

OSCAR: Online Soft Compression And Reranking

Maxime Louis* Thibault Formal Hervé Dejean Stéphane Clinchant
NAVER LABS Europe

Abstract

Retrieval-Augmented Generation (RAG) enhances Large Language Models (LLMs) by integrating external knowledge, leading to improved accuracy and relevance. However, scaling RAG pipelines remains computationally expensive as retrieval sizes grow. To address this, we introduce OSCAR, a novel query-dependent online soft compression method that reduces computational overhead while preserving performance. Unlike traditional hard compression methods, which shorten retrieved texts, or soft compression approaches, which map documents to continuous embeddings offline, OSCAR dynamically compresses retrieved information at inference time, eliminating storage overhead and enabling higher compression rates. Additionally, we extend OSCAR to simultaneously perform reranking, further optimizing the efficiency of the RAG pipeline. Our experiments demonstrate state-of-the-art performance with a $2\text{-}5\times$ speed-up in inference and minimal to no loss in accuracy for LLMs ranging from 1B to 24B parameters. The models are available at: huggingface.co/collections/naver/oscar.

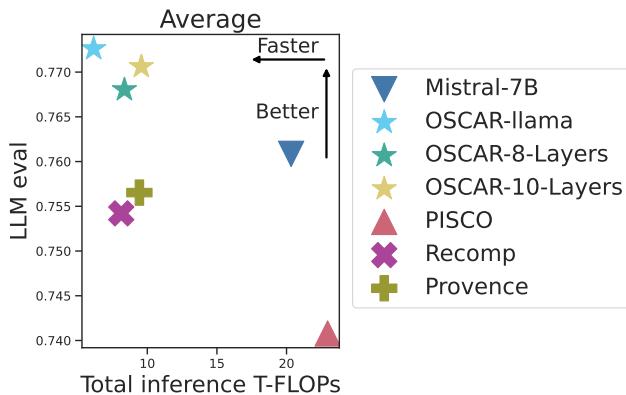


Figure 1: OSCAR models enable faster end-to-end inference with retrieval as well as improved accuracy compared to hard compression methods.

1. Introduction

Retrieval-Augmented Generation (RAG) [1, 12, 23] has become pivotal for solving a wide range of natural language processing challenges. RAG enhances Large Language Models (LLMs) by leveraging retrieved documents from curated datasets, enabling more accurate, well-grounded, and up-to-date responses. However, one major issue when scaling up RAG pipelines is the high computational cost, which increases quadratically with the number of tokens.

To improve efficiency, a natural idea consists in compressing the retrieved documents. To achieve this, there are two distinct families of methods. First, *hard* compression methods focus on textual compression: associating each retrieved document with a shorter text containing the useful semantic information. Second,

soft compression methods focus on directly mapping retrieved texts to continuous embedding spaces.

On one hand, hard compression produces texts, either by summarization or pruning [4, 18, 44, 45, 47]. Focusing on the surface form allows to develop LLM-agnostic methods, which are modular and more universal – at the expense of efficiency which remains limited due to small compression rates. Most methods perform *online* query-dependent compression of the retrieved documents: they compress the documents on the fly after retrieval without any offline pre-computation.

On the other hand, another class of methods focuses on the soft compression of retrieved documents into a set of embedding vectors that can be directly fed to the generator LLM in place of the usual token embeddings [3, 9, 14, 26, 37] – usually leading to higher compression rates. Most existing approaches in this direction perform *offline* compression of the document collection to optimize inference latency.

In this paper, we propose to bridge the gap between these two families by introducing OSCAR – for Online Soft Compression And Reranking – a query dependent online compression model. We argue that online adaptation of soft compression is particularly relevant, as it adjusts compression based on the current query, potentially achieving higher lossless compression rates [30]. Furthermore, building on recent observations by Chirkova et al. [4], we equip OSCAR with document reranking capabilities. Since reranking is an integral part of standard RAG pipelines, it makes the compression almost free. We show through experiments that OSCAR models built around backbone

LLMs ranging from 1B to 24B parameters enable 2 – 5x faster inference with little to no performance drop, thus achieving state-of-the-art compression performance for RAG.

To summarize, our contributions are:

- A state-of-the-art, fast and effective query-dependent online soft compression method for RAG: OSCAR,
- An extension of OSCAR which enables simultaneous reranking and compression of retrieved documents,
- Ablations of the key components of OSCAR and evaluations of its robustness to different RAG settings.

After discussing related works in §2, we describe the OSCAR method in §3. Main accuracy/efficiency results are presented in §4.2. In §4.3, we show how OSCAR models perform on various backbones. In §4.4, we further report results of the models with joint reranking capabilities. We finally present in §5 additional evidences of OSCAR robustness to RAG settings.

2. Related Works

There exists multiple research directions to improve RAG efficiency.

Long context optimizations for RAG. RAG scaling problems relate to the long-context (in)abilities of LLMs which is an active area of research. K/V caching techniques enable faster long context handling by diminishing the number of operations in self-attention [6, 21, 24]. FINCH [5] is more specifically designed for RAG: the retrieved content is chunked and only a small portion of the keys and values is kept in cache for each chunk for the subsequent attention computations – but compression rates remain limited. TurboRAG and block-attention RAG [27, 41] propose to modify the attention causal mask to compute attention independently on each retrieved documents, while the query still attends to each previous token in the context. Their methods require fine-tuning the LLM used, and achieve a 4× speed-up for contexts above 8k tokens. These K/V compression methods have an overhead which makes them prohibitive for contexts shorter than 6-8k tokens.

Hard compression methods aim at shortening the retrieved documents by means of summarization or pruning. Most of them have limited compression rates due to the nature of text but are agnostic to the LLM used for generation. Provence [4] proposes to fine-tune a DeBERTa [13] model to prune retrieved contexts. Provence is fast, prunes the context in a query-

dependent fashion and allows the simultaneous reranking of the retrieved documents – making pruning essentially free in a standard RAG pipeline. Extractive RECOMP [45] prunes contexts based on sentences embeddings but is limited by its independent processing of each retrieved sentence and the query itself. Abstractive RECOMP summarizes input contexts using an autoregressive LLM: the efficiency improvement is less clear than Provence since generating the summary is an expensive operation. Other methods include FILCO [44] or COMPACT [47], which also generate pruned contexts autoregressively.

Soft compression methods aim at compressing retrieved documents into vector representations, often to be used as input embeddings or K/V cache to the LLM used for generation. These methods generally achieve higher compression rates but require a training specific to the LLM used for generation. xRAG [2] proposes to use retrieval embeddings as precomputed compressed representations, and trains an adapter MLP to map these embeddings into inputs for the LLM – performances remain however limited. COCOM [37], building on Chevalier et al. [3], Ge et al. [9], proposes an end-to-end training pipeline where both the compression LLM and the generation LLM are fine-tuned using a large QA dataset. PISCO [26] is an extension of COCOM trained by sentence-level distillation from a teacher LLM: it allows to compress contexts by a factor of 16× with very limited performance drops. All these approaches process documents independently from the query – attempting to compress all the information of the retrieved documents into the (compressed) vector representation. FiD-light [14] proposes a form of query-dependent soft compression by using an encoder-decoder LLM, where the encoder is fed in parallel with the input query and each retrieved document. FiD-light decoder then takes only the first 50 hidden states for each document and thus has a very limited compression rate.

3. Method

Figure 2 provides an overview of OSCAR training and inference pipelines. A *compressor* LLM maps each document-query pair to an embedding space and a *generator* LLM generates an answer, receiving as input these embeddings and the (full) query. One of the key challenges in developing OSCAR was to enable a fast compression technique, unlike Chevalier et al. [3], Louis et al. [26], Rau et al. [37] where the authors use a generator-sized LLM for compression.

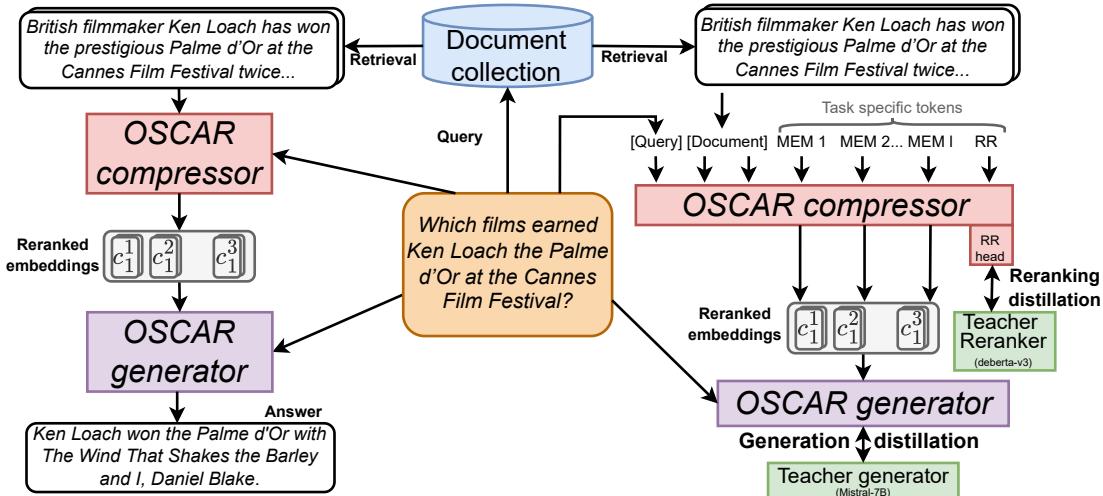


Figure 2: Overview of OSCAR inference (left) and training (right).

Query-dependent soft compression The compression procedure is shown in Figure 2 (right). It is similar to Ge et al. [9], Louis et al. [26], Rau et al. [37], except that the query is also used when compressing the documents. In details, the query q , the i -th retrieved document d_i , a set of memory tokens $\text{MEM}_{i=1\dots l}$ are fed forward to a *compressor* LLM C . We use the hidden states corresponding to each memory token as the query-dependent embedding representations $(c_i^1, \dots, c_i^l) := \mathbf{c}_i = C(q, d_i)$ of the document.

Compressor architecture Unlike PISCO, xRAG, or COCOM [2, 26, 37], OSCAR is intended to operate in an online fashion with no possibility to pre-compute document compressions. Therefore, the compression needs to be fast. To do so, we propose two methods:

1. **OSCAR-N-Layers**: we construct headless transformers using the first N layers of the pretrained backbone (same architecture as the generator). Compression complexity scales with N , enabling a family of compressors. A key advantage is that OSCAR-N-Layers models require no pre-training to align hidden representations with the generator, as shown in §4.2.
2. **OSCAR-llama**: we use a smaller LLM, primarily llama-1B¹, as our compressor. We map the compressor hidden space with the generator hidden space using two fully connected layers with ReLU non-linearity. Initial experiments showed that learning this mapping required a larger-scale pretraining task (see Appendix Table 8). Thus, following Rau et al. [37], we pretrain the com-

pressor/generator LLM on auto-encoding and text continuation tasks using Fineweb data. Pretraining details are provided in Appendix I.

Generation The embedding representations \mathbf{c} of each document, as well as the query q , are fed to a *generator* LLM which generates the answer autoregressively. Since each document has been replaced by l embeddings, generation is much faster compared to the original text.

Generation distillation The training objective on the generator LLM is identical to PISCO [26]: we use sentence-level distillation from some teacher LLM, i.e., a cross-entropy loss on teacher-generated labels. The motivation for distillation is clear: we would like for the generator LLM to have identical outputs as the outputs of some reference LLM using the uncompressed texts.

Simultaneous reranking Building on the insights from Chirkova et al. [4], query-dependent online context compression shares significant similarities with the document reranking task. Rerankers, such as cross-encoders [32], refine the ranking produced by the initial retrieval step. Unlike retrieval models, which encode queries and documents independently, rerankers contextualize documents with respect to queries, thereby generating more informative and effective representations. Since rerankers are already part of effective RAG pipelines [36], using a single forward-pass to compute both the compression and the reranking operations makes the compression essentially free – provided the compression operation is not more expensive than commonly used rerankers.

¹meta-llama/Llama-3.2-1B-Instruct

To develop OSCAR with reranking capability, we add a reranking token RR to the compressor LLM prompt – as shown in Figure 2 (right). An additional dense layer then maps the reranking token hidden state to a predicted relevance score r_i between the document and the query. To train the reranking component, we simply rely on a pointwise distillation of reranking scores of an effective cross-encoder. Distillation is now a standard technique to train ranking models [8, 15, 25, 39], and was already shown to work in the context of prompt compression in Provence [4]. Note that OSCAR-based rerankers leverage LLM backbones, which have been shown to outperform traditional cross-encoders [28, 35, 42].

Training objective Overall, denoting a_1, \dots, a_r the answer generated by the teacher LLM from the documents d_i : $a_i \sim \mathcal{T}(\cdot | d_1, \dots, d_k, q, \mathbf{a}_{\prec i})$, then the training objective on the compressor C and generator \mathcal{G} is the sum of the cross-entropy loss computed on the decoder conditioned on the compressed documents and the query and an (optional) l_2 loss on the reranking scores (given the scores r' from the teacher):

$$\begin{aligned} \mathbf{c}_i, r_i &= (c_i^s)_{s=1, \dots, l} = C(q, d_i), \quad i = 1, \dots, k \\ \mathcal{L}(C, \mathcal{G}) &= - \sum_{i=1}^r \log \mathcal{G}(a_i | q, \mathbf{c}_1, \dots, \mathbf{c}_k, \mathbf{a}_{\prec i}) \\ &\quad \left[+ \lambda \sum_{i=1}^k (r_i - r'_i)^2 \right] \end{aligned}$$

where k denotes the total number of documents used for generation, and λ an hyperparameter controlling the contribution of the ranking loss. Note that most of the models in the experiments are trained without the reranking component, a setting in which models can be used as standalone context processors. In §4.4, we study joint training and show how OSCAR models can also be used as rerankers in a RAG pipeline. Also note that OSCAR training does not require any ground truth labels. Whether for generation or reranking, training is performed through distillation from golden teachers.

4. Experiments

4.1. Experimental setup

Our training dataset comprises questions from [26] along with 500k queries extracted from MS MARCO [31], resulting in a total of 893k queries². The document collection used for training is Wikipedia-KILT [34], preprocessed into chunks of 128 tokens. For each query, we retrieve the top- k documents using

²We will release the queries as well as the distillation labels upon publication

SPLADE-v3 [7, 22]³ and subsequently rerank them with a DeBERTa-v3 [13]-based reranker⁴ (a robust RAG setting as shown in [36]). We employ sentence-level distillation from Mistral-7B⁵, as recommended in [26].

During training, the number k of retrieved documents is set to 5. We empirically found that this value provides sufficient context for models to generalize to a larger number of documents at inference time while keeping training costs low. Each document is then compressed into l embedding vectors, where l is fixed for each OSCAR model. Specifically, OSCAR models with a compression rate of 16 use 8 memory embeddings per document – given 128-sized input documents. We use this setting for all the experiments – using less memory embeddings having a low impact on efficiency improvements. All generators LLMs are trained with LoRA [16] adapters. For OSCAR-N-Layers models, we experiment with $N = 5, 8, 10$. OSCAR-llama relies on Llama-3.2-1B [11]. All compressors are trained with full-fine tuning – which was consistently more effective than LoRA adapters. For joint training (§4.4), early experiments suggested that $\lambda = 0.05$ usually offers the best compromise (in terms of compression quality and reranking effectiveness) on the validation set – and we use this default value for all further corresponding experiments.

For most of the experiments, we train the models as standalone compressors (without joint training). In particular, in §4.2, we provide efficiency metrics for OSCAR when compared to competitive approaches. In §4.4, we study joint training, and consider a scenario where in practice, the cost of compressing documents is almost zero.

4.2. Main results

Performance After training, we evaluate all models on multiple datasets, including Natural Questions [20], TriviaQA [17], HotpotQA [46], ASQA [40], PopQA [29], and BIOASQ-12B [19]. For each query, we retrieve documents from either KILT or PUBMED. The primary evaluation metric relies on LLM-based assessment of responses [36], as it better captures answer quality beyond exact matches, following the procedure detailed in Appendix E. OSCAR models have seen 5 retrieved documents per query at training time, but we evaluate them – and all other models – in a setting with 10 documents to verify generalization to larger contexts.

³[naver/splade-v3](#)

⁴[naver/trecll22-crossencoder-debertav3](#)

⁵[huggingface/mistralai/Mistral-7B-Instruct-v0.2](#)

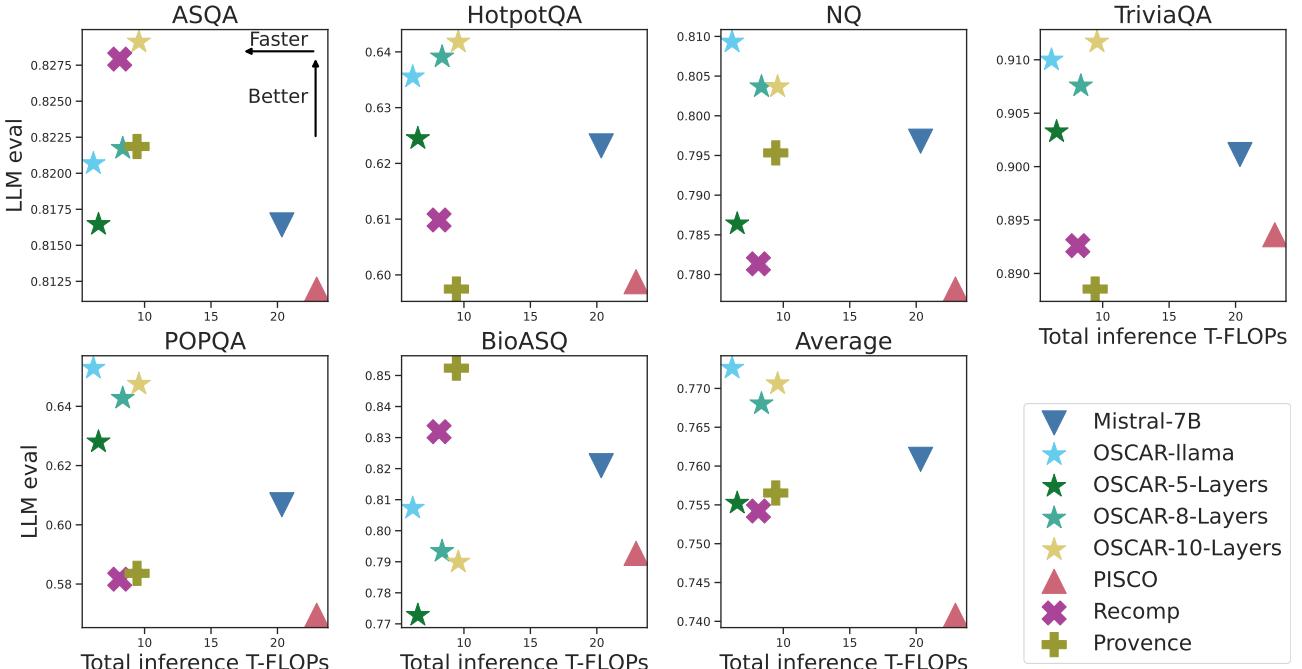


Figure 3: **LLM evaluation scores of each Mistral-7B-backboned models, in relation with the total number of floating point operations required at inference.** OSCAR models are faster and more effective on most datasets. OSCAR-llama in particular offers the best alternative.

Baselines We run identical evaluations on Provence [4] and Recomp [45] models as they are the state-of-the-art hard compression models for RAG. We also run evaluations of PISCO models, the state-of-the-art soft compression model, and on Mistral-7B with no compression for reference.

Computational efficiency To evaluate computational efficiency, we measure the number of floating-point operations required for both compression and answer generation. For consistency, we perform this calculation on a standardized input query of 128 tokens, concatenated with 10 documents of 128 tokens each or their compressed embeddings. Measurements are obtained using `torch.profiler`. Further details, including computation times and peak memory usage, are provided in Appendix D.

Figure 3 and Table 1 present our results. Across all datasets, OSCAR models achieve higher LLM evaluation scores than the Mistral-7B baseline while providing a 2.2–3.3× inference speed-up. Additionally, OSCAR outperforms lexical baselines like RECOMP and Provence on most datasets. Among OSCAR variants, OSCAR-llama is generally the strongest and fastest, though it requires pretraining (Appendix I). For OSCAR-N-Layers models, performance improves with more layers but at the cost of efficiency. Beyond 10 layers, accuracy plateaus while efficiency worsens (Appendix C). OSCAR-

5-Layers is slightly less effective than other compressors. Additional accuracy-based results (matching whether the label answer appears in the generation) are provided in Appendix A.2. Overall, OSCAR models enable faster inference and stronger performance than hard compression methods.

GPT evaluations In addition to a pointwise LLM evaluation, we use GPT-4 to perform pairwise comparisons of OSCAR-llama, Mistral-7B, PISCO and Provence. Results are reported in Figure 4 and confirm that OSCAR is on par with the uncompressed baseline and outperforms lexical pruning methods.

4.3. Other backbones

Unlike most hard compression methods for RAG [4,45], OSCAR models are backbone-specific and need to be retrained for every different generation LLM. To show how stable OSCAR training is, we produce models to improve RAG efficiency of Qwen2-7B⁶, Mistral-24B⁷ and Llama-1B⁸.

We report LLM evaluation results and efficiency measures in Table 1. OSCAR offers improvements in quality of responses for all 3 tested backbones, ranging from 1B to 24B parameters. Most interestingly, OSCAR-llama

⁶Qwen/Qwen2-7B-Instruct

⁷mistralai/Mistral-Small-24B-Instruct-2501

⁸meta-llama/Llama-3.2-1B-Instruct

	OSCAR wins	Tie	Mistral-7B wins
POPOQA	26.5%	55.3%	18.2%
ASQA	33.1%	37.1%	29.7%
HotpotQA	37.7%	37.0%	25.3%
TriviaQA	23.3%	58.7%	18.0%
NQ	30.8%	36.8%	32.4%
BioASQ	25.4%	43.1%	31.5%
	OSCAR wins	Tie	Provence wins
POPOQA	34.0%	51.1%	14.9%
ASQA	40.0%	31.2%	28.8%
HotpotQA	45.1%	30.5%	24.4%
TriviaQA	35.8%	49.2%	15.0%
NQ	37.1%	33.5%	29.4%
BioASQ	24.0%	43.5%	32.4%
	OSCAR wins	Tie	PISCO wins
POPOQA	34.2%	52.8%	13.0%
ASQA	38.2%	33.6%	28.2%
HotpotQA	40.6%	35.8%	23.6%
TriviaQA	25.7%	57.3%	17.0%
NQ	35.7%	39.4%	24.9%
BioASQ	29.9%	42.6%	27.6%

Figure 4: **GPT-4 pairwise comparisons.** OSCAR-llama, while faster, is on par – or better – than Provence, Recomp and PISCO.

model for Mistral-24B enables a $5\times$ decrease in computational complexity while improving the overall results. In fact, OSCAR efficiency improvements are proportional to the backbone size, and hence particularly advantageous for larger language models. Figure 9 shows the efficiency/performance comparisons of OSCAR with Recomp, Provence with Qwen2-7B backbone.

4.4. Adding reranking capability

Having demonstrated that OSCAR models function effectively as standalone compressors, with their efficiency closely tied to their backbone, we now train OSCAR models capable of both document compression and reranking. In a RAG pipeline incorporating reranking, the computational cost of compression becomes virtually negligible, as a single forward pass produces both compressed representations and reranking scores.

We report in Table 2 the performance of such jointly trained models under two evaluation settings: standalone, which corresponds to the previous setting (DeBERTa-v3 reranker), and e2e which corresponds to compressing documents reranked by the OSCAR model itself. Essentially, we observe no drop in performance between standalone and e2e settings, indicating that OSCAR effectively learns to rerank documents. This finding is further supported by OSCAR’s performance on the BEIR benchmark [43] where its reranking capabilities are nearly on par with the strong teacher model. Detailed BEIR results for individual datasets are provided in Appendix (Table 4). To match the teacher’s performance on BEIR, OSCAR requires an increased model depth to 16 layers. However, this model is less

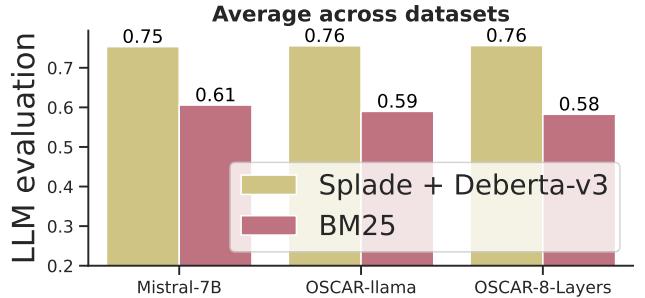


Figure 5: Testing the robustness of OSCAR to retrieval changes. OSCAR models performance drops are similar to the uncompressed backbone.

efficient, and its actual e2e performance (evaluated via LLM-based metrics or accuracy) remains unchanged.

5. OSCAR robustness

We assess OSCAR’s robustness for RAG pipelines by evaluating its performance with BM25 retrieval (no reranking) in §5.1 and with a large number of retrieved documents in §5.2.

5.1. Robustness to retrieval changes

In all training and test experiments so far, all documents were retrieved using SPLADE-v3 and reranked with a DeBERTa-v3-based reranker – a robust RAG setup [36]. Yet it still prompts the question of how OSCAR models perform when retrieval quality declines. In particular, the behavior of hard compression methods is clearly identifiable on noisy documents – and shown to be correctly handled by Provence [4] or Recomp [45]. It is more of an open question for soft compression models like OSCAR. To investigate this, we run evaluation experiments using BM25 [38] only (no reranking) and report results in Figure 5. Essentially, the performance drops of OSCAR models with respect to Mistral-7B are similar – indicating that OSCAR models are able to handle noisy documents. Detailed results for all datasets are found in Appendix B.

5.2. Long context abilities of OSCAR models

Since OSCAR models are trained with 5 retrieved documents, we investigate whether they remain able to extract and use information from a larger number of documents. Figure 6 shows the results when increasing the number of retrieved documents to up to 50 (which makes uncompressed contexts around 7k tokens). Note that as the number of documents increase, because of the quadratic cost of the attention, the larger compression rate of OSCAR models make them comparatively faster. With 50 documents, we measure $5\times$ less FLOPs for OSCAR than Mistral-7B.

Backbone	Compressor	LLM evaluation score							Tera-Floating point operations		
		ASQA	HotpotQA	NQ	TriviaQA	POPQA	BIOASQ	Average	Inference	Compression	Total
Mistral-7B	No compression	0.82	0.62	0.80	0.90	0.61	0.82	0.76	20.33	0.	20.33
	RECOMP	0.83	0.61	0.78	0.89	0.58	0.83	0.75	7.29	0.84	8.13 (2.5×)
	Provence	0.82	0.60	0.80	0.89	0.58	0.85	0.76	7.63	1.80	9.43 (2.2×)
	PISCO	0.81	0.60	0.78	0.89	0.57	0.79	0.74	3.49	offline	3.49 (5.8×) ^a
	OSCAR-llama	0.82	0.64	0.81	0.91	0.65	0.81	0.77	3.49	2.66	6.15 (3.3×)
	OSCAR-5-Layers	0.82	0.62	0.79	0.90	0.63	0.77	0.76	3.49	3.04	6.53 (3.1×)
Llama-1B	No compression	0.69	0.48	0.66	0.81	0.52	0.76	0.65	2.85	0.	2.85
	OSCAR-5-Layers ^b	0.71	0.53	0.70	0.85	0.55	0.72	0.68	0.50	0.88	1.38 (2.1×)
Qwen-7B	No compression	0.80	0.67	0.78	0.91	0.65	0.84	0.78	18.94	0.	18.94
	OSCAR-8-Layers	0.81	0.61	0.78	0.90	0.64	0.80	0.76	3.17	5.07	8.25 (2.3×)
	OSCAR-llama	0.82	0.62	0.79	0.90	0.65	0.81	0.76	3.17	2.65	5.83 (3.2×)
Mistral-24B	No compression	0.82	0.71	0.80	0.92	0.70	0.85	0.80	64.29	0.	64.29
	OSCAR-llama	0.82	0.65	0.82	0.92	0.67	0.84	0.79	10.72	26.55	13.37 (4.8×)

Table 1: **Performance and efficiency for OSCAR models and baselines based on various backbones.** OSCAR models are more effective and faster than their backbones with no compression. OSCAR models are also more efficient than the two hard compression baselines Provence and Recomp.

^aPISCO is intended to be used offline with precomputed documents compressions but is a strong soft compression baseline.

^bWe do not train an OSCAR-llama with llama-32-1B backbone as it would not increase global efficiency.

Model	Setting	LLM evaluation score							BEIR
		ASQA	HotpotQA	NQ	TriviaQA	POPQA	BIOASQ	Average	
OSCAR-llama	standalone	0.83	0.64	0.80	0.91	0.66	0.80	0.77	52.8
	e2e	0.81	0.63	0.79	0.91	0.66	0.80	0.77	
OSCAR-8-Layers	standalone	0.82	0.64	0.81	0.91	0.64	0.79	0.77	52.5
	e2e	0.81	0.63	0.79	0.90	0.64	0.78	0.76	
OSCAR-10-Layers	standalone	0.82	0.64	0.81	0.91	0.64	0.80	0.77	54.3
	e2e	0.81	0.65	0.82	0.91	0.66	0.78	0.77	

Table 2: **LLM evaluation and reranking performance on the BEIR benchmark (mean nDCG@10 on the 13 BEIR datasets).** We report results for three efficient OSCAR models on two RAG settings (with a Mistral-7B decoder). The reranking performance of the teacher (based on DeBERTa-v3) is 55.4. Note that the performance on the standalone setting might slightly differ from previous Tables as these models are trained with a different loss (joint training).

5.3. Relation between embeddings and query

While OSCAR offers greater computational efficiency and accuracy, it lacks the interpretability of hard compression methods. In this section, we offer a glimpse into the content of the compressed embeddings, to assess that they do indeed depend on the query. First, Figure 7 uses a needle-in-a-haystack test [10] to show that cosine similarity between compressed embeddings and text tokens is highest near the needle, indicating strong query dependence. Second, Figure 8 examines OSCAR embeddings via logit attributions [33], revealing that they align closely in vocabulary space with

context relevant to the query.

6. Conclusion

In this paper, we introduce OSCAR, the first online soft compression methods for RAG. The key challenge is designing an efficient compression technique for an online setting, which we address with two variants: one using a small compressor model and another leveraging the generator’s early layers. We compare OSCAR against hard compression methods (RECOMP, Provence) and soft ones (PISCO), showing that query-dependent compression is more effective than query-independent ap-

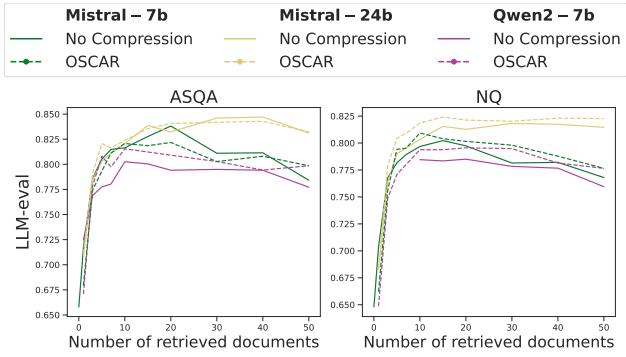


Figure 6: LLM evaluations of OSCAR models and their backbones with an increasing number of retrieved documents.

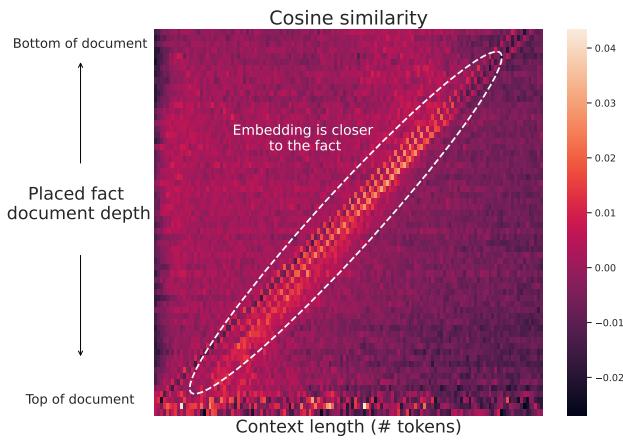


Figure 7: Cosine similarity between document embeddings and document individual tokens, on a needle-in-a-haystack test. The document embeddings are more similar to the area around the needle, indicating that the compression focuses on query-related elements.

Cats have excellent night vision and can see at one sixth the light level required for human vision. This is partly the result of cat eyes having a tapetum lucidum, which reflects any light that passes through the retina back into the eye, thereby increasing the eye's sensitivity to dim light. Snakes are elongated, limbless reptiles of the suborder Serpentes. Like all other squamates, snakes are ectothermic, amniote vertebrates covered in overlapping scales.

Query	What allows cats to see in light levels six times dimmer than what humans require?
Embedding logits attributions	dim, <MEMO>, any, cat, eye, back, reflect, tap, sensitivity, thereby, reflected, any el, lim, covered, Lim, Sub, overl, am, sub, of, cats, Cover, Se

Figure 8: Logits attributions on OSCAR embeddings. Attributed tokens predominantly correspond to an area of the context relevant to the query.

proaches. OSCAR also outperforms or matches hard pruning methods while being more efficient, proving the potential of soft compression. Additionally, we extend OSCAR with reranking, inspired by [4], reducing compression costs by factorization in the RAG pipeline. Our ablations analyze different backbones, weak re-

triever performance, behavior with large number of retrieved documents.

References

- [1] 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. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022. [1](#)
- [2] Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. *arXiv preprint arXiv:2405.13792*, 2024. [2](#), [3](#)
- [3] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*, 2023. [1](#), [2](#)
- [4] Nadezhda Chirkova, Thibault Formal, Vassilina Nikoulina, and Stéphane Clinchant. Provence: efficient and robust context pruning for retrieval-augmented generation. In *The Thirteenth International Conference on Learning Representations*, 2025. [1](#), [2](#), [3](#), [4](#), [5](#), [6](#), [8](#), [12](#)
- [5] Giulio Corallo and Paolo Papotti. Finch: Prompt-guided key-value cache compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1517–1532, 2024. [2](#)
- [6] Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for kv cache compression. *arXiv preprint arXiv:2406.11430*, 2024. [2](#)
- [7] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292, 2021. [4](#)
- [8] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. From distillation to hard negative sampling: Making sparse neural ir models more effective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 2353–2359, New York, NY, USA, 2022. Association for Computing Machinery. [4](#)
- [9] Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023. [1](#), [2](#), [3](#)
- [10] gkamradt. Needle in a haystack - pressure testing llms, 2024. [7](#)
- [11] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Srivankumar, Artem Korenev, Arthur Hinsvark, Arun Rao,

Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Jun-teng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhota, Lauren Rantala-Yearly, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsim-poukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Al-rassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shao-liang Nie, Sharan Narang, Sharath Raparth, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeni, Vitor Albiero, Vladan Petrovic, Weiwei Chu, Wen-han Xiong, Wenyin Fu, Whitney Meers, Xavier Mar-

tinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcuate, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelle, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madijan Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov,

- Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghatham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaoqian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. [4](#)
- [12] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020. [1](#)
- [13] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2021. [2, 4](#)
- [14] Sebastian Hofstätter, Jiecao Chen, Karthik Raman, and Hamed Zamani. Fid-light: Efficient and effective retrieval-augmented text generation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1437–1447, 2023. [1, 2](#)
- [15] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. Improving efficient neural ranking models with cross-architecture knowledge distillation, 2021. [4](#)
- [16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. [4](#)
- [17] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017. [4](#)
- [18] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacl-HLT*. Minneapolis, Minnesota, 2019. [1](#)
- [19] Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Palioras. Bioasq-qa: A manually curated corpus for biomedical question answering. *Scientific Data*, 10(1):170, 2023. [4](#)
- [20] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019. [4](#)
- [21] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023. [2](#)
- [22] Carlos Lassance, Hervé Déjean, Thibault Formal, and Stéphane Clinchant. Splade-v3: New baselines for splade. *arXiv preprint arXiv:2403.06789*, 2024. [4](#)
- [23] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020. [1](#)
- [24] Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhan-chao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen. A survey on large language model acceleration based on kv cache management. *arXiv preprint arXiv:2412.19442*, 2024. [2](#)
- [25] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173, Online, 2021. Association for Computational Linguistics. [4](#)
- [26] Maxime Louis, Hervé Déjean, and Stéphane Clinchant. Pisco: Pretty simple compression for retrieval-augmented generation. *arXiv preprint arXiv:2501.16075*, 2025. [1, 2, 3, 4, 12, 14](#)
- [27] Songshuo Lu, Hua Wang, Yutian Rong, Zhi Chen, and Yaohua Tang. Turborag: Accelerating retrieval-augmented generation with precomputed kv caches for chunked text. *arXiv preprint arXiv:2410.07590*, 2024. [2](#)

- [28] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning llama for multi-stage text retrieval, 2023. [4](#)
- [29] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 2022. [4](#)
- [30] Alliot Nagle, Adway Girish, Marco Bondaschi, Michael Gastpar, Ashok Vardhan Makkuva, and Hyeji Kim. Fundamental limits of prompt compression: A rate-distortion framework for black-box language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. [1](#)
- [31] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human-generated machine reading comprehension dataset. 2016. [4](#)
- [32] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019. [3](#)
- [33] nostalgebraist. Interpreting GPT: The logit lens. Less Wrong, 2020. [7](#)
- [34] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*, 2020. [4](#)
- [35] Ronak Pradeep, Sahel Sharifymoghadam, and Jimmy Lin. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze!, 2023. [4](#)
- [36] David Rau, Hervé Déjean, Nadezhda Chirkova, Thibault Formal, Shuai Wang, Vassilina Nikoulina, and Stéphane Clinchant. Bergen: A benchmarking library for retrieval-augmented generation. *arXiv preprint arXiv:2407.01102*, 2024. [3, 4, 6, 12, 14](#)
- [37] David Rau, Shuai Wang, Hervé Déjean, and Stéphane Clinchant. Context embeddings for efficient answer generation in rag. *arXiv preprint arXiv:2407.09252*, 2024. [1, 2, 3, 15](#)
- [38] Stephen E Robertson, Steve Walker, MM Beaulieu, Mike Gatford, and Alison Payne. Okapi at trec-4. *Nist Special Publication Sp*, pages 73–96, 1996. [6](#)
- [39] Ferdinand Schlatt, Maik Fröbe, Harrisen Scells, Shengyao Zhuang, Bevan Koopman, Guido Zuccon, Benno Stein, Martin Potthast, and Matthias Hagen. A systematic investigation of distilling large language models into cross-encoders for passage re-ranking, 2024. [4](#)
- [40] Ivan Stelmakh, Yi Luan, Bhuvan Dhingra, and Ming-Wei Chang. Asqa: Factoid questions meet long-form answers. *arXiv preprint arXiv:2204.06092*, 2022. [4](#)
- [41] East Sun, Yan Wang, and Lan Tian. Block-attention for efficient rag. *arXiv preprint arXiv:2409.15355*, 2024. [2](#)
- [42] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents, 2024. [4](#)
- [43] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [6](#)
- [44] Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*, 2023. [1, 2](#)
- [45] Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*, 2023. [1, 2, 5, 6](#)
- [46] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018. [4](#)
- [47] Chanwoong Yoon, Taewho Lee, Hyeyon Hwang, Minbyul Jeong, and Jaewoo Kang. Compact: Compressing retrieved documents actively for question answering. *arXiv preprint arXiv:2407.09014*, 2024. [1, 2](#)

A. Additional results

A.1. Extensive comparison with Qwen2-7B

We showed in Figure 3 the efficiency/performance plots for Mistral-7B backbone, including comparison with Provence, Recomp and the uncompressed backbone. We provide in Figure 9 the same results but for Qwen2-7B. OSCAR-llama models remains the best compression model, both in terms of efficiency and LLM evaluation score. In particular, OSCAR-llama score is on average 4 points above Provence and 6 points above RECOMP.

A.2. Accuracy results

In Section 4.2, we reported a score determined with an LLM evaluation of the generated responses, described in Appendix E. In this section, we provide accuracy results for each models. We define the accuracy here as 1 if the ground truth label is included as a sub-string of the generated answer, after normalization as proposed in [4, 26, 36]. Table 3 provides these results for all studied backbones while Figure 10 shows the efficiency/performance plots for mistral-7B-backboned models.

A.3. Full results on the BEIR dataset

We report in Table 4 the detailed BEIR results on individual datasets.

B. Detailed effects of BM25 retrieval

In section 5.1, we provided averaged effect across datasets of the change of retrieval/reranking pipeline. We provide in Figure 11 results for individual datasets. These results show that performance is preserved across all datasets, although it is likely that retrieval for Bioasq is noisier.

C. Influence of number of compressor layers

In Section 3, we proposed constructing a transformer by utilizing the initial layers of the backbone to develop an efficient compressor that operates without requiring pretraining. Since the inference cost scales with the number of retained layers, it is important to examine the impact of reducing the number of layers used for compression. This analysis is presented in Figure 12, where the performance appears to plateau around 4-5 layers for Mistral-7B. Notably, increasing the number of layers beyond 10 does not seem to justify the additional computational cost.

D. More about efficiency

D.1. Setup to measure efficiency

In Section 4.2, we measured efficiency of models based on the total number of floating-point operations as it is the primary indicator of the computational complexity. To generate these measures, we generate fake inputs of standardized size (a query/prompt of 128 tokens associated with 10 128-token documents) and do compression and the generation of a 32 token answer⁹ from an input of size computed from the compression rate of each method (e.g., for OSCAR with compression rates 16, the input to the generator is of size $128 + 10\frac{128}{16}$). To compute FLOP we set the batch size to 1 and use `torch.profiler`. We provide additional measures regarding inference time and peak GPU memory in each case. We set the batch size at 256 (32 for the larger Mistral-24B) to compute the inference time (simulating a busy service) and the peak GPU memory. In all cases we use hugging face implementation of the models. For memory usage and inference time, we average the results over 10 runs.

Results are shown in Table 5. Gains observed in terms of floating-point operations mostly translate to computational time (as can be expected for sufficiently large batch sizes). OSCAR models enable to save about 50-75% of memory across the various backbones. In practice, this larger batch sizes to be used and hence further latency improvements.

E. LLM evaluation

Our primary evaluation metric follows the LLM-based assessment proposed in [36]. This approach utilizes the SOLAR-107B model¹⁰ prompted to determine the correctness of a predicted answer by comparing it against both the given question and a reference answer. This metric can be viewed as an enhanced version of traditional accuracy, as it remains more robust to surface-level variations that do not alter the underlying semantic content. The prompt used is given in Figure 13.

F. Impact of the compression rate

In the main results we set the compression rate to 16. This choice came from the conclusion that compressing further does not sensibly increase the efficiency of the method, while it renders the compression more difficult. Indeed, while compressing by a factor 16 decreases generation the number of tera-floating point operations for generation from 20.33 down to 3.49, compressing by a factor of 128 only brings down this cost to 2.40 T-FLOP. Still, we show in Table 6 the results with mistral-

⁹The analysis for generated answers of 128 or 256 tokens leads to similar conclusions

¹⁰[huggingface/upstage/SOLAR-10.7B-Instruct-v1.0](https://huggingface.co/upstage/SOLAR-10.7B-Instruct-v1.0)

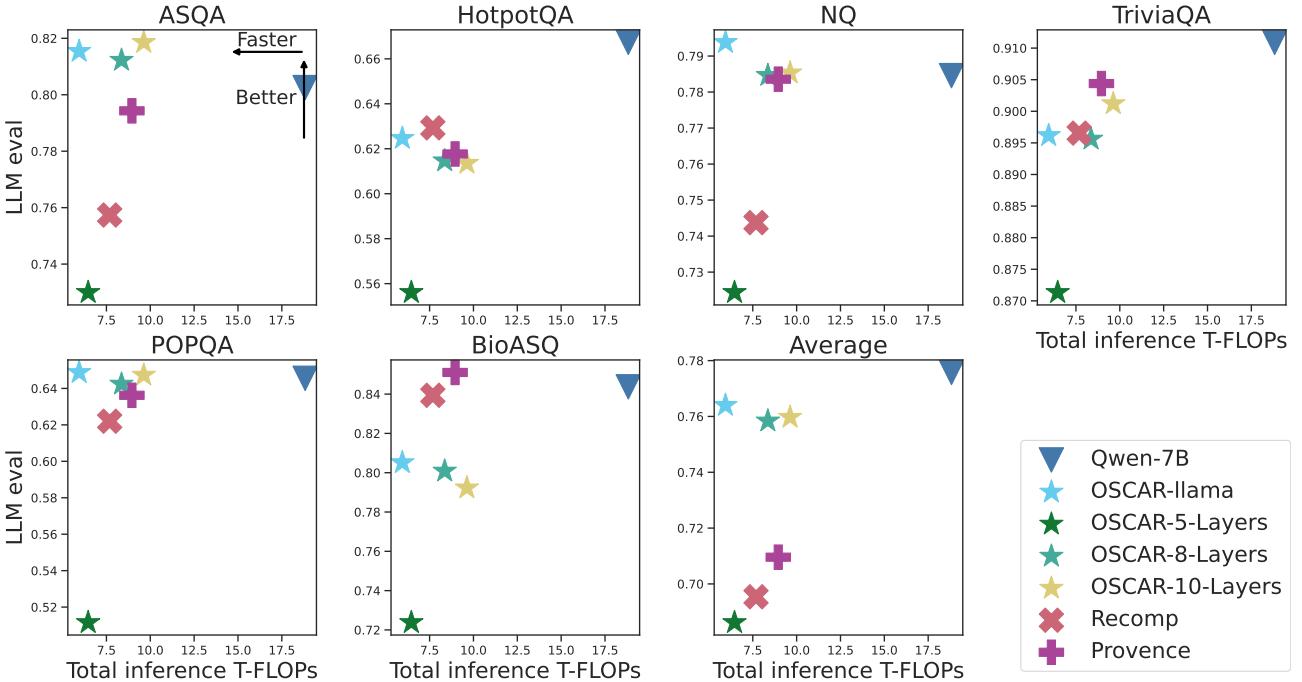


Figure 9: **LLM evaluation of Qwen2-7B-backboned models**, in relation with the total number of floating point operations required at inference. OSCAR-llama model is the fastest and best compression model.

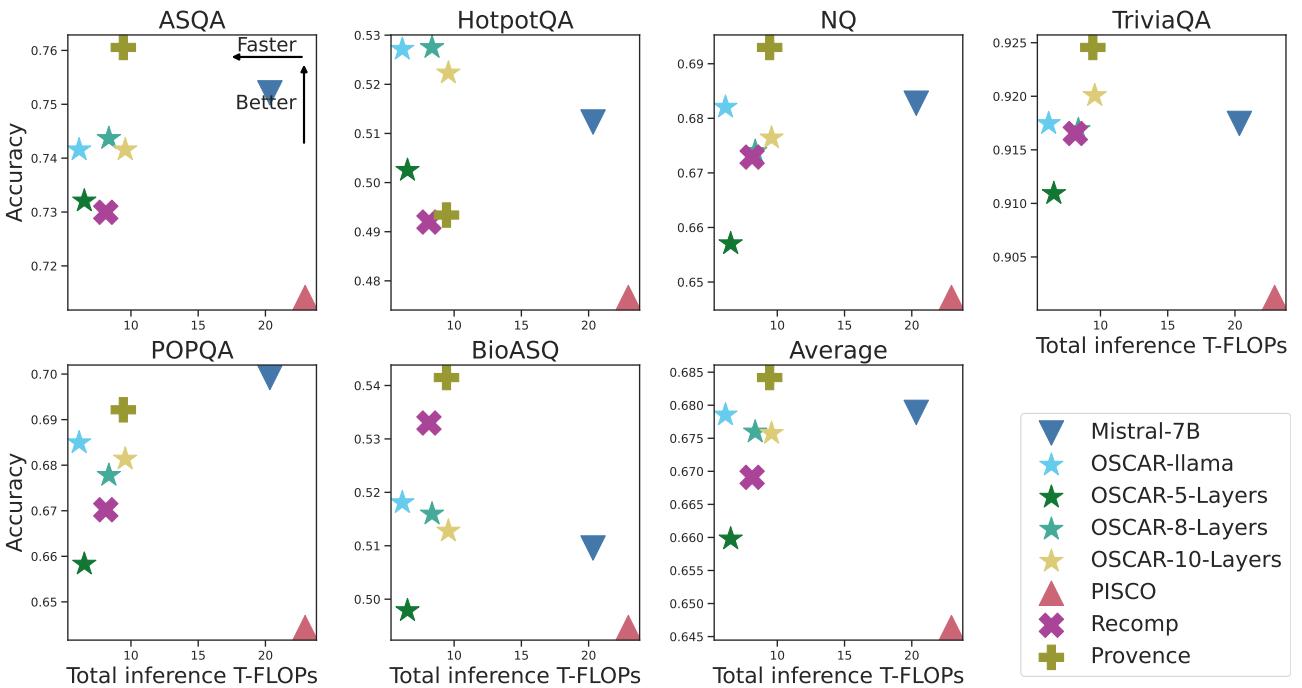


Figure 10: **Accuracy scores of each Mistral-7B-backboned models**, in relation with the total number of floating point operations required at inference. OSCAR models are faster and better on most datasets.

7B-backboned OSCAR models trained with compression rate $\times 128$.

G. Prompts

H. OSCAR training hyperparameters

We provide in this section details enabling the replication of OSCAR training results. Note that all OSCAR models for all backbones (from llama-1B all the way to mistral-24B) were trained using this configuration. Our training code relies on HuggingFace trainer and an

Backbone	Compressor	ASQA	HotpotQA	NQ	TriviaQA	POPQA	BIOASQ	Average
Mistral-7B	No compression	0.75	0.51	0.68	0.92	0.70	0.51	0.68
	RECOMP	0.73	0.49	0.67	0.92	0.67	0.53	0.67
	Provence	0.76	0.49	0.69	0.92	0.69	0.54	0.68
	PISCO	0.71	0.48	0.65	0.90	0.64	0.49	0.65
	OSCAR-llama	0.74	0.53	0.68	0.92	0.68	0.52	0.68
	OSCAR-5L	0.73	0.50	0.66	0.91	0.66	0.50	0.66
Llama-3.2-1B	OSCAR-8L	0.74	0.53	0.67	0.92	0.68	0.52	0.68
	No compression	0.61	0.35	0.54	0.82	0.59	0.40	0.55
	OSCAR-5L	0.64	0.43	0.59	0.86	0.59	0.46	0.60
Qwen-2-7B	No compression	0.70	0.51	0.64	0.90	0.64	0.53	0.65
	OSCAR-8L	0.72	0.50	0.64	0.91	0.67	0.51	0.66
	OSCAR-llama	0.72	0.51	0.66	0.91	0.68	0.52	0.67
Mistral-24B	No compression	0.74	0.54	0.68	0.92	0.70	0.53	0.68
	OSCAR-llama	0.75	0.54	0.70	0.93	0.70	0.53	0.69

Table 3: **Accuracy for OSCAR models and baselines based on various backbones.** Accuracy results are in line with LLM evaluations: OSCAR models are stronger in general than their uncompressed backbones.

Corpus	DeBERTa-v3	OSCAR-llama	OSCAR-8-Layers	OSCAR-10-Layers	OSCAR-16-Layers
TREC-COVID	88.3	83.1	81.4	84.4	86.1
NFCorpus	37.5	34.2	34.5	36.5	36.9
NQ	66.7	63.3	61.3	64.1	67.2
HotpotQA	74.5	72.9	72.2	73.5	74.3
FiQA-2018	47.8	42.7	40.8	44.3	47.5
ArguAna	29.8	29.5	32.5	32.4	34.0
Touché-2020	33.5	29.3	31.6	31.9	31.3
Quora	84.8	86.0	86.0	87.5	87.9
DBPedia	48.9	47.5	46.5	48.2	49.2
SCIDOCs	19.2	17.2	17.6	18.6	19.3
FEVER	86.6	83.6	83.1	84.1	83.9
Climate-FEVER	27.4	25.9	24.2	25.3	26.3
SciFact	75.8	71.2	71.2	75.2	75.5
average	55.4	52.8	52.5	54.3	55.3

Table 4: **nDCG@10 on the 13 open BEIR datasets.** DeBERTa-v3 is the reranker teacher used to train OSCAR models.

adaptation of the public Bergen library [36].

Note that OSCAR-N-layer models are directly trained by fine-tuning on the distillation data described in Section 4.1: they do not need pretraining. This is a similar effect as in [26]. On the contrary, OSCAR-llama models need a pretraining described in Appendix I.

Hyper-parameter search to build OSCAR models
We took hyperparameters from [26] and only conducted a small grid search over 8 values to tune the

learning rate required on the compressor, as we noticed performances were underwhelming with identical learning rates on compressor and generator. The total computation time to train an OSCAR model around Mistral-7B is around 50 hours on a single high-end GPU.

I. OSCAR-llama pretraining

OSCAR models using llama-1B as compressor models without any pretraining failed to reach satisfying per-

Backbone	Compressor	Inference time (ms) [†]				Peak memory (Gb) [‡]
		Architecture	Parameters	Inference	Compression	
Mistral 7B	No compression	-		141.6	0.	141.6
	OSCAR-5L	1.2B		33.0	18.0	51.0 (2.3×)
	OSCAR-8L	1.91B		33.0	28.8	61.8 (2.2×)
	OSCAR-llama	1.1B		33.0	17.1	50.1 (2.8×)
Llama 3.2 1B	No compression	-		30.2	0.	30.2
	OSCAR-5L			8.3	5	13.3 (2.3×)
Qwen-2-7B	No compression	-		109	0.	109
	OSCAR-5L	1.7B		25.6	15.2	40.8 (2.7×)
	OSCAR-llama	1.1B		25.6	17.1	42.7 (2.6×)
Mistral-24B	No compression	-		383.2	0.	383.2
	OSCAR-llama	1.1B		67.9	17.1	85.0 (4.5×)

Table 5: **Inference time and memory for each model.** Computed with 128-token queries and 10 128-token retrieved documents. [†] computed with batch size 256 (32 for Mistral-24B) but brought down to individual query cost [‡]for a batch of size 32.

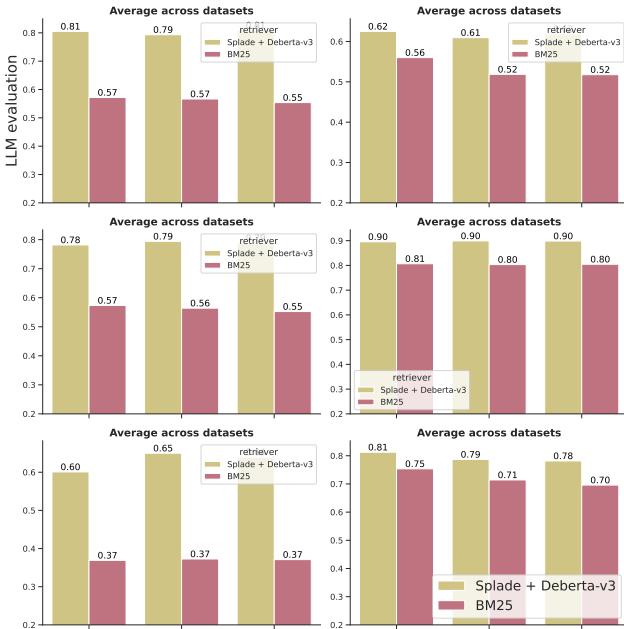


Figure 11: Effect of retrieval on OSCAR models, per dataset, compared to their uncompressed backbone.

formances (see Table 8). We attribute this effect to the need of building a map between the compressor hidden space and the decoder hidden space. To achieve this, we use the same pretraining as proposed in [37], with identical hyperparameters and a pretraining dataset consisting of chunks preprocessed from fineweb¹¹. Note that experiments show that as long as some form of extended pretraining is done which requires the decoder to use embeddings produced by the compressor, the ensuing OSCAR-llama models are strong. Therefore,

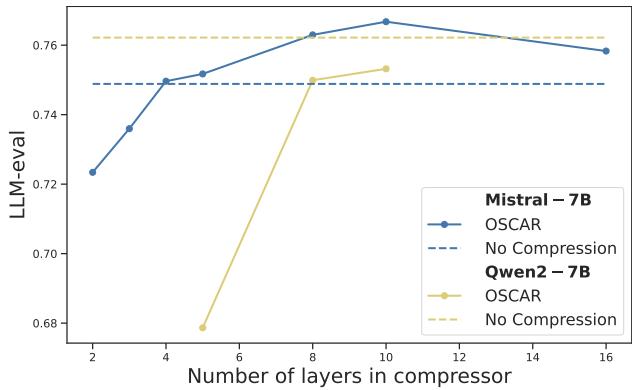


Figure 12: Average accuracy on general domain datasets for OSCAR models where the compressor has a variable number of layers. Performances increase with the number of layers but plateau above 8-10 layers for both Qwen2-7B and Mistral-7B backbones.

Model	ASQA	HotpotQA	NQ	TriviaQA	POPQA	BIOASQ
Mistral-7B	0.82	0.62	0.80	0.90	0.61	0.82
PISCO x128	0.81	0.57	0.75	0.89	0.51	0.77
OSCAR-llama x16	0.82	0.64	0.81	0.91	0.65	0.81
OSCAR-llama x128	0.81	0.61	0.79	0.90	0.63	0.77
OSCAR-10L x16	0.83	0.64	0.80	0.91	0.65	0.79
OSCAR-10L x128	0.80	0.61	0.78	0.90	0.61	0.75
OSCAR-8L x16	0.82	0.64	0.80	0.91	0.64	0.79
OSCAR-8L x128	0.79	0.61	0.77	0.90	0.61	0.75

Table 6: LLM evaluations of mistral-7B OSCAR models with compression rates 16 and 128.

the exact recipe of the pretraining is not crucial for replicating our work.

¹¹[huggingface./datasets/HuggingFaceFW/fineweb](https://huggingface.co/datasets/HuggingFaceFW/fineweb)

Figure 13: LLM Evaluation Prompt

system: "You are an evaluation tool. Answer with one of 1: Correct, 0.5: Partially correct, 0: wrong.
user: "Here is a question, a golden answer, and an AI-generated answer. Can you judge whether the AI-generated answer is correct according to the question and golden answer? Simply answer with one of 1: correct, 0.5: partially correct, 0: wrong.
 Question: {question}. Golden answer: {answer}. Generated answer: {prediction}."

Figure 14: Main Prompt

system: You are a helpful assistant. Your task is to extract relevant information from provided documents and to answer questions as briefly as possible.
user: Background:
 $\{doc_1\}SEP\{doc_2\} \dots SEP\{doc_k\}$
 Question: {question}

Figure 15: 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}."

Hyperparameter	Value
Batch Size	128
LR generator	1×10^{-4}
LR llama compressor	1×10^{-4}
LR N-layers compressor	5×10^{-5} ^a
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
Llama compressor hidden dim	8096
Weight Decay	0.1
Warmup Ratio	0.05
Max Gradient Norm	1.0
Documents max tokens	128

Table 7: Fine-tuning Hyper-parameters.

^aInitial results with identical learning rates between the LoRA-trained decoder and fully fine-tuned N-layers compressor gave poor results: learning rates need to be differentiated between compressor and decoder in this case.

Model	ASQA	HotpotQA	NQ	TriviaQA	POPQA
OSCAR-llama	0.82	0.64	0.81	0.91	0.65
+ without pretraining	0.78	0.56	0.75	0.89	0.51

Table 8: Ablation on the pretraining for OSCAR-llama model.

J. Smaller compressors

Results shown in the main sections with small compressors mainly focus on using llama-1B as the compressor LLM. To gain further efficiency gains, we tested using smaller compressors: bert-base and modern-bert. Figure 16 shows results after pretraining and fine-tuning with different compressors. Llama-1B performs the best. Smaller compressors reach some level of accuracy which remains below the uncompressed model.

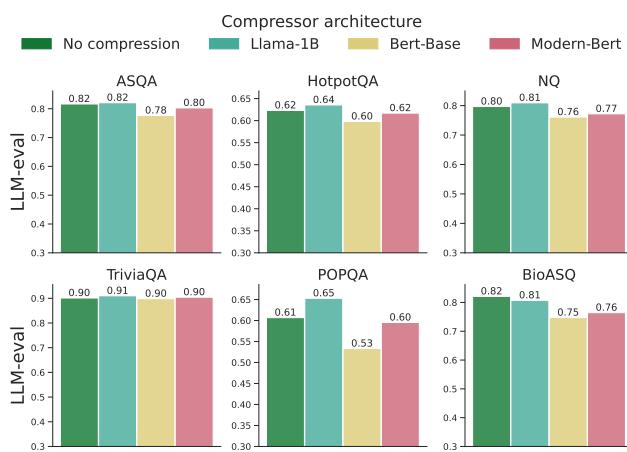


Figure 16: LLM evaluation of OSCAR models with different compressor architectures.