ConvX are reported in Appendix B. The trained model checkpoint[3] is publicly available.

Following the RAG environment of Rau et al. (2025), SPLADE-v3 (Lassance et al., 2024) is employed to retrieve passages but reranking is omitted due to unspecified configurations. Instead of on-the-fly retrieval, we construct the retrieval index in advance and re-encode the retrieved passages with SFR retriever (Meng, 2024) for convenience, which is aligned with Cheng et al. (2024).

Datasets We pretrain our model on 2.5M passages from Wikipedia-KILT (Petroni et al., 2021). For fine-tuning, we use MultiQA (Rau et al., 2025), which comprises a diverse set of QA benchmarks, including NQ (Karpukhin et al., 2020), MS-MARCO (Bajaj et al., 2018), AdversarialQA (Bartolo et al., 2020), HotpotQA (Yang et al., 2018), WikiQA (Yang et al., 2015), SCIQ (Welbl et al., 2017), ASQA (Stelmakh et al., 2022), TriviaQA (Joshi et al., 2017), FreebaseQA (Jiang et al., 2019), and SQuAD (Rajpurkar et al., 2018).

To ensure reliable supervision during self-distillation, we filter out training samples for which the answer generated by a vanilla RAG model does not contain the annotated gold answer. For evaluation, we report results on NQ, TriviaQA, HotpotQA, ASQA, and PopQA (Mallen et al., 2023). Furthermore, the passage collection[4] and the dataset[5] paired with retrieved passages used for both training and evaluation are publicly released on Huggingface to facilitate reproducibility and enable fair comparison in future work.

Baselines We compare our approach against five representative context compression methods: AutoCompressor (Chevalier et al., 2023), ICAE (Ge et al., 2024), xRAG (Cheng et al., 2024), PCC-lite (Dai et al., 2025), and PISCO (Louis et al., 2025), adopting their own configuration for compression and generation. The specifics for reproducing the baselines are detailed in Appendix A.

Evaluation Metrics We evaluate generated answers using the span exact match (spanEM) metric. Following prior work (Dai et al., 2025; Rau et al., 2025), we note that large language models often produce long, free-form outputs, making strict exact string matching impractical. Instead, we adopt a containment-based variant of exact match, which deems a prediction correct if the gold answer string appears as a contiguous span within the generated output.

4.2 Main Results

Table 1 summarizes the evaluation results across five open-domain question answering benchmarks. Overall, the proposed model, **ConvX**, demonstrates competitive performance compared to the state-of-the-art model PISCO. At both compression rates, ConvX consistently outperforms xRAG, which adopts a similar compression-oriented strategy, substantially narrowing the gap with the uncompressed RAG upper bound while achieving significant efficiency gains.

Notably, increasing the compression ratio to 128× results in only marginal performance degradation, suggesting that the quality and informative-

[2] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

[3] https://huggingface.co/bgchoi/convx-16x-mistral

[4] https://huggingface.co/datasets/bgchoi/kilt-256

[5] https://huggingface.co/datasets/bgchoi/multiqa_top5

	Models	Comp. ratio	Datasets					Average
			NQ	TriviaQA	HotpotQA	ASQA	PopQA	
Reference	RAG$^{\triangle}$	-	0.609	0.861	0.423	0.525	0.528	0.589
	LLM$^{\triangledown}$	-	0.405	0.748	0.295	0.324	0.278	0.410
Baseline	AutoCompressor	20×	0.337	0.590	0.212	0.248	0.259	0.329
	ICAE	10×	0.317	0.649	0.209	0.246	0.292	0.342
	xRAG	256×	0.428	0.783	0.321	0.337	0.312	0.436
	PCC-lite	16×	0.290	0.623	0.170	0.247	0.284	0.323
		128×	0.227	0.620	0.185	0.201	0.213	0.289
	PISCO	16×	**0.562**	0.831	**0.385**	**0.449**	**0.481**	**0.542**
Ours	ConvX	16×	0.534†	**0.846**	0.372†	0.399†	0.391†	0.508
		128×	0.537†	0.839	0.367†	0.394†	0.387†	0.505

Table 1: Performance comparison between baseline methods and the proposed model. RAG$^{\triangle}$ and LLM$^{\triangledown}$ denote the upper and lower performance bounds, corresponding to answer generation without compression and without contextual input, respectively. Higher compression ratio means more efficient generation. The best results are **bolded** and the second ones are underlined. Results marked with † indicate they are not statistically significant (p>0.05).

ness of the memory slots are more critical than their quantity. This finding highlights the importance of preserving salient lexical cues within memory slots for effective knowledge transfer. Qualitative examples of such lexical cues are provided in Appendix C.

A key finding is that several baseline methods fail to surpass the vanilla LLM lower bound, suggesting that their compressed representations often introduce noise rather than useful knowledge. We attribute this behavior to a mismatch between their training objectives and the requirements of RAG inference. Specifically, these methods are optimized to compress a single or a small number of coherent contexts jointly, rather than independently encoding multiple disjoint passages produced by retrieval. As a result, they struggle to form stable and informative representations when faced with fragmented or noisy retrieval outputs.

In contrast, approaches that adopt noisy retrieval environment for the training, such as xRAG, PISCO, and ConvX exhibit substantially greater robustness. ConvX, in particular, avoids the instability of ad-hoc context compression by directly reusing pre-computed dense retriever embeddings and explicitly preserving lexical information at the passage level. Memory slots are constructed independently for each retrieved passage, and the converter is frozen after Stage 1 to decouple lexical knowledge preservation from task adaptation. During downstream training, only the target LLM is adapted using retrieved passages. This design allows the converter to function as a stable lexical projection module, mitigating interference from noisy retrieval and enabling memory slots to serve as faithful carriers of passage-level lexical information. Consequently, ConvX achieves stronger and more robust generalization in realistic RAG environments.

	Model	Datasets	
		NQ	AdversarialQA
Relevant-Only	AutoCompressor	0.449	0.246
	ICAE	0.655	0.405
	xRAG	0.482	0.290
	PISCO	**0.723**	**0.430**
	PCC-lite 16×	0.528	0.340
	ConvX 16×	0.600	0.341
Top-1	AutoCompressor	0.348	0.137
	ICAE	0.393	0.155
	xRAG	0.405	0.175
	PISCO	0.471	0.204
	PCC-lite 16×	0.341	0.130
	ConvX 16×	**0.488**	**0.206**

Table 2: Answer generation performance under a single-passage setting. All evaluated datasets draw external knowledge from Wikipedia, which is included in the pretraining data of all baseline models, ensuring a fair comparison.

4.3 Considerations on Ad-hoc Compressors

Instability in RAG Environment Retrieved passages in RAG systems are often noisy, partially

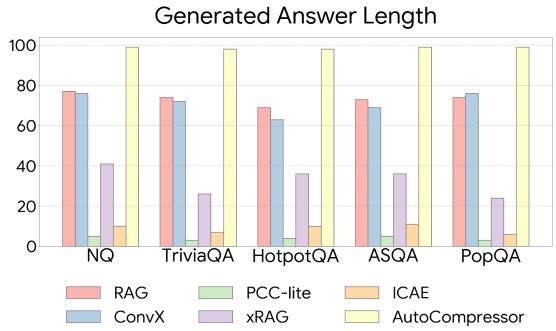

Figure 2: The comparisons for generated answer length between baselines and ours across the datasets.

Ablation	Datasets				
	NQ	TriviaQA	HotpotQA	ASQA	PopQA
LLM-Finetuned	0.489	0.789	0.326	0.368	0.321
ConvX	0.534	0.846	0.372	0.399	0.391
w/o Stage 1	0.549	0.847	0.381	0.405	0.415
w/o Stage 2	0.529	0.826	0.352	0.388	0.368
w/o Stage 1 & 2	0.525	0.815	0.343	0.385	0.358
w/o Self-Distil	0.511	0.776	0.320	0.317	0.324
w/o Compression	0.602	0.867	0.434	0.520	0.503
w/o Context	0.428	0.757	0.314	0.321	0.275

Table 3: Performances of the ablation study. Each component shows positive effect to construct full ConvX.

redundant, or entirely irrelevant. However, most existing compression-based methods are designed to compress a single long context into a fixed set of memory slots under the assumption that the input context is guaranteed to be relevant or explicitly constructed to contain the answer, which is rarely holds in realistic RAG environments. For instance, AutoCompressor is trained on open-ended generation data without retrieval noise; ICAE relies on its proprietary Prompt-with-Context (PwC) dataset, which closely resembles single-context machine reading comprehension (MRC) settings; and PCC-lite is fine-tuned on MRC-style data and evaluated in RAG settings using only positive passages. Consequently, these models are never exposed to irrelevant or competing evidence during training, limiting their robustness under real-world retrieval conditions.

Table 2 reports answer generation performance when only a single passage is provided, explicitly matching the compression configuration assumed by these baselines. When the passage is guaranteed to be relevant, several compressor-based methods achieve strong performance, demonstrating their ability to exploit idealized, answer-containing contexts. In contrast, when evaluated using a retrieved passage, which may be noisy or irrelevant, their performance degrades substantially. This gap reveals a critical limitation: even in a single-passage setting, ad-hoc compressors struggle to generalize once retrieval noise is introduced. These results highlight the importance of training RAG models under noisy retrieval conditions to ensure robust and reliable performance at inference time.

Ad-hoc Compressor as Soft Prompts Generator As shown in Figure 2, most baseline methods generate unusually short answers across all datasets, even without fine-tuning the target LLM or imposing explicit length constraints through instructions. In particular, compressor-based approaches such as PCC-lite and ICAE consistently produce extremely short outputs. xRAG also generates shorter answers than standard RAG, despite being regularized by the vanilla LLM's logit distributions via a KL-divergence loss. These observations suggest that compressed memory slots encode not only contextual information from retrieved passages, but also strong implicit priors inherited from the compressor's training data, such as expected answer length and response style, which in turn constrain the LLM's generation behavior.

In contrast, ConvX consistently produces answer lengths comparable to those of standard RAG, while avoiding the excessive verbosity exhibited by AutoCompressor, which is trained with open-ended generation objectives. This robustness arises from our training strategy: after pretraining the LLM to consume memory slots, we adaptively fine-tune the target LLM via self-distillation, using answers generated by the vanilla LLM as supervision. This procedure explicitly restores the LLM's native generation behavior, which may be weakened during Stage 2 pretraining, while ensuring that memory slots serve as faithful knowledge carriers rather than soft prompts that dictate output length.

4.4 Analysis

Ablation Study Table 3 presents the results of our ablation study, analyzing the contribution of each component in the ConvX training pipeline. Removing Stage 2 results in consistent performance degradation across all datasets, indicating that explicitly exposing the LLM to memory-slot inputs during pretraining is essential for effectively utilizing the converted representations at inference time. Without Stage 2, the LLM is required to adapt to memory slots only during task-specific

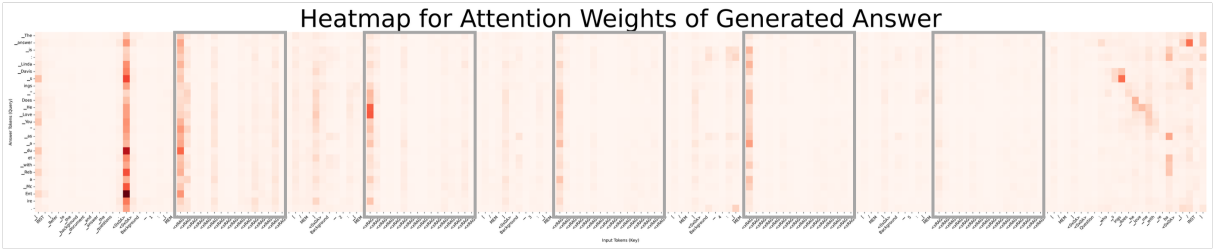

Figure 3: Visualization of attention weights between the generated answer (y-axis) and the input prompt (x-axis). The gray box indicates the position of memory slots.

fine-tuning, which limits its ability to fully exploit the lexical information preserved by the converter. We do not consider removing Stage 1, as it is a fundamental component of our method that equips the converter with the ability to recover vocabulary-level signals from retriever embeddings.

Eliminating the self-distillation stage leads to a substantial drop in performance, highlighting its importance in restoring the LLM's native generation behavior. Training solely with annotated short answers biases the model toward shorter, extractive outputs, whereas self-distillation helps preserve the original generative characteristics of the LLM while adapting it to compressed inputs. Furthermore, when retrieved passages are provided in raw text without compression, our model achieves performance comparable to RAG, which serves as an upper-bound reference. The relatively small gap suggests that our training strategy effectively preserves the LLM's core reasoning and generation capabilities.

Interestingly, when Stage 1 is removed and the converter is instead trained jointly during Stages 2 and 3 without explicitly preserving lexical information, the model achieves the best performance. However, this setting requires retraining the LLM whenever the retriever is changed, which contradicts our architectural objective of minimizing retriever dependency, as discussed in Section 3.1.

We also finetuned a vanilla LLM on the same training dataset to examine whether the model can learn answer distributions without access to retrieved documents. While this approach yields performance improvements, it does not surpass ConvX. Additionally, ConvX demonstrates that the original LLM's generation behavior is well preserved, as evidenced by results in the settings without compression and without context, which approximate the upper bound (RAG) and the lower bound (LLM-only), respectively.

Informative Knowledge Carrier Our goal is for the target LLM to actively rely on memory slots during answer generation, with the expectation that these slots convey lexical information from the original passages. To examine whether this behavior emerges in practice, we analyze the attention weights assigned to input prompts when generating answers. We select a representative example in which the vanilla LLM fails to produce a correct answer, while ConvX succeeds.

As illustrated in Figure 3, the generated answer attends substantially to the memory slots, indicating that the model effectively leverages them as knowledge inputs. Notably, memory slots corresponding to the fifth retrieved passage receive minimal attention, consistent with the fact that this passage is irrelevant to the question. This behavior suggests that the model can selectively attend to informative memory slots while ignoring noisy or irrelevant contexts.

A further observation is that, within each passage, the first memory slot consistently receives the highest attention weight. This implies that the LLM does not uniformly rely on all memory slots, but instead focuses on specific slots that carry the most salient information. This finding aligns with the results in Section 4.2, which show diminishing returns beyond a certain compression ratio, and provides additional evidence that ConvX learns to encode and exploit lexical cues efficiently through a small number of memory slots.

Computational Efficiency Figure 4 illustrates the trade-off between compression latency and computational cost (GFLOPs). Compared to xRAG, ConvX with a $16\times$ compression ratio incurs slightly higher computational overhead due to the multi-head attention operations within the converter. However, this overhead results in only a modest increase in latency, keeping ConvX within a practical efficiency regime. In contrast, other

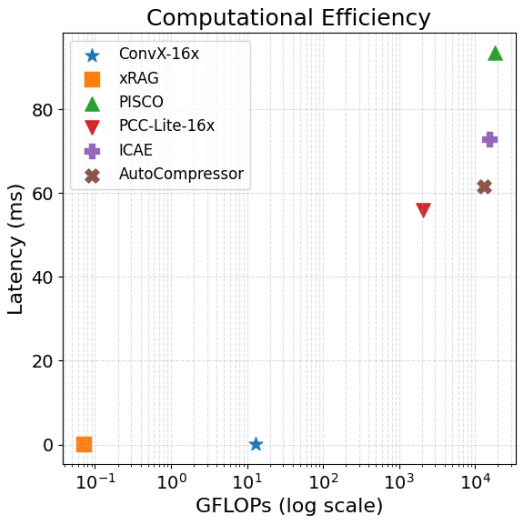

Figure 4: Computational efficiency comparison across models, measured in GFLOPs (log-scale) and latency for compression.

baseline methods exhibit substantially higher compression latency and computational cost because they rely on transformer-based compressor models. Although compression is most beneficial when generating long responses, many baseline models produce consistently short answers, as discussed in Section 4.3. In such cases, the additional compression overhead outweighs its potential benefits, further undermining practical efficiency, particularly in latency-sensitive RAG settings.

5 Conclusion

In this paper, we proposed ConvX, a compression-based RAG framework that bridges indexed dense retriever representations and LLMs. Instead of compressing retrieved text, ConvX directly employs retriever embeddings by converting them into a fixed set of memory slots, enabling richer knowledge integration while maintaining a compact input representation. These memory slots are explicitly trained to preserve lexical cues from the original passages, allowing the generator to effectively exploit fine-grained context information. Extensive experiments demonstrate that ConvX consistently outperforms existing context compression methods, particularly in realistic RAG settings where retrieved passages are noisy, redundant, or irrelevant, avoiding cross-passage interference and maintaining stable lexical representations throughout downstream adaptation.

We believe our results highlight the effectiveness of lightweight converters for directly reusing retriever embeddings as knowledge sources beyond retrieval. An important direction for future work is to develop more sophisticated alignment strategies between dense retriever representations and LLM embedding spaces, further improving robustness and generalization across diverse retrieval models and tasks.

Limitations

As discussed in Appendix C, not all memory slots consistently capture informative vocabularies. This observation indicates that the target LLM selectively attends to a subset of memory slots that convey salient lexical information, while ignoring less informative ones. Consequently, memory slots do not contribute equally to generation, resulting in redundancy in the current representation. Improving the specialization of individual memory slots so that each encodes complementary and informative lexical cues more consistently remains an important direction for future work.

In addition, the current implementation of ConvX is evaluated only with the SFR retriever index, which limits its immediate compatibility with other retrieval models. However, because the converter is frozen after Stage 1, it can in principle be trained independently for different retrieval indices and then plugged into the finetuned LLM. Exploring such plug-and-play adaptation by pretraining lightweight converters for diverse retrievers is another promising direction for future work.

Acknowledgments

This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00350379, 40), Institute of Information & communications Technology Planning & evaluation (IITP) grant funded by the Korea government (MSIT) (No. IITPRS-2022-II220680, 40), and Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2019-II190421, Artificial Intelligence Graduate School Program(Sungkyunkwan University), 20).

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh

Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the AI: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678.

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 *Advances in Neural Information Processing Systems*, volume 37, pages 109487–109516. Curran Associates, Inc.

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

Yuhong Dai, Jianxun Lian, Yitian Huang, Wei Zhang, Mingyang Zhou, Mingqi Wu, Xing Xie, and Hao Liao. 2025. Pretraining context compressor for large language models with embedding-based memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28715–28732, Vienna, Austria. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv: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. *Preprint*, arXiv:2307.06945.

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 *International Conference on Learning Representations*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2).

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore. Association for Computational Linguistics.

Kelvin Jiang, Dekun Wu, and Hui Jiang. 2019. FreebaseQA: A new factoid QA data set matching trivia-style question-answer pairs with Freebase. In *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. Association for Computational Linguistics.

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

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

Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. Splade-v3: New baselines for splade. *Preprint*, arXiv:2403.06789.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Maxime Louis, Hervé Déjean, and Stéphane Clinchant. 2025. PISCO: Pretty simple compression for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15506–15521, Vienna, Austria. Association for Computational Linguistics.

Guangyuan Ma, Xing Wu, Zijia Lin, and Songlin Hu. 2024. Drop your decoder: Pre-training with bag-of-word prediction for dense passage retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 1818–1827, New York, NY, USA. Association for Computing Machinery.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Rui Meng. 2024. Sfr-embedding-mistral: Enhance text retrieval with transfer learning. Blog post.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2025. Large language models: A survey. *Preprint*, arXiv:2402.06196.

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. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

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

David Rau, Shuai Wang, Hervé Déjean, Stéphane Clinchant, and Jaap Kamps. 2025. Context embeddings for efficient answer generation in retrieval-augmented generation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, WSDM '25, page 493–502, New York, NY, USA. Association for Computing Machinery.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. ASQA: Factoid questions meet long-form answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8273–8288, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023. SimLM: Pre-training with representation bottleneck for dense passage retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2244–2258, Toronto, Canada. Association for Computational Linguistics.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-training retrieval-oriented language models via masked auto-encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 538–548, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. 2024. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1028–1043, Bangkok, Thailand. Association for Computational Linguistics.

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2025. A survey of large language models. *Preprint*, arXiv:2303.18223.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

A Details for Baselines

For AutoCompressor, ICAE, and PCC-lite, retrieved passages are concatenated and compressed jointly, as this strategy yields better performance than compressing each passage independently.

- **AutoCompressor**[6] (Chevalier et al., 2023) partitions long contexts into fixed-length chunks and iteratively generates summary vectors for each chunk, conditioned on the summaries of preceding chunks.
- **ICAE**[7] (Ge et al., 2024) compresses long contexts into a small set of memory tokens and

[6] https://huggingface.co/princeton-nlp/AutoCompressor-Llama-2-7b-6k

[7] https://huggingface.co/sggetao/icae

fine-tunes only the ad-hoc compressor using LoRA. The same backbone model is used for both the compressor and the target LLM.

- **xRAG**[8] (Cheng et al., 2024) directly integrates projected dense retriever embeddings as knowledge inputs, eliminating explicit ad-hoc compression. Only the projection module is trained during both pretraining and instruction tuning.
- **PCC-lite**[9] (Dai et al., 2025) follows a similar memory-based compression paradigm but fine-tunes only the compressor. It employs GPT-2 Large as a lightweight compressor and LLaMA-3-8B-Instruct as the target LLM.
- **PISCO**[10] (Louis et al., 2025) also represents long contexts using a limited number of memory tokens. However, unlike ICAE and PCC, it uses the same backbone model for both the compressor and the target LLM with different LoRA adapters and only finetunes both the compressor and the LLM for task adaptation.

B Training Details

In recent context compression and memory-augmented RAG methods, multi-stage training has become a standard practice. In particular, prior compressor-based approaches typically involve (i) pretraining a compression module and (ii) adapting the target LLM. From this perspective, the only additional component introduced by ConvX, relative to these baselines, is Stage 1 (converter pretraining), while Stages 2–3 align with the LLM adaptation procedures commonly adopted in existing approaches.

Importantly, the converter is intentionally designed to be lightweight (a small 3-layer module with attention over slots), so its cost is modest. In our implementation, its pretraining time is roughly half of the LLM pretraining stage. To make this concrete, the full training pipeline required approximately:

- Stage 1 - Converter pretraining: 28 GPU hours (1× A6000 Pro 96 GB)
- Stage 2 - LLM pretraining: 60 GPU hours (1× A6000 Pro 96 GB)
- Stage 3 - Self-distillation: 9 GPU hours (1× A6000 Pro 96 GB)

The specific hyperparameters for training the model are reported in Table 4, 5 and 6.

Stage 1 Hyperparameters	
Training steps	$10,000$
Warmup ratio	0.03
Learning rate	10^{-3}
Scheduler type	Linear
Batch size	128
Gradient accumulation steps	1
Maximum passage length	256

Table 4: Hyperparameters for pretraining the converter.

Stage 2 Hyperparameters	
Training steps	$10,000$
Warmup ratio	0.03
Learning rate	2×10^{-4}
Scheduler type	Linear
Batch size	4
Gradient accumulation steps	32
Maximum passage length	256
LoRA r	8
LoRA α	32
LoRA dropout	0.05

Table 5: Hyperparameters for pretraining the LLM.

Stage 3 Hyperparameters	
Training epoch	1
Warmup ratio	0.05
Learning rate	5×10^{-5}
Scheduler type	Linear
Batch size	4
Gradient accumulation steps	32
Maximum passage length	256
LoRA r	32
LoRA α	128
LoRA dropout	0.1

Table 6: Hyperparameters for the self-distillation.

C Lexical Information of Memory Slots

Table 7 summarizes the vocabularies recovered from the converted memory slots. Our design aims

[8] https://huggingface.co/Hannibal046/xrag-7b
[9] https://huggingface.co/collections/BroAlanTaps/pcc-finetuned
[10] https://huggingface.co/naver/pisco-mistral

(Original passage) Does He Love You "Does He Love You" is a song written by Sandy Knox and Billy Stritch, and recorded as a duet by American country music artists Reba McEntire and Linda Davis. It was released in August 1993 as the first single from Reba's album "Greatest Hits Volume Two". It is one of country music's several songs about a love triangle. Section:History. "Does He Love You" was written in 1982 by Billy Stritch and Sandy Knox. He recorded it with a trio in which he performed at the time, because he wanted a song that could be sung by the other two members of the trio, both of whom were women. It had been pitched to Barbara Mandrell and Liza Minnelli, but McEntire ended up recording it. She had wanted to include Linda Davis, then a vocalist in her road band, as a duet partner. McEntire's husband and manager, Narvel Blackstock, told her that MCA Records "would rather [she] record this with somebody more established", such as Wynonna Judd or Trisha Yearwood,
ConvX 128× memory slots:
M_1: Love , 9 , du , album , Section , single , two , 1 , written
M_2: es, Question, Q, y, yes, ie, els, on, ley, rell
ConvX 16× memory slots:
M_1: Love , Du, Mc , Do, Mine, Me, Bel, Sand, Kiss, My
M_2: L , W , first , R, B, co, H , F, C, G
M_3: love , recorded , written , relationship, career, hit, record , track, version, release
M_4: 2 , 3 , 4, the, 1, a, , in, and
M_5: es, y, is, on, ley, ney, er, et , ing, ver
M_6: 9 , 0, 8 , 7, 6, 5, B, C, L, R
M_7: Q, Question, of, to, the, and
M_8: 1 , the, 2 , in, and, a
M_9: two , single , one , three, would, time , number, year, work, several
M_{10}: song , album , singer, vocal , songs , chart, music , solo, recording , studio
M_{11}: Billboard, Country, Love, Heart, Know, Country, Billy , Bi, Want, Records
M_{12}: Section , country , later, husband , band , American , wife, October, title, United
M_{13}: du , rem, pe, like, together, collabor, ac, often, even, along
M_{14}: yes, rell , acks, ocal, oves, itt, icks, ais, ough, ights
M_{15}: Question, Q, the, in, a, of, to, and
M_{16}: hits, released , performed , including, includes, charts, tells, vocals, appears, married

Table 7: Examples of lexical cues that are contained in the memory slots for the passage used for illustrating attention heatmap in Section 4.4.

to encourage each memory slot to specialize in capturing a coherent subset of lexical signals. The results show that this behavior emerges only partially: while some memory slots encode informative lexical cues relevant to the original passage, others tend to capture high-frequency or less informative tokens, such as stop words or corpus-dominant vocabulary. For instance, in the $128\times$ ConvX setting, memory slot $\mathrm{M}2$ is largely dominated by high-frequency tokens, reflecting distributional biases inherited from the Stage 1 pretraining corpus. Although M_1 contains several informative tokens (e.g., "Love," "du," "album," and "written"), the semantic coherence among these vocabularies remains limited.

In contrast, ConvX $16\times$ provides a larger number of memory slots, enabling each slot to focus on more coherent lexical patterns while also increasing the number of slots that effectively filter out less informative signals. This observation suggests that the number of memory slots plays an important role in achieving robust RAG performance.