PISCO: Pretty Simple Compression for Retrieval-Augmented Generation

Maxime LOUIS
Naver Labs Europe
maxime.louis@naverlabs.com

Hervé Dejean
Naver Labs Europe

Stéphane Clinchant
Naver Labs Europe

Abstract

Retrieval-Augmented Generation (RAG) pipelines enhance Large Language Models (LLMs) by retrieving relevant documents, but they face scalability issues due to high inference costs and limited context size. Document compression is a practical solution, but current soft compression methods suffer from accuracy losses and require extensive pretraining. In this paper, we introduce PISCO[1], a novel method that achieves a 16x compression rate with minimal accuracy loss (0-3%) across diverse RAG-based question-answering (QA) tasks. Unlike existing approaches, PISCO requires no pretraining or annotated data, relying solely on sequence-level knowledge distillation from document-based questions. With the ability to fine-tune a 7-10B LLM in 48 hours on a single A100 GPU, PISCO offers a highly efficient and scalable solution. We present comprehensive experiments showing that PISCO outperforms existing compression models by 8% in accuracy.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Guu et al., 2020; Borgeaud et al., 2022) pipelines have become a crucial component in addressing various natural language tasks. By incorporating documents retrieved from a selected collection, RAG enhances Large Language Models (LLMs) enabling them to provide more accurate, current, and domain-specific responses.

The primary drawback is the increased inference cost, which scales quadratically with the number of tokens and, consequently, with the number of retrieved documents when using transformer-based architectures. In addition to inference costs, the limitations on LLM context size restrict the number of documents—and thus the amount of information—that can be utilized. This constrains the potential scaling of inference time (Yue et al., 2024).

[1]Code and models will be released soon.

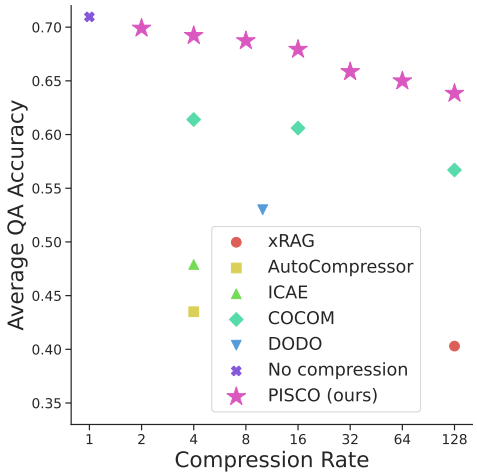

Figure 1: PISCO substantially outperforms existing context compression methods for question answering with RAG. Shown here with Mistral-7B backbone.

Compressing documents is a practical way to reduce the computational burden of processing large contexts. Hard compression techniques focus on altering the surface structure of the documents, such as by pruning (Pan et al., 2024; Li et al., 2023; Wang et al., 2023) or summarization (Xu et al., 2023). These methods are easily interpretable and can typically be applied to any LLM without requiring modifications. However, the compression rate is limited by the amount of information that can be effectively conveyed through text tokens, usually achieving a reduction of 2x-5x.

Soft compression techniques aim to condense documents into vector representations (Wingate et al., 2022). They may also involve attention key-value pairs that the model attends to during generation, either through self-attention (Rau et al., 2024a; Cheng et al., 2024) or dedicated cross-attention mechanisms (Yen et al., 2024). These methods trade off interpretability for efficiency, achieving higher compression rates while maintaining some performance levels. Most existing soft compression approaches for RAG follow a similar pipeline

Findings of the Association for Computational Linguistics: ACL 2025, pages 15506–15521

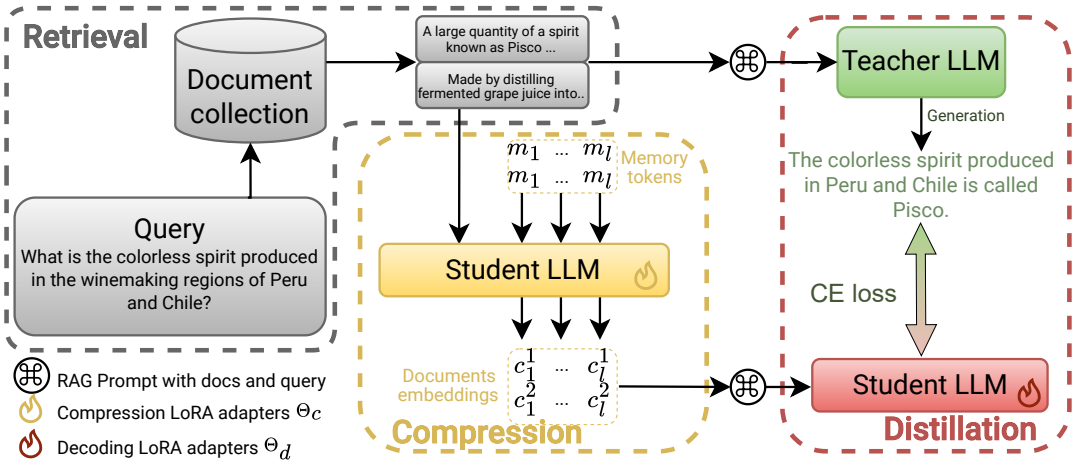

Figure 2: Overview of PISCO training, shown here with $k = 2$ documents. Training is supervised by distillation from a teacher model. Once trained, the full collection of documents can be compressed once to allow fast inference.

(Cheng et al., 2024; Chevalier et al., 2023; Ge et al., 2023; Rau et al., 2024b; Qin et al., 2024; Li et al., 2024; Yen et al., 2024). Typically, a pretraining task (such as auto-encoding and/or causal language modeling) on unlabeled data is used to train an initial compressor, followed by fine-tuning for RAG question answering (QA) to optimize the embeddings for QA tasks.

Currently, all existing soft methods experience significant accuracy losses ($> 8\%$, see Table 2) on RAG-QA benchmarks when compared to the original, uncompressed generator. This hinders the deployment of such systems, as accuracy is a primary concern over inference costs for most RAG systems. Additionally, all existing methods require pretraining on a large dataset as well as annotated QA datasets for fine-tuning.

This paper presents **PISCO, a compression method for RAG that achieves a x16 document compression rate with minimal to no loss in accuracy** (0-3%) across a wide range of RAG-QA tasks, covering multiple domains. Unlike prior approaches, **PISCO requires neither pretraining nor annotated data**: it relies solely on distillation from open-ended, document-based questions. With x16 compression, PISCO models achieve a 5.7x inference speed-up. This makes PISCO a highly efficient and scalable solution for practical applications. Notably, **fine-tuning a 7-10B LLM into a PISCO model can be completed in under 48H on a single A100 GPU**.

Here is a summary of our main contributions:

- A simplified and more efficient pipeline for training compression models for RAG,
- Strong experiments results on in-domain, out-of-domain and multilingual QA: PISCO outperforms current state-of-the-art compression models by 8% in accuracy,
- An in-depth analysis demonstrating that pretraining has little benefits for compression models, investigating the quality of labels and illustrating the structure of the compressed documents embeddings.

In Section 2, we review related work on compression techniques for RAG. Section 3 describes the proposed method for PISCO. We then report experimental results on standard RAG benchmark 4.2. Furthermore, the importance of the design choices of PISCO is analyzed in section 4.3. Finally, we evaluate PISCO robustness and generalization to new tasks and domains in section 4.4.

2 Related Work

Table 1 compares key methods for soft compression, which, although not always explicitly designed for this purpose, can be applied in RAG (retrieval-augmented generation) applications. (Verma, 2024) presents a more thorough survey or introduction to context compression.

2.1 Dealing with long contexts via compression

In (Chevalier et al., 2023), the authors present the Autocompressor, a recursive context compression method trained on a language modeling task. By appending compression tokens to the context and extracting the hidden states, this approach supports longer contexts and can be applied to document

Paper	Compression rate†	Pre-training Required		Distillation fine-tuning	Decoder training	Supervised fine-tuning
		AE	LM			
AutoCompressor (Chevalier et al., 2023)	6-40x	✗	✓	✗	✓	✗
ICAE (Ge et al., 2023)	2-8x	✓	✓	✗	✗	✓
xRAG (Cheng et al., 2024)	100-180x	✓	✓	✓‡	✗	✓
DODO (Qin et al., 2024)	5-10x	✗	✓	✗	✓	✗
x500 (Li et al., 2024)	6-480x	✓	✗	✗	✗	✓
CEPE (Yen et al., 2024)	∼ 256x	✗	✓	✓‡	✓	✓
COCOM (Rau et al., 2024b)	4-128x	✓	✓	✗	✓	✓
PISCO (ours)	2-128x	✗	✗	✓	✓	✓

Table 1: Summary of papers with different soft compression strategies. AE=Auto-encoding, LM=next token prediction. † *Compression rates are taken from QA/RAG experiments when available.* ‡ *xRAG/CEPE use token-level distillation which requires ground truth labels.*

compression in RAG-QA. The in-context auto-encoder (ICAE) (Ge et al., 2023) simplifies this by freezing the decoder, removing recursion, and pre-training through document auto-encoding. In (Yen et al., 2024), multiple contexts are encoded in parallel, with cross-attention layers introduced between the query and documents in the decoder LLM. This separation of query-document and self-attention reduces complexity and accelerates inference. However, large compressed documents remain a limitation for efficient storage. Finally, DODO (Qin et al., 2024) compresses earlier context sections into adaptive *nugget* states, using cross-attention for nuggets and self-attention for recent context. Though not specifically optimized for document QA, it can be used in that perspective.

2.2 Compression specific to RAG-QA

In (Rau et al., 2024a), the authors specifically address the RAG-QA problem. After large-scale auto-encoding and language modeling pretraining, they fine-tune their decoder models to handle multiple documents simultaneously. Although this approach enhances the usability and performance of the RAG pipeline, there remains a significant performance drop (~8%) between uncompressed and x16 compressed models. The x500 Compressor (Li et al., 2024) is similar to COCOM except that the document embeddings consist directly of the K/V values obtained on the memory tokens during forward pass. This saves decoding computations but substantially increases the storage size of the embeddings. The xRAG method (Cheng et al., 2024) proposes leveraging existing document embeddings—such as those used for retrieval—to reduce the storage and computational costs of generating additional embeddings. To achieve this, they train a small adapter to map the retrieval embeddings into the input space of a frozen decoder LLM. Similar to (Yen et al., 2024), xRAG also utilizes token-level distillation for fine-tuning in QA tasks.

All current compression methods (Cheng et al., 2024; Chevalier et al., 2023; Ge et al., 2023; Rau et al., 2024b; Qin et al., 2024; Li et al., 2024; Yen et al., 2024) rely on large-scale pretraining tasks and require annotated labels. Despite their advancements, these methods still fall short of achieving the QA performance of their uncompressed LLM backbones (see 2).

3 Methods

3.1 Retrieval-Augmented Generation

In RAG, each query q is augmented with a set of relevant documents $(d_1, d_2, \ldots, d_k)$ retrieved from a large database of documents $\mathcal{D}$. For improved performance, this process typically involves two steps: first, a retriever identifies an initial pool of relevant documents, and then a re-ranker refines and prioritizes the k most relevant ones. The final response r is generated by prompting an language model $\mathcal{F}$ with both the query and the set of retrieved documents.

In general, the accuracy of the generated response improves as the number of documents increases. However, since documents tend to be longer than queries and the computational complexity of transformer-based models scales quadratically with the context length, this can make generation computationally expensive and cause delays. A soft compression model addresses this by mapping each document d_i into a shorter set of embeddings or a key-value (K/V) cache, $\mathbf{c_i}$. The generation process is then conditioned on these compressed representations: $r \sim \mathcal{F}(\cdot \mid q, \mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_k})$.

3.2 PISCO

PISCO adopts a standard architecture, involving a compressor and decoder model, detailed in the following section. The main difference lies in its training task. The method is described on Figure 2.

Compression is performed following the approach in (Chevalier et al., 2023; Ge et al., 2023; Rau et al., 2024b), utilizing the language model $\mathcal{F}$ with LoRA (Hu et al., 2021) adapters θ_c. Specifically, a set of l memory tokens $(m_1, \dots, m_l)$ is appended to each document d_i: the corresponding prompt $(d_i; m_1, \dots, m_l)$ is passed through $\mathcal{F}_{\theta_c}$. The l final hidden states, corresponding to the memory tokens, are extracted to form the document embeddings $\mathbf{c_i} = (c_i^s)_{s=1\dots l}$. Each document is encoded into l vectors, each sharing the dimension of the encoder's embedding space, $\mathcal{F}$. The number of tokens l effectively controls the compression rate. The memory tokens $(m_1, \dots, m_l)$ are optimized jointly with the LoRA adapters θ_c. During optimization, $\mathcal{F}_{\theta_c}$ is encouraged via the distillation objective described below to encode information about the compressed document.

Decoding is carried out using the language model $\mathcal{F}$ with a separate set θ_d of LoRA adapters. Previous works (Cheng et al., 2024; Ge et al., 2023; Li et al., 2024) attempt to freeze the decoder, to allow for plug-and-play use of the compressor. Early experiments suggested such an approach is unlikely to reach satisfying results (see Appendix G). Fine-tuning the decoder might be crucial as it allows to adapt its interactions with the compressed representations **depending on the query**. Additionally, this fine-tuning process does not compromise the ease of setting up PISCO, as our end-to-end fine-tuning completes in only a day for $7-10$B-parameter models on a single high-end GPU.

Distillation objective While the architecture of PISCO is similar to existing approaches, its training principle is fundamentally different. The motivation for using a distillation approach stems from an invariance principle: language models should give the same answers whether their input is compressed or not. To achieve this, we propose to use Sequence-level Knowledge Distillation (SKD) (Kim and Rush, 2016): generating labels with a teacher model rather than token-level distillation based on existing labels as done in previous works.

Specifically, given a query q, let $a_1, \dots, a_r$ represent the tokens generated by the teacher based on the documents and query and $\mathbf{a_{<i}} = (a_1, \dots, a_{i-1})$:

$$a_i \sim \mathcal{T}(\cdot \mid d_1, \dots, d_k, q, \mathbf{a_{<i}}).$$

The training objective on the parameters θ_c and θ_d is the cross-entropy loss computed on the decoder conditioned on the compressed documents and the query:

$$\mathbf{c_i} = (c_i^s)_{s=1,\dots,l} = \mathcal{F}_{\theta_c}(d_i, m_1, \dots, m_l)$$

$$\mathcal{L}(\theta_c, \theta_d) = -\sum_{i=1}^{r} \log \mathcal{F}_{\theta_d}(a_i \mid q, \mathbf{c_1}, \dots, \mathbf{c_k}, \mathbf{a_{<i}})$$

Further details on this process are provided in Appendix A. Note that the teacher-generated labels can be precomputed and re-used across different training runs.

Note that in xRAG (Cheng et al., 2024), the authors minimize the Kullback-Leibler (KL) divergence between the logits of the teacher and student models, with both models being teacher-forced on a reference answer. Similarly, CEPED (Yen et al., 2024) minimizes a weighted combination of the KL divergence between the teacher logits and student models and the standard cross-entropy loss on reference labels. Both methods implement in fact token-level distillation and rely on labeled data. However, token-level knowledge distillation is often less efficient than sequence-level knowledge distillation (Kim and Rush, 2016) and less convenient as it requires labeled data, which is not the case for SKD.

4 Experiments

Our experiments aim to measure the performance of PISCO models §4.2, then to analyse the importance of training data, distillation and pretraining §4.3. Furthermore, we evaluate PISCO models generalization §4.4 to out-of-domain, multilingual data and large number of documents. Finally, we investigate how information is stored within the document embeddings §4.5.

4.1 Experimental details

We run experiments using Mistral-7B-instruct (Jiang et al., 2023)[2], LLama-3.1-8B-instruct [3] and SOLAR-10.7B-Instruct [4] as different backbones

[2] huggingface/mistralai/Mistral-7B-Instruct-v0.2
[3] huggingface/meta-llama/Llama-3.1-8B-Instruct
[4] huggingface/upstage/SOLAR-10.7B-Instruct-v1.0

	Compression rate	ASQA	HotpotQA	NQ	TriviaQA	POPQA	Average
Decoders with no compression							
Mistral-7B	-	0.74	0.51	0.69	0.92	0.70	0.71
Llama-3.1-8B	-	0.71	0.50	0.65	0.90	0.68	0.69
Solar-10.7B	-	0.75	0.55	0.71	0.93	0.71	0.73
Compression Models							
xRAG-mistral-7B† (Cheng et al., 2024)	x128	0.34	0.27	0.32	0.77	0.33	0.40
AutoCompressor†‡ (Chevalier et al., 2023)	x4	0.57	0.31	0.35	0.70	0.24	0.43
ICAE‡ (Ge et al., 2023)	x4	0.47	0.29	0.42	0.78	0.43	0.48
DODO‡ (Qin et al., 2024)	x10	0.52	0.38	0.48	0.82	0.47	0.53
COCOM (Rau et al., 2024b)	x16	0.63	0.46	0.58	0.89	0.48	0.61
PISCO - Mistral$^{\Delta}$ (Ours)	x16	0.72	0.48	0.65	0.90	0.66	0.68
PISCO - Mistral (x128) (Ours)	x128	0.68	0.46	0.61	0.89	0.55	0.64
PISCO - Llama (Ours)	x16	0.72	0.50	0.64	0.91	0.66	0.69
PISCO - Solar (Ours)	x16	**0.78**	**0.57**	**0.70**	**0.94**	**0.71**	**0.74**

Table 2: **Performance (accuracy) on general domain QA with 5 retrieved documents.** PISCO x16 models, built using Mistral, Llama, and Solar decoders, **outperform all other compression models across all datasets and closely match the performance of their uncompressed counterparts**. † methods limited to one context. ‡ methods using Llama-2-8b. $^{\Delta}$ achieves a x5.7 inference speed up compared to Mistral-7B.

for PISCO. Our training set of questions [5] is taken from (Rau et al., 2024b): it consists of 453k questions based on documents from Wikipedia-KILT (Petroni et al., 2020) which we preprocess in chunks of 128 tokens and use as our database collection [6]. For each question, we search the first top-k documents and feed them to a teacher LLM to obtain the silver label used for distillation. During training, the number of retrieved documents k is set to 5. Each document is compressed into l embedding vectors where l is fixed for each PISCO model. PISCO models with compression rate 16 use 8 memory embeddings per document.

All experiment details, including the choice of the retriever, reranker and prompts are provided in Appendix C and Appendix B. Trainings and evaluations were performed using the Bergen (Rau et al., 2024a) library.

4.2 Main results

After training, we first evaluate the PISCO models on general knowledge QA tasks: Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), HotpotQA (Yang et al., 2018), ASQA (Stelmakh et al., 2022), and PopQA (Mallen et al., 2022) datasets. Our main evaluation metric is the accuracy –also called match in the QA context– which we define as 1 if the normalized label is found within the normalized prediction and 0 otherwise, as detailed in Appendix J.

[5] huggingface/datasets/dmrau/multi_qa
[6] huggingface/datasets/dmrau/kilt-128

Results are shown on Table 2. PISCO largely outperforms the other compression methods with +8% accuracy on average compared to COCOM with Mistral backbone. PISCO - Mistral trained with 128 compression rate outperforms xRAG by more than 20%. **In fact, all PISCO models with compression rate 16 are very close (0-3%) to their uncompressed backbones.** Most notably, PISCO-Solar outperforms Solar, showing that the compression can have a de-noising effect, discarding irrelevant information. Regarding efficiency, PISCO-Mistral requires only 17% of the floating-point operations at inference required by Mistral model, which shows compression gains directly translate into latency gains.

By changing the number of memory tokens l prompted to the compressor $\mathcal{F}_{\theta_c}$, we can train models with different compression rates. Figure 1 shows the average accuracy for PISCO-Mistral models with compression rates between x2 and x128. Performance decreases gradually as the compression rate increases up to 16, after which the decline becomes more pronounced.

Then, we use gpt-4o (Hurst et al., 2024) and its strong ability to assess semantic content to perform pairwise comparisons (Dubois et al., 2024) between generated answers to compare models. Results are shown on Figure 3. It shows that PISCO-Mistral outperforms COCOM model and is on par with its uncompressed backbone. The exact setup for this evaluation is detailed in appendix E.

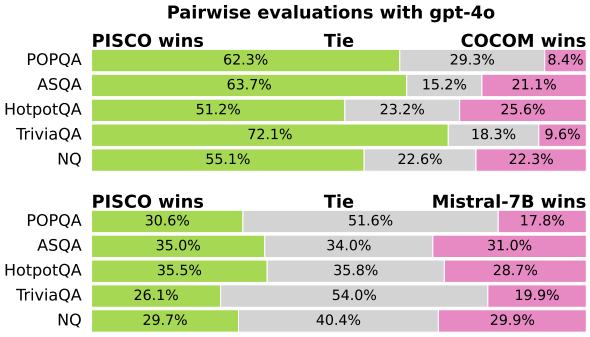

Figure 3: Pairwise comparison with GPT-4o shows that PISCO, utilizing the Mistral-7B backbone, outperforms COCOM across all datasets. It performs comparably to Mistral-7B while achieving a 16x compression rate.

4.3 Analysis of Training Data and Tasks for Compression

In this section, we aim to give evidences justifying the design choices of the PISCO approach.

Pretraining has little benefits. A potential way to improve compression models was by improving pretraining with new or refined pretraining tasks. To explore this, we conducted experiments using pretraining on 10B tokens extracted from FineWeb[7] with a variety of tasks. These tasks included auto-encoding (Ge et al., 2023; Rau et al., 2024b), text continuation from compressed representations (Rau et al., 2024b; Cheng et al., 2024), and a novel task of text continuation from a sequence of keywords within a compressed segment—enabling access to information embedded in the learned representations, aimed at mitigating a potential "lost-in-the-middle" effect. Additionally, we tested continuation from multiple documents, where the model was prompted to continue text either from within or following a designated document among several compressed ones (see Appendix I). Figure 4 illustrates that, across all preliminary experiments, there is **a weak correlation between success in pretraining tasks and performance in QA**. Notably, training on auto-encoding often achieves near-perfect Rouge-L scores (>0.99) without any significant improvement in QA performance.

To analyze in detail the impact of the adopted fine-tuning strategy, we ran experiments with variable number of fine-tuning samples. We compare performances when fine-tuning is applied to a pretrained model or from scratch. Results are shown on Figure 5. Pretraining benefits final performances

[7] huggingface./datasets/HuggingFaceFW/fineweb

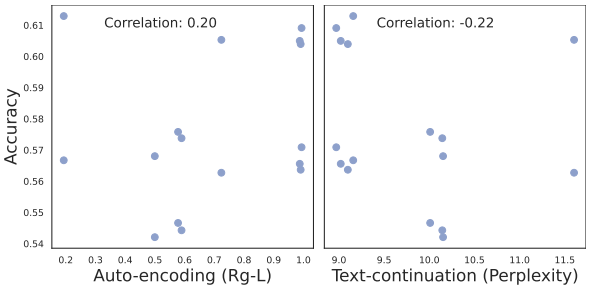

Figure 4: Performances on pretraining tasks versus performance on RAG-QA. Correlations are very small, indicating that pretraining has only little benefits on the downstream QA task.

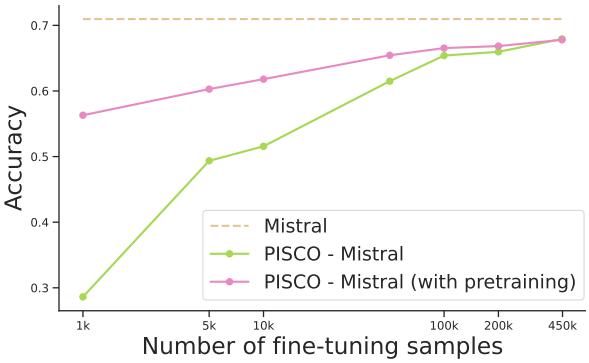

Figure 5: Impact of the number of fine-tuning samples on performance, with and without pretraining. Pretraining only improves QA performance for low fine-tuning sample size.

for low fine-tuning sample size, but it is not useful at 450k samples.

Needle-in-a-Haystack analysis. Secondly, we analyzed the different model behaviors on a needle-in-a-haystack test (gkamradt, 2024), an accessible proxy for RAG-QA that effectively measures the model's retrieval and localization capabilities, crucial for accurate question-answering on large datasets. Interestingly, while pretraining enables some success on the needle-in-a-haystack task, fine-tuning on the raw labels used in (Rau et al., 2024b; Cheng et al., 2024) diminishes this capability, as illustrated in Figure 6. This result highlighted the need for a better fine-tuning approach. Higher-quality labels from a teacher LLM emerged as a promising solution: providing more informative signals during fine-tuning (Ho et al., 2022; Ren et al., 2024). Early experiments shown on Figure 6 and main results shown on Table 2 indeed confirm the benefits.

Impact of Teacher and Labels Quality. To understand the impact of the teacher LLM we train

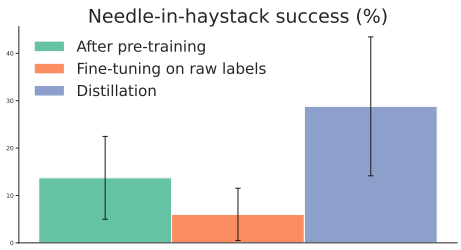

Figure 6: Needle-in-a-haystack results from preliminary experiments: fine-tuning on raw labels hinders performance, while sentence-level distillation enhances it.

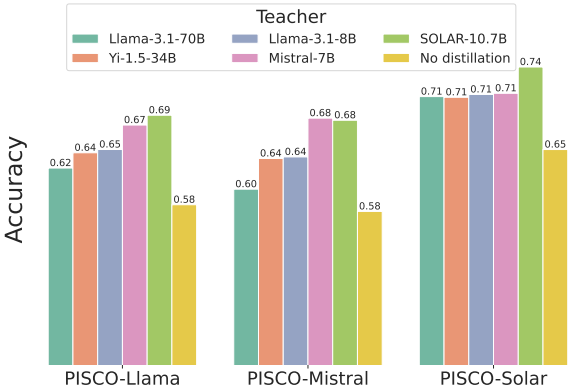

Figure 7: Impact of the choice of teacher on PISCO.

PISCO models with varying teachers. Average accuracy in general domain QA for each obtained model are shown on Figure 7. First, not resorting to distillation and relying on the usual labels leads very bad results, showing the importance of distillation for PISCO models. Then, interestingly, the best-performing teachers are generally Mistral-7B or Solar-10B models, not necessarily the stronger teachers, as found in (Xu et al., 2024). A manual analysis of the labels for each teacher suggests that these models often include justifications for their answers based on the given contexts: training the PISCO models to replicate this reasoning process may account for the observed performance improvements, as shown in (Ho et al., 2022; Ren et al., 2024). Note that PISCO-Solar models are very robust across all teachers.

To summarize our analysis, we have shown that there is little transfer between pretraining tasks and downstream RAG-QA performance, thus justifying why pretraining is not key for compression models. Secondly, we have seen that standard fine-tuning on raw labels is problematic since it leads to poor results in needle-in-a-haystack test. Then, we showed that SKD with a teacher LLM solves this problem. Finally, we tested various teacher LLMs, recovering some of the findings of (Ho et al., 2022; Xu et al., 2024; Ren et al., 2024). Overall, these analysis explain the rationale which led to each design decisions of the training pipeline for PISCO.

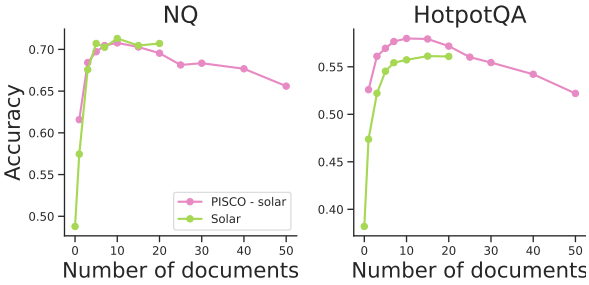

Figure 8: Accuracy of PISCO-Solar compared to Solar with increasing number of documents.

4.4 Generalization Evaluation

Increasing the number of documents. To evaluate the PISCO models' ability to handle a large volume of documents, we conduct inference on PISCO - Solar with document sets ranging from 1 to 50 on NQ and HotpotQA. Results are presented in Figure 8. For comparison, we include the base SOLAR model, which can process only up to about 20 documents due to its 4096-token context length limit. PISCO's performance aligns closely with the base model, with a gradual decrease beyond 20 documents, a common trend in RAG models (Jin et al., 2024), which could be addressed by increasing k during fine-tuning (all of our experiments train with $k = 5$ documents).

Out-of-domain. We evaluate whether PISCO models generalize to unseen domains using a diverse set of datasets (details in Table 7 in the appendix). The results are shown in Table 3. Results in Table 3 show that PISCO models generalize nearly as effectively as their backbone decoders, with some performance drops (-3–10%) on the ParaphraseRC task. This indicates robust compression capabilities, showing PISCO models do not rely on memorizing the general knowledge in the KILT collection.

Multilinguality. We evaluate whether PISCO models generalize to unseen languages using the MKQA dataset (Longpre et al., 2021). The experiments use the bge-m3 retriever and the recall-3gram metric, more resilient to language variation (Chirkova et al., 2024). We choose a latin language (french) as well as Korean and Russian. Note that the PISCO backbones Llama and Mistral are not strong multilingual models, but these experiments

				RobustQA					Multilingual QA		
Dataset	Bio-QA	Covid	ParaphraseRC	Lifestyle	Writing	Science	Recreation	Tech	FR	KO	RU
Metric	Recall	F1	Accuracy	F1	F1	F1	F1	F1	recall-3gram		
Llama	0.26	**0.17**	0.48	0.25	0.23	0.25	0.25	0.23	0.56	0.28	0.47
PISCO - Llama	0.24	0.12	0.38	0.28	0.27	**0.26**	**0.26**	**0.26**	0.54	0.24	0.44
Mistral	0.27	0.13	0.49	0.28	0.27	**0.26**	0.25	0.25	0.57	0.26	0.40
PISCO - Mistral	0.26	0.11	0.38	0.28	0.27	**0.26**	0.25	**0.26**	0.52	0.17	0.35
Solar	0.28	0.14	**0.50**	0.28	0.27	**0.26**	**0.26**	0.25	0.59	0.20	0.52
PISCO - Solar	**0.29**	0.10	0.47	**0.29**	**0.28**	0.25	0.25	**0.26**	0.60	0.16	0.48

Table 3: Out-of-domain and multilingual QA performance. PISCO models generalize well to new domains and languages, achieving performance comparable to their uncompressed decoders. See Table 7 for datasets details.

serve mostly to analyze the compression behavior. Results are shown on Table 3 (right). PISCO models seem to generalize fairly well to other languages, with still a small drop compared to their backbones. Further analysis is needed to determine whether the drop is due to compression or language generation limitations.

Summarization Here, we evaluate PISCO-Mistral on CNN/daily-mail (See et al., 2017): a summarization dataset. Results are shown on Table 4 for Mistral and PISCO-Mistral. The PISCO model has the same performance as the Mistral model. This indicates that PISCO embeddings are not only suited for QA but can be extended to other tasks.

Model	Rouge-1	Rouge-2	Rouge-L
Mistral	0.21	0.04	0.19
Pisco-Mistral	0.20	0.04	0.18

Table 4: ROUGE scores for Mistral and Pisco-Mistral on CNN/daily-mail summarization task.

To sum-up, we showed that **PISCO models are robust to the number of documents** used, **generalize well to unseen domains** beyond the Wikipedia collections, and that **compression works relatively well for unseen languages**. Overall, it shows that PISCO models are strong language models for RAG.

4.5 Document embeddings analysis

To better understand how compression works, we compute, on a set of documents, the cosine similarity between the l embeddings and each document token. Interestingly, Figure 9 shows there is a spatial specialization of the embeddings, each attending preferably to some part of the text. This

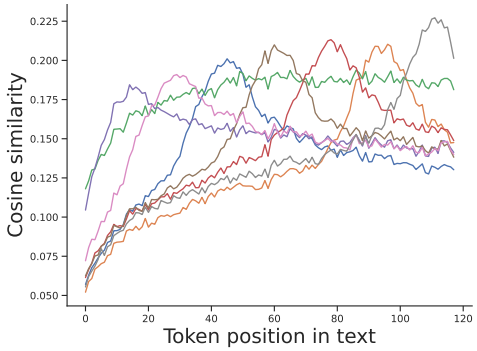

Figure 9: Average cosine similarities between each of the $l = 8$ document embeddings and individual document tokens reveals spatial specialization in the memory embeddings, with each focusing on neighboring positions in the encoded document.

specialization does not occur when the decoder is kept frozen. Then, we used the logits lens (nostalgebraist, 2020) to determine, for each document embedding, the top tokens when applying the LLM head. Figure 10 shows, for some text, how often text tokens correspond to one of these attributed tokens. Additional results are shown on Table 9.

Second, we gather all embeddings of all documents as well as token embeddings and visual-

Pisco is a colorless or yellowish-to-amber-colored spirit produced in winemaking regions of Peru and Chile. Made by distilling fermented grape juice into a high-proof spirit, it was developed by 16th-century Spanish settlers as an alternative to orujo, a pomace brandy that was being imported from Spain. It had the advantages of being produced from abundant domestically grown fruit and reducing the volume of alcoholic beverages transported to remote locations.

Figure 10: Each colored word in this text appears in the top-10 token attributed with logit lens in the document embeddings. Most words appear.

ize using t-SNE, in the input space of the decoder $\mathcal{F}_{\theta_d}$. Result is shown in the appendix on Figure 12. The document embeddings tend to lie outside of the document tokens embeddings. This underscores the need of fine-tuning the decoder to exploit newly formed compressed representations: an effective pipeline leverages new areas of the embedding space, with different semantic content.

5 Conclusion

We proposed PISCO, the first compression method for RAG which enables large compression rates with little to no accuracy loss. Our analysis and ablations revealed the ineffective transfer between pretraining and the question answering task for compression models. We also showed the importance of training labels for compression: using an appropriate teacher LLM for distillation is key. Given the strong evidences of robustness and accuracy, PISCO models may be used as drop-in replacement for existing RAG systems currently relying on their uncompressed backbones. Adopting PISCO would reduce inference costs and latency with minimal performance impact.

6 Limitations

While achieving state-of-the-art performances for context compression for question-answering with retrieval augmented generation, our work has some limitations.

First, we only tested the compression method in a QA setting: it is clear that the compressed embeddings carry a lot of semantic value and there are reasons to believe it should enable to deal with more complex tasks such as summarization, long chats, long contexts via compression etc and this would form a natural continuation of our work.

Second, at variance with (Rau et al., 2024b), we do not propose in this contribution a light version of the compressor to be used in an online setting (i.e. compressing documents on the fly with a compression method sufficiently fast to accelerate the full RAG pipeline overall). Therefore our approach is only valuable when the collection of documents can be compressed once beforehand and one expects a sufficiently large volume of queries to compensate for the cost of the compression (NB: our tests show that compressing 1 million documents takes approximately 3h on a high-end GPU).

Third, our experiments suggest some multilingual generalization of the model abilities but to enable strong performances, we should augment our training set with multilingual data.

References

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

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. *arXiv preprint arXiv:2405.13792*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

Nadezhda Chirkova, David Rau, Hervé Déjean, Thibault Formal, Stéphane Clinchant, and Vassilina Nikoulina. 2024. Retrieval-augmented generation in multilingual settings. *arXiv preprint arXiv:2407.01463*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

gkamradt. 2024. Needle in a haystack - pressure testing llms.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Rujun Han, Peng Qi, Yuhao Zhang, Lan Liu, Juliette Burger, William Yang Wang, Zhiheng Huang, Bing Xiang, and Dan Roth. 2023. Robustqa: Benchmarking the robustness of domain adaptation for open-domain question answering. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4294–4311.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Bowen Jin, Jinsung Yoon, Jiawei Han, and Sercan O Arik. 2024. Long-context llms meet rag: Overcoming challenges for long inputs in rag. *arXiv preprint arXiv:2410.05983*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. 2023. Bioasq-qa: A manually curated corpus for biomedical question answering. *Scientific Data*, 10(1):170.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. 2024. Splade-v3: New baselines for splade. *arXiv 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, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*.

Zongqian Li, Yixuan Su, and Nigel Collier. 2024. 500xcompressor: Generalized prompt compression for large language models. *arXiv preprint arXiv:2408.03094*.

Shayne Longpre, Yi Lu, and Joachim Daiber. 2021. Mkqa: A linguistically diverse benchmark for multilingual open domain question answering. *Transactions of the Association for Computational Linguistics*, 9:1389–1406.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. 2020. Covid-qa: A question answering dataset for covid-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*.

nostalgebraist. 2020. Interpreting GPT: The logit lens. LessWrong.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *arXiv preprint arXiv:2403.12968*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2020. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*.

G Qin, C Rosset, E Chau, N Rao, and B Van_Durme. 2024. Dodo: Dynamic contextual compression for decoder-only lms. Proceedings of the 62nd Annual Meeting of the Association for Computational ....

David Rau, Hervé Déjean, Nadezhda Chirkova, Thibault Formal, Shuai Wang, Vassilina Nikoulina, and Stéphane Clinchant. 2024a. Bergen: A benchmarking library for retrieval-augmented generation. *arXiv preprint arXiv:2407.01102*.

David Rau, Shuai Wang, Hervé Déjean, and Stéphane Clinchant. 2024b. Context embeddings for efficient answer generation in rag. *arXiv preprint arXiv:2407.09252*.

Xuan Ren, Biao Wu, and Lingqiao Liu. 2024. I learn better if you speak my language: Understanding the superior performance of fine-tuning large language models with llm-generated responses. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10225–10245.

Amrita Saha, Rahul Aralikatte, Mitesh M Khapra, and Karthik Sankaranarayanan. 2018. Duorc: Towards complex language understanding with paraphrased reading comprehension. *arXiv preprint arXiv:1804.07927*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. Asqa: Factoid questions meet long-form answers. *arXiv preprint arXiv:2204.06092*.

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

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. 2023. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*.

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. 2022. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. *arXiv preprint arXiv:2210.03162*.

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

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, and Radha Poovendran. 2024. Stronger models are not stronger teachers for instruction tuning. *arXiv preprint arXiv:2411.07133*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. *arXiv preprint arXiv:2402.16617*.

Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. 2024. Inference scaling for long-context retrieval augmented generation. *arXiv preprint arXiv:2410.04343*.

Hyperparameter	Value
Batch Size	128
LR	1×10^{-4}
LR scheduler	linear
optimizer	AdamW
Epochs	1
Max Tokens Teacher Generation	128
LoRA Layers (r)	all-linear
LoRA Rank (r)	16
LoRA Dropout	0.1
LoRA Alpha	32
LoRA Rank (r)	16
Weight Decay	0.1
Warmup Ratio	0.05
Max Gradient Norm	1.0
Documents max tokens	128

Table 5: Fine-tuning Hyper-parameters.

A Implementation details

Both for teacher-label generation and student evaluation, generation is done using greedy decoding, limited to a maximum of 128 tokens. Both for training and evaluation, documents are retrieved using Splade-v3 (Lassance et al., 2024) and Debertav3 (He et al., 2021). They are prompted from most relevant to less relevant according to the reranking scores.

B Training Hyper-parameters

Table 5 gives the hyper-parameters we used for training. Note that on top of the LoRA adapters, the embeddings of the memory tokens given as input to the encoder (as in (Rau et al., 2024b; Chevalier et al., 2023)) are also optimized, adding an extra $l\times$ model_hidden_size trainable parameters to each PISCO model (much less than the number of parameters in the adapters).

C Main prompt

Below is the prompt used in our experiments. <DOC> is replaced by the corresponding document compressed embeddings before the generation while <QUESTION> is replaced with the query q. It is formatted as an instruction prompt to the instruction-tuned models.

	NQ	TriviaQA	HotpotQA
Prompt 1	0.697	0.937	0.569
Prompt 2	0.694	0.942	0.571
Prompt 3	0.684	0.940	0.567
Prompt 4	0.694	0.937	0.572

Table 6: Effect of 4 different prompts on match for three datasets.

Main prompt

system: "You are a helpful assistant. Your task is to extract relevant information from provided documents and to answer to questions as briefly as possible."
user: "Background:
<DOC><SEP><DOC>...<SEP><DOC>
Question: <QUESTION>"

D Effect of prompts on PISCO models

We evaluate the robustness of PISCO models to prompt variations by testing with modified versions of the prompt shown in C. The results in Table 6 show minimal performance differences, indicating that the models are stable with different prompts and do not overfit to the specific prompt used during training. Notably, Prompt 3 provides no guidance to the model beyond the information in the documents.

Other prompts

- **system**: "Refer to the background document and answer the questions:"
  user: "Background:
  <DOC><SEP>...<DOC>
  Question: <QUESTION>"
- **system**: ""
  user: "Background:
  <DOC><SEP>...<DOC>
  Question: <QUESTION>"
- **system**: ""
  user: "<DOC><SEP>...<DOC> Question: <QUESTION>"

E Pairwise comparison using gpt-4o

To compare answers generated by different methods in a more precise way that using the accuracy

metric, we use gpt-4o with the following prompt, inspired from Alpaca-eval (Dubois et al., 2024). Evaluations were run using gpt-4o-2024-11-20. To limit costs, only a 1000 samples were used for each dataset. Answer positions in the prompt were randomly switched to prevent position bias.

Gpt pairwise comparison prompt

system: "You are a helpful assistant, that ranks models by the quality of their answers. Please act as an impartial judge. Do not allow the length of the responses to influence your evaluation. Be as objective as possible."
user: "Here is a question, a ground truth answer, an AI-generated answer 1 and an AI-generated answer 2. Which answer is the most correct one ? Simply answer 1 if the first is better, 2 if the second is better and 3 if it's a tie.
Question: <QUESTION>.
Ground truth answer: <REF_ANSWER>.
Answer 1: <ANSWER1>.
Answer 2: <ANSWER2>."

F Out-of-domain datasets

Table 7 provides details on the out-of-domain datasets used and the primary evaluation metric for each. We use the F1 score for RobustQA test suites, given the extended format of the reference answers.

G PISCO with frozen decoder

Freezing the decoder for compression models is appealing: it would enable to use compressed representations without any major change to the decoding pipeline of an existing system, as in (Cheng et al., 2024). To that end, we ran fine-tuning (with and without pre-training) of PISCO models with frozen decoder. Table 8 shows the difference in performance is huge. In fact, a look at the loss curves 11 seems to show that fitting only the compressor does not offer nearly enough flexibility for learning.

H Embeddings analysis

To better understand how information is compressed within the document embeddings, we apply the logit lens (nostalgebraist, 2020) to each embedding. This allows us to identify the top 10 tokens

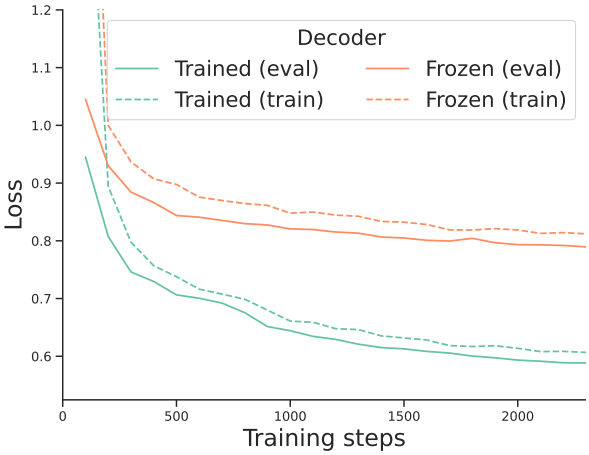

Figure 11: Train and eval loss curves PISCO models with and without training the decoder.

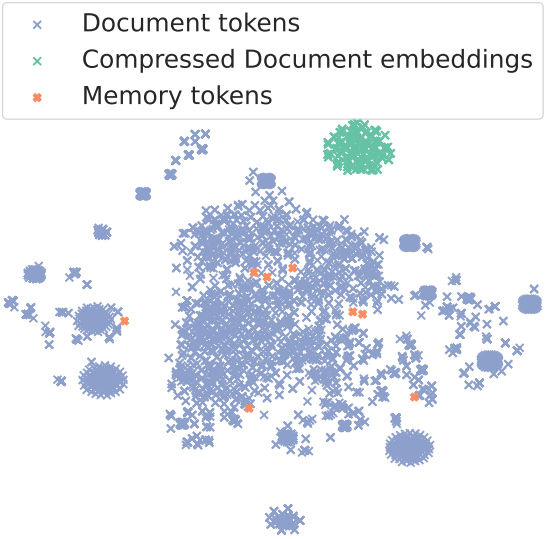

Figure 12: t-SNE visualization of document tokens, memory tokens, and compressed document embeddings. The compressed embeddings lie outside the token distribution, supporting why freezing the decoder in a compression model is ineffective.

by mapping the embeddings to logits space using the LLM head. An example of the results is provided in Table 9. Most top tokens correspond or are close to some token in the compressed text. We also recover the spatial specialization shown on Figure 9.

I Attempted pretraining tasks

As described in §4.3, our initial approach focused on designing more complex pretraining tasks. Pretraining followed the lines and configurations of (Rau et al., 2024b). The pretraining task we implemented consisted in:

- Auto-encoding (AE): the compressed repre-

Dataset		Document collection	Evaluation metric	N
Bio-QA (Krithara et al., 2023)		Pubmed	Recall	3837
Covid (Möller et al., 2020)		CORD-19	F1	2019
ParaphraseRC (Saha et al., 2018)		ParaphraseRC	Accuracy	13111
RobustQA (Han et al., 2023)	Lifestyle 1	LoTTE	F1	2198
	Writing			2694
	Science			1404
	Recreation			2090
	Technology			2064
MKQA (Longpre et al., 2021)		Wikipedia[8]	recall-3gram	10000

Table 7: Out-of-domain datasets characteristics. We use F1 for RobustQA and CovidQA since their labels are long and not suitable for computing the accuracy directly.

Decoder	Trained	Frozen
ASQA	0.72	0.65
HotpotQA	0.48	0.42
NQ	0.65	0.72
TriviaQA	0.90	0.87
POPQA	0.66	0.51

Table 8: Performance (accuracy) on general domain with and without decoder training.

sentation of a single document as well as a task-specific token <AE> is prompted to the decoder during pretraining: the labels is the plain text document.

- Text Continuation (TC): as for general LM training, this task prompts the decoder with the compressed representation of the document and its task is to generate the following text.

- Keyword-Based Text Continuation (KBTC): A potential concern –especially since auto-encoding works so well even with high compression rates– was that accessing information within the middle of texts while working on their compressed representations was difficult. To address this, the keyword-based text continuation task prompted the decoder with the compressed document as well as a small sequence (the keyword) extracted randomly from the compressed text. Its target was to generate what followed this keyword in the text. Here also it is possible to reach a very high Rouge-L in string reconstruction, with no effect on final QA performance. This showed that lost-in-the-middle effect was not really a concern within compressed documents.

- Multi-document Keyword-Based Text Continuation (multi-KBTC) is identical to Keyword-Based Text Continuation except that multiple encoded documents are prompted to the decoder at once. The motivation here was to address a potential between-document lost-in-the-middle effect.

In all cases, the loss was the cross-entropy loss on the target generation. The configurations analysed in the paper consist of mixtures of these different tasks and are described on Table 10.

We also tested formulating these tasks as instruction tasks with corresponding prompts as in (Cheng et al., 2024), without further success.

J String Normalization for metric computation

To measure accuracy, F1 score or recall between a ground truth label and a prediction, we check that the normalized label is included in the normalized prediction. When multiple labels are possible, we take maximum values across the available labels. Normalization consists in:

- Converting the string to lowercase
- Removing punctuation
- Removing articles: “a”, “an”, “the”
- Standardizing word splits by replacing multiple spaces and line returns with a single space

Text	Logits attributions of the text embeddings
the Gardner-Harvey Paper company, installing a very large paper board machine, and in 1916 organized the Gardner Paper Board company and took over the old National Box Board company, which was at that time in the hands of the receiver. All three of these companies met with phenomenal success, due to the efforts of Colin Gardner, who it is conceded was one of the most brilliant business men of his day. He was a Republican, but took no active part in politics, nor did he care for fraternal connections. During the Civil war he served with the 100-	**Token 0**: receiver, hands, giornata, bo, boxes, national, box, board, old, nacional **Token 1**: cares, fr, connection, neither, cared, connections, actively, nor, politics, political, active, care **Token 2**: papers, paper, companies, company, harvey, harold, newspaper, pap, board, har **Token 3**: businesses, efforts, business, republican, gardens, republicans, bright, gard, , garden, days, effort, brains, day **Token 4**: papers, machines, paper, company, companies, boards, pap, board, installation, machine, installed, install **Token 5**: succeed, met, efforts, three, phenomen, succeeded, success, meet, , successful, effort, achievements, meeting **Token 6**: organ, giornata, company, paper, companies, boards, organiz, board, took, organisation, take, organized, organization **Token 7**: serv, rera, served, volunteers, serving, servants, with, during, civil, -, serves, service, serve
tenth of 21 Franciscan missions built in upper California. Soon Yankee traders, tourists and health seekers, followed by wealthy Easterners settled in Santa Barbara because of the mild winters. The mixture of newcomers and Spanish descendants has shaped the area for what it has become today. Accommodations: Santa Barbara boasts over 90 motels and hotels, plus numerous bed and breakfast inns, all of which provide over 4,500 rooms from the modest to the mos6t luxurious for both business travelers and tour	**Token 0**: attracted, accommod, , attract, mixture, spanish, mi, santa, new, mild, win, winter, kennis **Token 1**: accommod, mos, from, plus, room, over, both, rooms, thousand, kennis **Token 2**: accommod, california, bo, biologie, santa, area, acc, plaat, områ, accom, over, accommodate, área, accommodation, kennis **Token 3**: accommod, countless, mot, hotels, beds, plus, hotel, bed, numerous, accommodation **Token 4**: lower, california, building, built, upper, missions, mission, up, build, francis **Token 5**: shape, today, area, spanish, descend, new, span, sha, accom, areas, accommodation, shapes, shaped **Token 6**: attracted, settled, yan, sett, health, healthy, settlement, healthcare, eastern, settle, wealthy, traders, tourists, kennis **Token 7**: businesses, travel, luxury, tourist, business, tour, tours, both, tournament, tourists, alike

Table 9: Text and the logits attribution of its memory embeddings: for each memory embedding, we compute the top-10 tokens using the head matrix of the decoder. Almost all top tokens correspond to a token in the text. Each memory embedding puts more emphasis on some part of the text.

Pretraining Mixtures			
AE	TC	KBTC	multi-KBTC
0.5	0.5	0.	0.
0.25	0.25	0.5	0.
0.	0.5	0.5	0.
0.	0.25	0.75	0.
0.	0.25	0.25	0.5

Table 10: Tested pretraining configurations. All were run with compression rate of 16 and 128.