Eliciting In-context Retrieval and Reasoning for Long-context Large Language Models

Yifu Qiu[1*], **Varun Embar**[2], **Yizhe Zhang**[2],
Navdeep Jaitly[2], **Shay B. Cohen**[1], **Benjamin Han**[2]
[1] University of Edinburgh, [2] Apple
{yifu.qiu, scohen}@ed.ac.uk ,
{v_embar, yizhe_zhang, njaitly, ben.b.han}@apple.com

Abstract

Recent advancements in long-context language models (LCLMs) promise to transform Retrieval-Augmented Generation (RAG) by simplifying pipelines. With their expanded context windows, LCLMs can process *entire* knowledge bases and perform retrieval and reasoning directly – a capability we define as **I**n-**C**ontext **R**etrieval and **R**easoning (**ICR^2**). However, existing benchmarks like LOFT often overestimate LCLM performance by providing overly simplified contexts. To address this, we introduce ICR^2, a benchmark that evaluates LCLMs in more realistic scenarios by including confounding passages retrieved with strong retrievers. We then propose three methods to enhance LCLM performance: (1) *retrieve-then-generate fine-tuning*, (2) *retrieval-attention-probing*, which uses attention heads to filter and de-noise long contexts during decoding, and (3) *joint retrieval head training* alongside the generation head. Our evaluation of five well-known LCLMs on LOFT and ICR^2 demonstrates significant gains with our best approach applied to Mistral-7B: +17 and +15 points by Exact Match on LOFT, and +13 and +2 points on ICR^2, compared to vanilla RAG and supervised fine-tuning, respectively. It even outperforms GPT-4-Turbo on most tasks despite being a much smaller model. [1]

1 Introduction

The ability of large language models to process long contexts has significantly expanded their applicability across various domains, including book-level information retrieval (Ding et al., 2024; Jin et al., 2024), summarization (Kim et al., 2024; Saxena and Keller, 2024; Qiu et al., 2023), and question answering (Liu et al., 2024; Wang et al., 2024a). They also enable more complex tasks, such as agent trajectory modeling and planning (Zhao et al., 2024; Zhang et al., 2024b), video captioning (Xue et al., 2024; Zhang et al., 2023), and text-to-video generation (Wang et al., 2024b; Lin et al., 2023). Recent advancements in long-context language models (LCLMs) hold particular promise for reshaping the Retrieval-Augmented Generation (RAG) paradigm (Lee et al., 2024; Li et al., 2024). With their expanded context windows, LCLMs reduce reliance on complex pipelines required by context-length limitations and simplify knowledge updates by allowing context modifications. For instance, LCLMs can accommodate entire knowledge bases within the context windows, effectively serving as working memory for new queries.

Achieving this goal requires LCLMs to effectively retrieve and reason within their "contextual knowledge base," a capability we define as **I**n-**C**ontext **R**etrieval and **R**easoning (**ICR^2**). However, existing benchmarks often fail to accurately evaluate this capability. For example, Needle-in-a-Haystack (NIAH; Kamradt 2023) is a popular test to determine whether a model can retrieve a "needle" (a specific fact or statement) randomly inserted into a "haystack" (a corpus). Yet, the semantic discontinuity between the needle and the haystack can unintentionally reveal the needle's location, making the task overly simple. LOFT (Lee et al., 2024), the first large-scale benchmark for evaluating retrieval and reasoning within contextual knowledge bases, uses human-annotated relevant documents as the needle, and fills the haystack with randomly sampled documents from an external knowledge base. However, this random sampling results in a context with virtually no *confounding* information that is relevant but misleading, causing LOFT to significantly overestimate LCLM performance.

To bridge this evaluation gap, we introduce ICR^2, a novel and challenging benchmark designed to assess LCLMs under more realistic conditions. ICR^2 builds upon KILT (Petroni et al., 2021), a compre-

[*] Work done while the author was an intern at Apple.
[1] Our code and datasets are available at https://github.com/apple/ml-icr2.

Findings of the Association for Computational Linguistics: ACL 2025, pages 3176–3192
July 27 - August 1, 2025

hensive knowledge base sourced from Wikipedia. Unlike LOFT, which relies on random sampling, ICR^2 uses strong retrievers to select challenging confounding documents, creating a more difficult "haystack." Experimental results reveal that current LCLMs struggle on ICR^2, with exact match rates dropping by up to 51% compared to evaluations on LOFT. These findings underscore the significant challenges LCLMs face in accurately performing in-context retrieval in realistic scenarios.

We next explore improving LCLMs' in-context retrieval and reasoning capabilities. While RAG demonstrates strong results, it remains hindered by complex multi-stage pipelines. Encouraged by recent studies such as LLM2Vec (BehnamGhader et al., 2024; Ma et al., 2024) showing that LCLMs can be effectively adapted for accurate retrieval tasks, offering a more natural approach by enabling joint optimization of both retrieval and generation steps, we propose three approaches: (1) **Retrieve-then-generate fine-tuning**: Inspired by the distillation of step-by-step reasoning abilities (Shridhar et al., 2023; Hsieh et al., 2023), we train LCLMs to frame tasks as two-hop reasoning chains. In this formulation, LCLMs first retrieve relevant information from the context and then generate the final responses. (2) **Retrieval-attention probing**: At inference time, we probe attention heads activated for in-context retrieval (Wu et al., 2024) and use their top predictions to filter out confounders from lengthy contexts. (3) **Joint retrieval head training**: We introduce an architectural modification to equip LCLMs with a dedicated retrieval head, enabling joint optimization of retrieval and generation.

We conduct extensive experiments using five LCLMs on both LOFT and our ICR^2 benchmark. We compare our methods against baselines including Vanilla RAG, Closed-book, Oracle RAG, and supervised fine-tuning (SFT). Notably, our best approach, applied to Mistral-7B with a 32K token limit, has the best performance across the tasks. It outperforms Vanilla RAG and SFT baselines by an average of +17 and +15 points measured by Exact Match on LOFT, and by +13 and +2 points on ICR^2, respectively. The approach achieves performance comparable to the state-of-the-art GPT-4, despite using only 7B parameters. We also provide in-depth analyses of our approaches.

Our key contributions are as follows:

- We introduce ICR^2, a realistic and challenging benchmark for evaluating the in-context retrieval and reasoning capabilities of long-context language models (LCLMs), demonstrating that existing benchmarks overestimate model performance.
- We explore three novel methods to enhance LCLMs ranging from supervised fine-tuning, inference-time approach, to model architecture modification.
- Our best approach, applied to a small LCLM, tightens the performance gap with the Oracle RAG while beating the other baselines. It even matches GPT-4 on both in-domain (ICR^2) and out-of-domain benchmarks (LOFT), albeit with a much smaller model size.

2 Related Work

Long-context large language models (LCLMs) have garnered significant attention for their ability to process extended sequences. Models like Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020) introduced sparse attention mechanisms to efficiently handle long context. Scaling efforts, exemplified by GPT-4 (Achiam et al., 2023), underscore the importance of expanding context windows for the long-context tasks. Data engineering methods (Fu et al., 2024; Xiong et al., 2024; Jin et al., 2024), expanding positional encoding (Ding et al., 2024), and the parameter-efficient fine-tuning (Chen et al., 2024) have been effective.

LCLMs hold the promise in reshaping RAG (Achiam et al. 2023; Jiang et al. 2023; Yang et al. 2024; Abdin et al. 2024). By replacing static knowledge bases with the contextual one, this new paradigm simplifies the deployment by discarding intermediate components while enabling the direct updates on LCLM's knowledge (Lee et al., 2024).

Unlike the existing works, we systematically evaluate the use of LCLMs for scenarios where knowledge bases are directly placed as the context with a novel benchmark, ICR^2. ICR^2 is different with the popular long-context benchmark such as (Bai et al., 2024) by emphasizing in the contextual confounders. We also show that with targeted enhancements, small-scale LCLMs can achieve performance comparable to state-of-the-art models.

3 Are LCLMs Competent for RAG?

3.1 LLMs Are Sensitive to the Confounders

LOFT (Lee et al., 2024) introduces a Corpus-in-Context (CiC) approach for retrieval-augmented

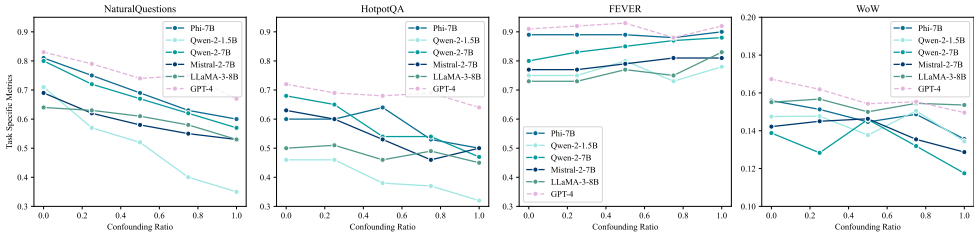

Figure 1: Task-specific performance of five LCLMs on 32K ICR^2 test sets with varying confounding ratios. For all tasks, a higher value of task-specific metric indicates a better performance.

generation, integrating a large-scale external knowledge base directly into the LLM's context. However, LOFT operates under the assumption that the context is free of *confounders*. In practice, a corpus often contains confounders – documents related to the query but potentially leading to incorrect answers. For example, given the query "*Who is the 44th U.S. President?*", the corpus might include documents about the other presidents, such as "*The 45th U.S. President is Donald Trump*", which could mislead the LLM. Consequently, LOFT's design reduces the complexities of using real-world corpora for RAG, potentially overestimating the performance of LCLMs.

We show that LCLMs are indeed sensitive to the confounders missed in LOFT. We construct multiple test sets with varying *confounding ratios* – $\{0, 25\%, 50\%, 75\%, 100\%\}$ – and evaluate the LCLMs under zero-shot settings. The confounding ratio p denotes the proportion of confounding context that are selected by retrievers while filtering out the gold provenance, with the remaining, i.e., $(100\% - p)$, randomly sampled from an external knowledge base. At $p = 0$, all confounding passages in the contextual knowledge base are randomly sampled, equivalent to the setup in LOFT (Lee et al., 2024). Conversely, at $p = 1$, all confounders are selected by the retrievers.

We evaluate five LCLMs with a context length of at least 32K tokens: Phi-3-7B (Abdin et al., 2024), Qwen-2-1.5B/7B (Yang et al., 2024), Mistral-003-7B (Jiang et al., 2023), LLaMA-3-Instruct-8B (Dubey et al., 2024) and GPT-4-Turbo (Achiam et al., 2023). Our findings in Figure 1, demonstrates that LCLM performance is highly sensitive to confounders, with performance generally degrading as the confounding ratio increases. This indicates that confounders in P^- obtained via retrievers, as in ICR^2, pose greater challenges for the models compared to those randomly sampled in LOFT. These retriever-selected passages are relevant to the queries but fail to lead to the correct answers,

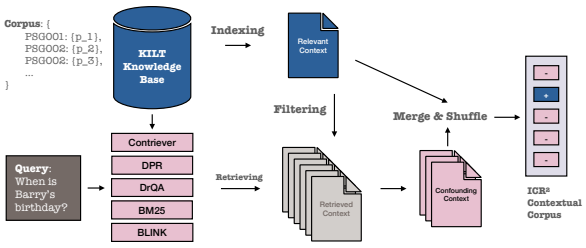

Figure 2: Construction pipeline for ICR^2. Based on KILT, we use five strong retrievers to retrieve confounders to serve as the "haystack", while the human-annotated contexts are used as the "needle".

making them particularly difficult for LCLMs to handle. Consequently, benchmarks like LOFT that overlook these "strong" confounders risk overestimating model performance, as confounders are prevalent in real-world scenarios.

3.2 ICR^2 Benchmark

To address this limitation, we propose an alternative benchmark, ICR^2, which leverages strong retrievers to identify and incorporate these confounders into the contextual corpus used in CiC, providing a more realistic and challenging evaluation framework. We choose KILT (Petroni et al., 2021), a comprehensive suite of benchmarks designed for knowledge-intensive NLP tasks, to be our external knowledge base (Figure 2). KILT covers tasks such as question answering (Kwiatkowski et al., 2019; Yang et al., 2018), fact verification (Thorne et al., 2018), and dialogue completion (Dinan et al., 2018), all paired with a single Wikipedia snapshot. Each KILT instance $\langle q, a, P^+ \rangle$ consists of a query q, the reference answer a, and its provenances $P^+ = \{p_1, \ldots, p_m\}$, which is a set of relevant Wikipedia pages and specific locations within them that support the answer.

Building on KILT, each ICR^2 instance is represented as $\langle q, a, C \rangle$, where q and a is a query-answer pair from KILT, and C is the contextual knowledge base required to answer the query. At test time, a standardized prompt template, `rag_prompt`$[C]$, is applied to the context, and the

Query:
Where are the giant redwoods located in California?

Reference Answer:
Humboldt County

Contextual Knowledge Base:
`ID: PSG001`
Title: 2010–11 California Golden Bears Men's Basketball Team
Context: The 2010–11 California Golden Bears men's basketball team represented the University of California, Berkeley in the 2010–11 NCAA Division I men's basketball season.

`ID: PSG002`
Title: Redwood National and State Parks
Context: Scenes set on the forest moon Endor in Star Wars were filmed in the Tall Trees Redwood Grove in the northern part of Humboldt County, though the majority of filming was in private and public forests near the town of Smith River ...

... [90 context are omitted for the simplicity] ...

`ID: PSG100`
Title: Malibu Grand Prix
Context: ... and San Antonio, Texas. Palace operates additional locations in Los Angeles, California and Dallas, Texas. The Redwood City, CA location closed on August 18, 2013 and the San Antonio location closed on September 7, 2015.

Table 1: An example ICR^2 instance. We show the query with the reference answer, and the constructed contextual knowledge base. We highlight the true provenance in blue and the confounding passages in red.

result is sent to an LCLM to generate the final answer for evaluation. To construct C, we first include all provenances P^+ from KILT as the positive passages. For confounder P^-, unlike LOFT's random sampling approach, we run five strong retrievers on queries formed by concatenating q with a to select Top-K results as the confounding passages to be included in C.

We show the pipeline for ICR^2 creation in Figure 2. To eliminate bias in the retrievers' preferences during the construction process, we pool the Top-K predictions from all retrievers and uniformly sample passages from top to bottom until reaching the maximum allowable size for the contextual knowledge base. The retrievers used include Contriever (Izacard et al.), DPR (Karpukhin et al., 2020), DrQA (Chen et al., 2017), BM25 (Robertson and Jones, 1976; Robertson et al., 2009), and BLINK (Wu et al., 2020). For passage-level retrievers such as Contriever, BM25, and DPR, as input we chunk all documents in KILT's knowledge base and exclude passages overlapping with any annotated provenance in P^+. We also set $K = 200$ for the 32K version of ICR^2. For document-level retrievers like DrQA and BLINK, we set $K = 20$, and chunk retrieved documents using the same process as above. For all retrievers, we exclude retrieved passages overlapping with P^+ or containing exact substring matches with a, and the remaining passages are collected to form the confounding provenance set, P^-. The contextual corpus C is then constructed by concatenating P^+ with P^- up to the maximum length, i.e., $C = [P^+; P^-]$. Finally, we shuffle all items in C to remove position bias.

Table 7 compares statistics between LOFT and our benchmark ICR^2. An example of an ICR^2 instance is provided in Table 1, and the `rag_prompt` template used is detailed in Table 8 in Appendix B.

3.3 State of LCLMs Performance on ICR^2

We evaluate five LCLMs that support 32K or longer context length using LOFT (Lee et al., 2024) and ICR^2 benchmarks, with each one tested with input up to 32K tokens. We include three baselines for comparison: (1) **Vanilla RAG**, where the CiC prompt from Lee et al. (2024) is used; (2) **Closed-book**, where the entire context (C) is removed to evaluate LCLM performance based solely on parametric knowledge; and (3) **Oracle RAG**, which includes only the ground-truth relevant contexts Z^* in C to estimate an upper-bound performance.

Table 2 presents our results. Overall, we observe that the Vanilla RAG significantly outperforms the closed-book baseline for most models, except for Qwen-2-1.5B on the WoW dataset. This trend persists even with the state-of-the-art GPT-4-Turbo, suggesting that external knowledge remains crucial for knowledge-intensive tasks despite the large parametric capacity of LCLMs. Furthermore, RAG performance on ICR^2 lags notably behind that on LOFT, reflecting ICR^2's greater complexity due to its more realistic context construction. Lastly, the large performance gap between RAG and Oracle setups on both benchmarks underscores the impact of confounding information, which hampers accurate retrieval and response generation in LCLMs.

4 Eliciting ICR^2 for LLMs

4.1 Retrieve-then-generate Fine-tuning

Compared to the standard supervised fine-tuning where a model is asked to generate a **D**irect **A**nswer (**DA**) to a given query, our first proposal, *retrieve-then-generate*, is a two-step process: the model first retrieves relevant passages from context and then generates the final answer based on the context and retrieved passages, all in one decoding pass.

	LOFT				ICR^2					All
	NQ	HotpotQA	MUSIQUE	Avg	NQ	HotpotQA	FEVER	WoW	Avg	
Closed-book										
Qwen-2-1.5B-32K	0.21	0.22	0.03	0.15	0.29	0.24	0.62	0.15	0.32	0.25
Qwen-2-7B-32K	0.31	0.30	0.07	0.23	0.35	0.30	0.71	0.15	0.38	0.31
Mistral-2-7B-32K	0.46	0.38	0.05	0.30	0.49	0.38	0.73	0.14	0.43	0.38
Phi-3-7B-128K	0.39	0.35	0.09	0.28	0.46	0.32	0.73	0.13	0.41	0.35
LLaMA-3-instruct-8B	0.41	0.19	0.01	0.20	0.44	0.19	0.53	0.14	0.32	0.27
GPT-4-turbo	**0.58**	**0.57**	**0.28**	**0.48**	**0.61**	**0.50**	**0.85**	**0.16**	**0.53**	**0.51**
Vanilla RAG										
Qwen-2-1.5B-32K	0.61	0.43	0.12	0.39	0.35	0.32	0.78	0.12	0.39	0.39
Qwen-2-7B-32K	0.79	0.61	0.29	0.56	0.57	0.47	0.88	0.13	0.51	0.53
Mistral-2-7B-32K	0.64	0.62	0.27	0.51	0.53	0.50	0.81	0.13	0.49	0.50
Phi-3-7B-128K	0.76	0.68	0.41	0.62	0.60	0.50	0.90	0.14	0.53	0.57
LLaMA-3-instruct-8B	0.45	0.56	0.16	0.39	0.53	0.45	0.83	**0.15**	0.49	0.45
GPT-4-turbo	**0.85**	**0.78**	**0.51**	**0.71**	**0.67**	**0.64**	**0.92**	**0.15**	**0.59**	**0.65**
Oracle RAG										
Qwen-2-1.5B-32K	0.80	0.76	0.44	0.67	0.77	0.65	0.86	0.15	0.61	0.63
Qwen-2-7B-32K	0.88	0.81	0.56	0.75	0.83	0.80	0.91	0.17	0.68	0.71
Mistral-2-7B-32K	0.89	0.81	0.41	0.70	0.83	0.81	**0.94**	0.18	**0.69**	0.70
Phi-3-7B-128K	**0.90**	0.85	0.63	0.79	**0.87**	0.81	0.91	0.14	0.68	0.73
LLaMA-3-instruct-8B	0.81	0.77	0.60	0.73	0.80	0.74	0.86	**0.19**	0.65	0.68
GPT-4-turbo	0.88	**0.87**	**0.72**	0.82	0.79	**0.82**	**0.94**	0.18	0.68	**0.74**

Table 2: Performance evaluation of six LCLMs on LOFT and ICR^2 benchmarks. All models benefit from the Vanilla RAG approach, but a gap remains between vanilla and oracle RAG performance.

Formally, we train an LCLM to optimize,

$$p(y \mid q, c) = \sum_{z_i \in Z} p(y \mid q, c, z_i) p(z_i \mid q, c), \quad (1)$$

where q, y is the query and target, respectively, c is the contextual knowledge base, and Z is the collection of all relevant passages necessary for answering q. This objective can be easily integrated into the next-token prediction task trained with the maximum likelihood estimation, just by sequentially executing the retrieval and generation in a single forward pass. The overall loss is given as

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left(\log p(Z_i^* | q_i, c_i) + \log p(y_i^* | Z_i^*, q_i, c_i) \right), \quad (2)$$

where N is the number of training samples, and $Z_i^* = \{z_{i,1}^*, z_{i,2}^*, \ldots, z_{i,|Z_i^*|}^*\}$ is the collection of all relevant passages for the query q_i and contextual knowledge base c_i.

We use two variants of retrieve-then-generate fine-tuning, both using the special tokens `<RETRIEVAL>` and `</RETRIEVAL>` to delineate the retrieval step. In the first, **R**etrieve-**T**hen-**A**nswer (**RTA**), the model copies relevant passages. In the second, **C**ite-**C**ontext-**ID** (**CCI**), the model generates only the IDs of the relevant passages. Figure 8 shows the templates for all variations.

4.2 Retrieval Attention Probing

Our second proposal, **R**etrieval **A**ttention **P**robing (**RAP**), is an inference-time approach compatible with LCLMs without requiring re-training. Building on Wu et al.'s findings that specific attention heads are highly active during retrieval tasks (e.g., NIAH), RAP utilizes these retrieval-focused attention heads for context filtering before generating responses. For each attention head h and query q, we track the Top-M attention scores $A_h^M(q)$. $C_h^M(q)$, representing the M passages corresponding to $A_h^M(q)$, are then selected, and the *hit rate* for head h is calculated as follows:

$$\text{HitRate}_h = \frac{1}{N} \sum_{i=1}^{N} \frac{|C_h^M(q) \cap Z^*(q)|}{|Z^*(q)|}, \quad (3)$$

where N is the number of validation samples, and $Z^*(q)$ is the set of all relevant contexts given query q. Finally we select Q heads with the top hit rates to be the retrieval heads as follows,

$$\mathcal{H}_{\text{ret}} = \underset{\mathcal{H} \subseteq \{1,\ldots,H\}, |\mathcal{H}_{\text{ret}}| = Q}{\arg\max} \sum_{h \in \mathcal{H}} \text{HitRate}_h, \quad (4)$$

where H is the number of attention heads, and $\mathcal{H}_{\text{ret}}$ is the set of Q attention heads with the top hit rates selected as the retrieval heads for context filtering.

Note that our definition of retrieval heads differs from (Wu et al., 2024) in two key ways: (1) Instead of focusing solely on the Top-1 passages,

we allow each head to retain the Top-M, which is particularly important for multi-hop reasoning tasks requiring multiple passages for reasoning. (2) We evaluate attention heads using the retrieval hit rate, a more direct metric for our downstream tasks.

During inference, we union all passages selected by all retrieval heads, $\mathcal{H}_{\text{ret}}$,

$$C^* = \bigcup_{h \in \mathcal{H}_{\text{ret}}} \underset{c \in C}{\text{Top-}M}\ \alpha_h(c), \tag{5}$$

where $\alpha_h(c)$ is the attention score of head h on passage c, and C^* is the set of the Top-M selected passages from all heads $\mathcal{H}_{\text{ret}}$. We use C^* to form a new CiC-style prompt and proceed to generating the final response. Since $|C^*| \ll |C|$, the final decoding is actually performed on a filtered contextual knowledge base with a much smaller length.

4.3 Joint Retrieval Head Training

Our final proposal introduces a dedicated retrieval head to the LCLM model architecture. During inference, the model first uses the head to identify relevant passages, after which the generation head decodes a response conditioned on the retrieved content. During training, the retrieval and generation heads are jointly optimized using the Gumbel-TopK trick (Kool et al., 2019), which mitigates the non-differentiability of the retrieval process.

Figure 9 illustrates the modified model architecture. The retrieval head generates a binary mask, $M \in \{0,1\}^{|C|}$, indicating which passages to select (1) or ignore (0). The selected passages are then passed to the generation head for response generation. The retrieval head consists of: (1) encoders for the query and passage, using the LCLM's final hidden states, and (2) a scoring layer that computes relevance scores by concatenating their encoded vectors. The top K passages are then selected for the generation head to produce a response.

More specifically, let q denote a query and $C = \{c_1, c_2, \ldots, c_n\}$ be the set of n passages. For each pair (q, c_i), two single-layer encoders pool the final hidden states from an LCLM and generate a query vector, $\mathbf{h}_Q$, and a passage vector, $\mathbf{h}_{c_i}$:

$$\mathbf{h}_q = \text{enc}_q(q); \quad \mathbf{h}_{c_i} = \text{enc}_c(c_i),$$

Each pair $(\mathbf{h}_q, \mathbf{h}_{c_i})$ is then concatenated to form an input vector $\mathbf{v}_i = [\mathbf{h}_q; \mathbf{h}_{c_i}]$. A single scoring layer then predicts a scalar relevance score s_i for each pair, essentially computing $s_i = f(\mathbf{v}_i)$ where $f(\cdot)$ is our scoring function. With the set of scores $S = \{\ldots s_i \ldots\}$, we can finally identify the indices T of the selected passages as the ones that receive the top K scores,

$$T = \text{TopK}(S, K).$$

We then set the binary mask M for filtering the original context C down to C^*, which is then used to prompt the LCLM to produce the final response.

5 Experiment

Benchmarks. We use LOFT (Lee et al., 2024) and our ICR^2 benchmarks to evaluate LCLMs' in-context retrieval and reasoning capabilities. LOFT tests retrieval, single- and multi-hop question answering, and reasoning using NaturalQuestions (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), and MuSiQue (Trivedi et al., 2022). ICR^2 uses NaturalQuestions and HotpotQA, and additionally includes FEVER (Thorne et al., 2018) for fact verification and WoW (Dinan et al., 2018) for dialogue completion. Similar to LOFT, we report average scores across 100 test cases per task using the 32K context length versions, which is the maximum supported by all tested LCLMs.

Metrics. For the question answering and fact verification tasks, we use the exact match in (Lee et al., 2024; Adlakha et al., 2024). We use ROUGE (Lin, 2004) to assess on dialogue completion.

Training Details. We use ICR^2's training set to fine-tune all models. Specifically, we randomly sample 7500, 7500, 5000, and 5000 instances for NaturalQuestions, HotpotQA, FEVER, and WoW, respectively. To verify the effectiveness of our proposed methods, we focus our experiments on Mistral-Instruct-7B model (Jiang et al., 2023).

Baselines. We compare our approaches with the baselines in Sec. 3.3: Vanilla RAG, Closed-book, and Oracle RAG. We include the traditional ranking pipeline where we use TinyBERT (Jiao et al., 2020) and MiniLM (Wang et al., 2020) fine-tuned on MS Marco passage retrieval corpus (Bajaj et al., 2016) to select the top-relevant passages for LCLMs. We report their retrieval performance in Appendix D. We include the direct-answer SFT (SFT-DA) which follows Zhang et al. by concatenating the confounders with gold documents for SFT, and our methods in Sec. 4 include two retrieve-then-generate SFT variants — Retrieve-then-Answer (SFT-RTA), and Cite-Context-ID (SFT-CCI) — and the joint retrieval head training (RetHead) and retrieval attention probing (RAP).

Models & Methods	LOFT			ICR^2			
	NQ	HotpotQA	MUSIQUE	NQ	HotpotQA	FEVER	WoW
GPT-4 w/							
Close-book	0.58	0.57	0.28	0.61	0.50	0.85	0.16
Vanilla RAG	0.85	0.78	0.51	0.67	0.64	0.92	0.15
Oracle	0.88	0.87	0.72	0.79	0.82	0.94	0.18
Mistral-7B w/							
Close-book	0.46	0.38	0.05	0.49	0.38	0.73	0.14
Vanilla RAG	0.64	0.62	0.27	0.53	0.50	0.81	0.13
RAG w/ RTA Prompting	0.60	0.70	0.27	0.54	0.51	0.83	0.15
Re-ranking Strategy							
w/ TinyBERT ($k = 8$)	0.88	0.78	0.29	0.52	0.47	0.88	0.12
w/ MiniLM ($k = 8$)	0.83	0.77	0.27	0.51	0.46	0.86	0.13
w/ TinyBERT ($k = 32$)	**0.87**	**0.87**	0.33	0.62	0.51	0.91	0.13
w/ MiniLM ($k = 32$)	0.84	0.84	0.39	0.61	0.47	**0.92**	0.12
Oracle RAG	0.89	0.81	0.41	0.83	0.81	0.94	0.18
Supervised Fine-tuning							
SFT-Direct Answer	0.70	0.65	0.25	0.59	0.70	0.90	0.22
SFT-Retrieve-then-Answer	0.74	0.69	0.33	0.60	0.67	0.91	0.22
SFT-Cite-Context-ID	0.76	0.54	0.35	0.63	0.63	0.89	0.21
Joint Retrieval Head Training							
RetHead $w/\mathcal{L}_{gen} + \mathcal{L}_{ret}$	0.15	0.13	0.07	0.48	0.54	0.9	0.21
RetHead $w/\mathcal{L}_{gen}$	-	-	-	0.28	0.25	0.82	0.18
RetHead $w/\mathcal{L}_{ret}$	-	-	-	0.39	0.44	0.82	0.13
Retrieval-Attention Probing							
SFT-DA w/ RAP	0.78	0.76	**0.47**	**0.64**	0.67	0.89	0.21
SFT-RTA w/ RAP	0.85	0.79	0.39	0.63	**0.71**	**0.92**	**0.23**

Table 3: Main results on LOFT (Lee et al., 2024) and ICR^2 for our methods applied on Mistral-2-7B-Instruct (Jiang et al., 2023). We also report the GPT-4 performance in the top panel. We highlight the improved and worsen performances compared with Vanilla RAG, and **bold** the best method based on Mistral-7B, except the Oracle.

5.1 Main Results

As shown in Table 3, all SFT variants outperform the Vanilla RAG on both benchmarks, indicating that LCLMs struggle to effectively leverage context as a knowledge base for RAG tasks. Furthermore, the gap between SFT-DA and the Oracle RAG highlights that supervised fine-tuning alone is insufficient to achieve optimal results.

We first apply RTA-style generation as a prompt-only method (RTA Prompting). The resulting performance gains, though modest, demonstrate the potential of decoupling the generation process into separate retrieval and generation stages. Among the SFT variants, SFT-RTA in general outperforms the others with an average improvement of 2%. Specifically on LOFT benchmark, both SFT-RTA and SFT-CCI outperform SFT-DA with 6% and 2% improvement on average, respectively, indicating the retrieve-then-generate strategy helps. On ICR^2, however, all SFT variants perform the same. This demonstrates that ICR^2 is a more discriminative benchmark than LOFT.

We apply RAP to all SFT models. On average, RAP enhances the models significantly, with SFT-DA + RAP improving by 6% and SFT-RTA + RAP by 8%. The best-performing approach, SFT-RTA + RAP, achieves notable gains on the challenging ICR^2 benchmark, with improvements of 3%, 4%, 1%, and 1% on NaturalQuestions, HotpotQA, FEVER, and WoW, respectively. It also achieves top performance on 5 out of the 7 tasks, demonstrating its superiority.Remarkably, it achieves comparable performance with the state-of-the-art GPT-4-Turbo on LOFT and ICR^2 while using a much smaller model. Finally, RAP decoding is more effective for SFT models than with the original model, as SFT better activates retrieval-specific attention heads for the approach (see Sec. 6.3). To verify the generalization of methods across various LLMs, we conduct experiments with LLaMA-3-Instruct (Dubey et al., 2024) in Appendix C: we observe the similar improvements for retrieve-then-generate training, and RAP decoding.

Compared to the traditional pipeline, LCLMs can effectively unify in-context retrieval and generation, particularly on challenging benchmarks such as ICR^2. This highlights the advantage of an end-to-end approach, wherein LCLMs contextually select retrieved items based on both the query and the evolving generation.

For the joint retrieval head training (RetHead), we performed experiments on training only the generation head ($w/\mathcal{L}_{gen}$), only the retreieval head ($w/\mathcal{L}_{ret}$), and both ($w/\mathcal{L}_{gen} + \mathcal{L}_{ret}$). The last variant achieves the best performance, outperforming the Vanilla RAG baseline on ICR^2. However, it is

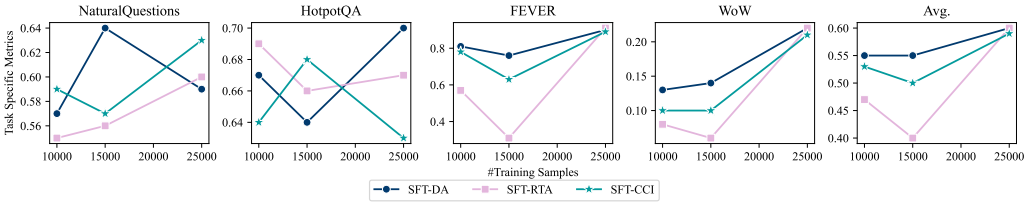

Figure 3: Scaling the number of training samples in ICR^2's training set.

Methods	LOFT			ICR^2			
	NQ	HPQA	MUSI.	NQ	HPQA	FEV.	WoW
SFT-RTA	0.81	0.83	0.37	0.58	0.78	0.64	0.43
SFT-CCI	0.65	0.61	0.26	0.63	0.73	**0.69**	0.49
RetHead							
$w/\mathcal{L}_{gen} + \mathcal{L}_{ret}$	0.01	0.01	0.01	0.36	0.44	0.55	0.51
$w/\mathcal{L}_{gen}$	-	-	-	0.00	0.00	0.00	0.02
$w/\mathcal{L}_{ret}$	-	-	-	0.33	0.49	0.51	0.49
RAP Decoding							
Vanilla RAG	0.19	0.25	0.10	0.36	0.13	0.00	0.00
SFT-DA	**0.95**	0.70	0.27	**0.75**	0.67	0.60	**0.69**
SFT-RTA	**0.95**	**0.96**	**0.39**	0.61	**0.85**	0.60	0.41

Table 4: Retrieval performance measured by recall rate for various methods using Mistral-2-7B.

not comparable with the SFT variants.

5.2 In-Context Retrieval Performance

Table 4 reports the *recall rates* for all methods, meaning how likely the models retrieve the relevant provenances. For the SFT variants, we evaluate the retrieval predictions produced in the retrieval phase. For the joint retrieval head training approach (RetHead), we analyze the predictions from the retrieval head. For RAP decoding, we assess the passages identified by the selected attention heads.

We find a strong correlation between a model's recall rate and its downstream task performance, highlighting the importance of in-context retrieval ability for LCLMs. Our best approach, SFT-RTA + RAP, achieves the highest recall. In contrast, Vanilla RAG + RAP exhibits poor retrieval performance, consistent with its limited improvement on the downstream tasks. The near-random retrieval performance of joint training of retrieval head in LOFT explains its failure of generalization.

6 Discussion

6.1 Scaling the Supervised Fine-tuning

We are also interested in how the performance of the proposed SFT variants scale with the training set size. We train the three variants, SFT-DA, SFT-RTA, and SFT-CCI, with the same 10K, 15K and 25K examples from ICR^2's training set, and report their performance on each task in the ICR^2 benchmark, as shown in Fig 3. We observe that an increased training set size in general leads to an improved model performance. In particular, a smaller amount of training data fares worse with the retrieve-then-generate approaches, as they are by nature more challenging to learn compared to the SFT-DA approach, where answer is directly generated without an explicit retrieval step.

ICR^2	SFT-RTA		Blocking Context	
	Metrics	Recall	Metrics	Recall
NQ	0.6	0.58	0.61 (↑ .01)	0.59 (↑ .01)
HPQA	0.67	0.78	0.68 (↑ .01)	0.78 (= .00)
FEVER	0.91	0.64	0.91 (= .00)	0.66 (↑ .02)
WoW	0.22	0.43	0.21 (↓ .01)	0.46 (↑ .03)
Avg	0.6	0.61	0.6 (= .00)	0.62 (↑ .01)

Table 5: SFT-RTA's performance before and after blocking the contextual knowledge base with attention mask.

6.2 Blocking Context Attention in Retrieve-then-generate Model

To verify if models actually learn to generate the final responses only from the retrieval predictions in our retrieve-then-generate methods (Sec. 4.1), additional experiments were performed where we block attention to be paid onto the context beyond the retrieval predictions at the generation step. Table 5 reports the impact of the blocking on the SFT-RTA variant. We observe the performance essentially stays the same, indicating the model indeed learned to generate largely on the retrieved passages, ignoring the original context.

6.3 Effect on Attention Heads of Retrieve-then-Generate Fine-tuning

To understand if fine-tuning sharpens attention heads' focus on relevant passages, we compare the hit rates (Sec. 4.2) achieved by the attention heads between Vanilla RAG and all of our SFT variants (Sec. 4.1), and the results are shown in Figure 4.

Similar to (Wu et al., 2024), we find that a small group of attention heads can obtain higher hit rates than the others. However, unlike Vanilla RAG, SFT methods produce more retrieval-focused attention heads, and achieve higher peak hit rates. This demonstrates the effectiveness of our SFT methods and our curated ICR^2 training set in enhancing LCLMs' performance of in-context retrieval.

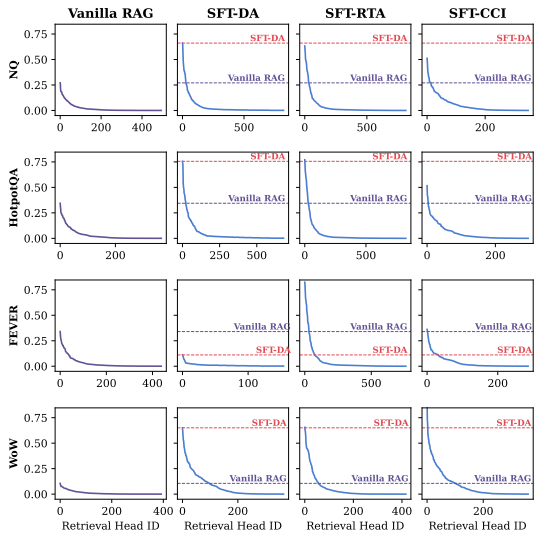

Figure 4: Attention heads with above-zero hit rates. SFT produces more retrieval attention heads. Retrieve-then-generate training activates a higher peak of hit rate.

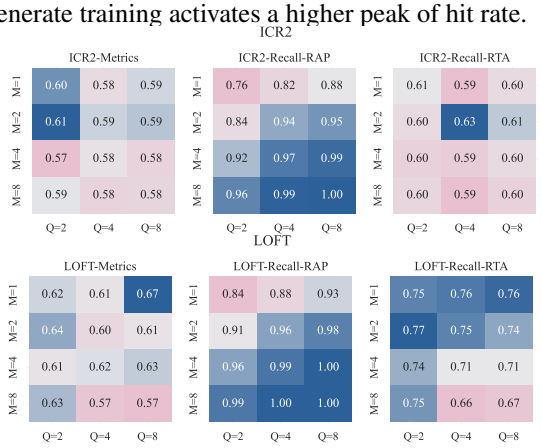

Figure 5: Effect of adjusting Q (number of attention heads for retrieval) and M (number of passages to select) when applying SFT-RTA + RAP. Left column: average model performance measured by the tasks-specific metrics. Middle column: attention heads' retrieval measured by the recall rate. Right column: RTA's retrieval measured by the exact match.

Among the three SFT variants, we find that SFT-RTA in general achieves higher peak hit rates and activates more attention heads for retrieval. In particular, SFT-CCI is not as effective in recruiting as many attention heads, possibly because context IDs themselves are not informative enough for the model to learn the retrieval task well. We also note that SFT-DA fares a lot worse on FEVER, possibly due to the lack of chain-of-thought style of assist.

6.4 Retrieval Attention Probing

Our inference-time method RAP (Sec. 4.2) uses two hyperparameters: Q is the number of the attention heads we recruit for retrieval, and M is the number of passages each head retrieves. In this section, we apply different value settings when deploying the SFT-RTA + RAP combined approach to explore their effect on model performance.

Benchmark	SFT-DA	SFT-DA + RAP
NQ	4.23	4.99 (↑ .76)
HotpotQA	4.19	4.81 (↑ .62)
MUSIQUE	4.24	5.05 (↑ .81)

Table 6: Latency for SFT-DA with and without the RAP decoding. Time is measured in seconds per query.

Figure 5 shows the high-level results on both LOFT and ICR^2 (Appendix E has more details). As expected, we observe in the middle column that an increasing M or Q will increase the recall rate as the resulting larger pools of selected passages will more likely include the relevant ones. We also find that increasing M as opposed to Q is more effective in improving the recall rate as certain task such as HotpotQA requires multi-hop retrieval. However, the increased recall rate does not always translate to a better performance (left column) or RTA's recall rate , as a higher M or Q may also introduce more confounders. This suggests the further reduction in confounding effects is still an opening future work.

6.5 Decoding Speed

Our final analysis is on the efficiency of RAP. Based on the SFT-DA variant, we report its latency with and without the RAP enhancement in Table 6. We find that the RAP decoding does not increase latency significantly, despite it adds one additional decoding step to the base method. This can be attributed to the much shorter context retrieved by the attention heads, thus avoiding expensive long-context computation as in the baseline. This increase can be further reduced with approaches such as KV caching, which we leave for future work.

7 Conclusion

In this paper, we introduce ICR^2, a new benchmark designed as a more discriminative benchmark for evaluating LCLMs in in-context retrieval and reasoning. Our findings highlight the limitations for the current models. We propose three methods —retrieve-then-generate fine-tuning, retrieval attention probing, and joint retrieval head training — to enhance models, achieving the results comparable to GPT-4 with a smaller model footprint.

Ethical Considerations

We do not expect any ethical concerns to be raised with respect to this work.

Limitations

We acknowledge several limitations in this work. First, most experiments were conducted with a context length of 32K tokens. Our findings indicate that while many LCLMs claim to support longer contexts, their performance on tasks with 32K tokens remains suboptimal. Future work could focus on extending ICR^2 and the proposed approaches to effectively support scenarios with longer context lengths.

Second, while Joint Retrieval Head Training demonstrates improved performance compared to Vanilla RAG, it still falls short of the performance achieved by the SFT variant. Future research could explore improved architectural designs to better integrate the supervision signals from both the retrieval and generation tasks.

Finally, our evaluation primarily utilizes the Mistral-7B model due to computational constraints. Extending the proposed methods to other LCLMs would provide a broader assessment of their generalization capabilities and effectiveness across different model architectures.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Vaibhav Adlakha, Parishad BehnamGhader, Xing Han Lu, Nicholas Meade, and Siva Reddy. 2024. Evaluating correctness and faithfulness of instruction-following models for question answering. *Transactions of the Association for Computational Linguistics*, 12:775–793.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations*.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. In *Forty-first International Conference on Machine Learning*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024. Data engineering for scaling language models to 128k context. In *Forty-first International Conference on Machine Learning*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

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*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. Llm maybe longlm: Self-extend llm context window without tuning. *CoRR*.

Greg Kamradt. 2023. Needle in a haystack - pressure testing llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

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

Yekyung Kim, Yapei Chang, Marzena Karpinska, Aparna Garimella, Varun Manjunatha, Kyle Lo, Tanya Goyal, and Mohit Iyyer. 2024. Fables: Evaluating faithfulness and content selection in book-length summarization. *arXiv preprint arXiv:2404.01261*.

Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, pages 3499–3508. PMLR.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan, Sébastien MR Arnold, Vincent Perot, Siddharth Dalmia, et al. 2024. Can long-context language models subsume retrieval, rag, sql, and more? *arXiv preprint arXiv:2406.13121*.

Huayang Li, Pat Verga, Priyanka Sen, Bowen Yang, Vijay Viswanathan, Patrick Lewis, Taro Watanabe, and Yixuan Su. 2024. Alr: A retrieve-then-reason framework for long-context question answering. *arXiv preprint arXiv:2410.03227*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Han Lin, Abhay Zala, Jaemin Cho, and Mohit Bansal. 2023. Videodirectorgpt: Consistent multi-scene video generation via llm-guided planning. *arXiv preprint arXiv:2309.15091*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2421–2425.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Yifu Qiu, Yftah Ziser, Anna Korhonen, Edoardo Ponti, and Shay Cohen. 2023. Detecting and mitigating hallucinations in multilingual summarisation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8914–8932, Singapore. Association for Computational Linguistics.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Stephen E Robertson and K Sparck Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146.

Rohit Saxena and Frank Keller. 2024. MovieSum: An abstractive summarization dataset for movie screenplays. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4043–4050, Bangkok, Thailand. Association for Computational Linguistics.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Minzheng Wang, Longze Chen, Fu Cheng, Shengyi Liao, Xinghua Zhang, Bingli Wu, Haiyang Yu, Nan Xu, Lei Zhang, Run Luo, Yunshui Li, Min Yang, Fei Huang, and Yongbin Li. 2024a. Leave no document behind: Benchmarking long-context LLMs with extended multi-doc QA. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5627–5646, Miami, Florida, USA. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788.

Yuqing Wang, Tianwei Xiong, Daquan Zhou, Zhijie Lin, Yang Zhao, Bingyi Kang, Jiashi Feng, and Xihui Liu. 2024b. Loong: Generating minute-level long videos with autoregressive language models. *arXiv preprint arXiv:2410.02757*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Zero-shot entity linking with dense entity retrieval. In *EMNLP*.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval head mechanistically explains long-context factuality. *arXiv preprint arXiv:2404.15574*.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. 2024. Effective long-context scaling of foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4643–4663.

Fuzhao Xue, Yukang Chen, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, et al. 2024. Longvila: Scaling long-context visual language models for long videos. *arXiv preprint arXiv:2408.10188*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

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

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Ce Zhang, Taixi Lu, Md Mohaiminul Islam, Ziyang Wang, Shoubin Yu, Mohit Bansal, and Gedas Bertasius. 2023. A simple llm framework for long-range video question-answering. *arXiv preprint arXiv:2312.17235*.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024a. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arik. 2024b. Chain of agents: Large language models collaborating on long-context tasks. *arXiv preprint arXiv:2406.02818*.

Jun Zhao, Can Zu, Hao Xu, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Longagent: Scaling language models to 128k context through multi-agent collaboration. *arXiv preprint arXiv:2402.11550*.

A Statistics for LOFT and ICR^2

	Task	#CTX	#Tokens	#Prov
LOFT	**NQ**	215	28,911	1
	HPQA	276	28,924	1.98
	MUSIQ.	207	28,806	1.67
ICR^2	**NQ**	202	20,441	1
	HPQA	202	20,818	2.09
	FEVER	202	21,136	1
	WoW	202	21,068	1

Table 7: Statistics for LOFT and our dataset ICR^2: #CTX is to the average number of passages per query in the contextual knowledge base, #Tokens is the average length of the CiC prompt, and #Prov. is the average number of provenance (positive passages) per query.

We present the comparison in dataset statistics between LOFT and ICR^2 in Table 7.

B Prompt Template for the Retrieval-augmented Generation

We show the Corpus-in-Context (CiC; Lee et al. 2024) prompt template used in our experiments in Table 8.

C Generalization to Other LCLMs

Table 9 presents the performance of the LLaMA-3-Instruct-8B model across two benchmarks—NQ-LOFT and HPQA-LOFT—evaluated under LOFT and ICR^2. The results highlight several key trends. First, the baseline Closebook performs poorly, particularly on the HPQA dataset, underscoring the need for external information. Incorporating retrieval via RAG offers significant improvements, especially on HPQA, where performance increases from 0.19 to 0.56. The Oracle setting represents the upper bound of retrieval quality, and shows the strongest performance overall.

We observe that adding a reranker such as TinyBERT to RAG further improves results under the LOFT setup, reaching 0.89 on both NQ and HPQA—approaching Oracle-level performance. Among supervised fine-tuning methods, SFT-DA and SFT-RTA provide moderate improvements over vanilla RAG. Notably, integrating our proposed RAP mechanism (e.g., in SFT-DA w/ RAP) leads to consistent gains, especially under ICR^2. For example, SFT-DA w/ RAP achieves the best NQ score (0.61), validating the effectiveness of RAP in enhancing decision quality during retrieval and answer generation.

D Retrieval Performance for Re-ranking Strategies

Table 10 presents a comprehensive evaluation of different re-ranking strategies applied to retrieval-augmented generation (RAG) across two benchmark suites: LOFT (comprising NQ, HPQA, and MUSI.) and ICR^2(including NQ, HPQA, FEV, and WoW). We compare baseline RAG without re-ranking, re-ranking using lightweight models (TinyBERT and MiniLM) at varying retrieval depths $k \in \{8, 32, 50\}$, and our best-performing variant (SFT-RTA w/ RAP). Results show consistent improvements with increased k, and MiniLM-based re-rankers often slightly outperform TinyBERT counterparts. Notably, the SFT-RTA w/ RAP model yields the highest performance across most datasets, particularly excelling in more challenging ICR^{2}domains. These findings suggest the strong retrieval capabilities of our SFT-RTA approach compared to the traditional re-ranking strategies.

E Detailed Results for RAP Decoding

We show the detailed results for each task of LOFT and ICR^2 in Figure 6 and 7.

F Details of Experiment Setup

F.1 Training Details

For all models, we set the base value of the rotary position embedding to $1e6$ following Su et al. (2024). Training is conducted with a batch size of 1 per Nvidia A100-40G GPU, and gradients are accumulated every 4 steps. Each model is trained for up to 10,000 steps (approximately 2 epochs), with the best-performing checkpoint on the validation set selected as part of an early-stopping strategy. The learning rate is set to $1e-5$, and the maximum sequence length during training is capped at 32,768 tokens, discarding any sequences exceeding this threshold.

F.2 Inference Parameters

During inference, we use greedy decoding for all models, allowing a maximum generated sequence length of 1,024 tokens. For RAP decoding, 100 random instances from the validation set are used to probe the attention heads responsible for retrieval. For SFT-DA, retrieval is performed using the designated retrieval attention heads based on the first and only decoded token. For SFT-RTA, retrieval is performed using all tokens generated in the retrieval step to ensure complete context coverage.

Task	Prompt Template

QA	`[INST] Please answer the following question given the following passages:`
	`{Corpus}`
	`Question: {Query}`
	`Answer: [/INST]`

Fact Verification	`[INST] According to the following passages, please verify the given claim and predict your judgment on its factuality as TRUE or FALSE:`
	`{Corpus}`
	`Claim: {Query}`
	`Judgement: [/INST]`

Dialogue Completion	`[INST] According to the given passages, please provide a single response to complete the following conversation by role-playing as either Person A or Person B. Your response should be as knowledgeable and coherent with the conversation history as possible:`
	`{Corpus}`
	`Conversation: {Query}`
	`[/INST]`

Table 8: Prompt template used for RAG tasks in our experiments. `{Corpus}` refers to the provided contextual knowledge base, and `{Query}` refers to a query in ICR^2 or LOFT. A `{Query}` can be a question in the question answering tasks (Kwiatkowski et al., 2019; Yang et al., 2018), a claim to be verified in the fact verification task (Thorne et al., 2018), or a conversation history in the dialogue completion task (Dinan et al., 2018).

LLaMA-3-Instruct	LOFT		ICR^2	
	NQ	HPQA	NQ	HPQA
Closebook	0.41	0.19	0.44	0.19
RAG	0.45	0.56	0.53	0.45
RAG w/ TinyBERT	**0.89**	**0.89**	0.55	0.48
Oracle	0.81	0.77	0.80	0.74
SFT-DA	0.80	0.60	0.51	0.52
SFT-RTA	0.84	0.63	0.51	**0.57**
SFT-DA w/ RAP	0.83	0.72	**0.61**	0.60

Table 9: LLaMA-3's performances on NQ and HotpotQA for LOFT and ICR^{2}with our methods on NQ-LOFT and HPQA-LOFT.

F.2.1 RAP Hyperparameter Settings

In this section, we detail the hyperparameters used in applying RAP decoding to SFT-DA and SFT-RTA on ICR^2 and LOFT.

For SFT-DA on ICR^2:

- **NaturalQuestions**: $Q = 4, M = 1$
- **HotpotQA**: $Q = 8, M = 1$
- **FEVER**: $Q = 4, M = 4$
- **WoW**: $Q = 4, M = 4$

For SFT-DA on LOFT:

- **NaturalQuestions**: $Q = 4, M = 4$
- **HotpotQA**: $Q = 4, M = 4$
- **WoW**: $Q = 8, M = 2$

For SFT-RTA on ICR^2:

- **NaturalQuestions**: $Q = 2, M = 1$
- **HotpotQA**: $Q = 2, M = 1$
- **FEVER**: $Q = 2, M = 8$
- **WoW**: $Q = 4, M = 8$

For SFT-RTA on LOFT:

- **NaturalQuestions**: $Q = 8, M = 4$
- **HotpotQA**: $Q = 4, M = 2$
- **WoW**: $Q = 8, M = 2$

G Prompt Templates for Supervised Fine-tuning

We visualize three templates used for supervised fine-tuning LCLMs to perform retrieve-then-generate generation in Figure 8.

Re-ranker Model	K	LOFT			ICR^2			
		NQ	HPQA	MUSI.	NQ	HPQA	FEV	WoW
TinyBERT	8	0.99	0.98	0.80	0.61	0.65	0.68	0.35
MiniLM	8	**1.00**	0.98	0.82	0.66	0.65	0.70	0.27
TinyBERT	32	0.99	0.99	0.81	0.89	0.82	0.87	0.55
MiniLM	32	**1.00**	**1.00**	0.90	0.86	0.81	0.89	0.43
TinyBERT	50	0.99	0.99	0.82	**0.91**	0.84	0.92	0.60
MiniLM	50	**1.00**	**1.00**	0.92	**0.91**	0.88	0.91	0.50
SFT-RTA w/ RAP	/	0.95	0.99	**1.00**	0.90	**0.95**	**0.97**	**0.92**

Table 10: Retrieval performance comparison of re-ranker models across LOFT and ICR^{2}benchmarks.

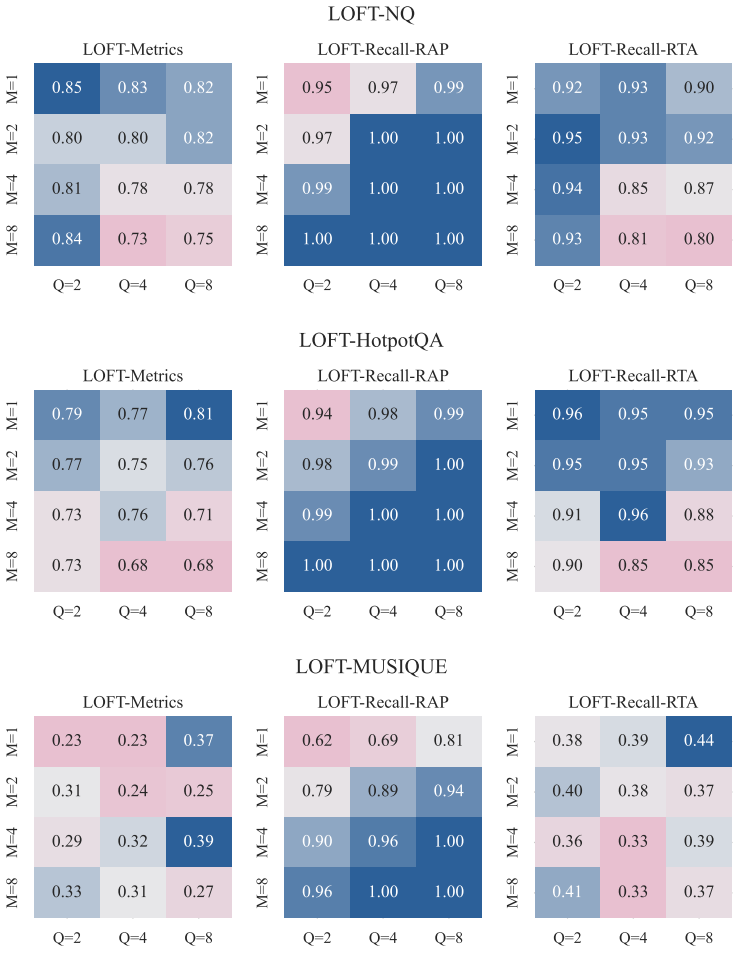

Figure 6: Effect of adjusting Q (number of attention heads for retrieval) and M (number of passages to select) when applying SFT-RTA + RAP on all tasks in LOFT. Left column: average model performance measured by the tasks-specific metrics. Middle column: retrieval performance measured by the recall rate. Right column: retrieval performance measured by the exact match.

H Visualization for Joint Retrieval Head Training

We visualize the architecture for adding a retrieval head jointly trained with language generation in Figure 9 for easing the understanding of the approach.

I Effect of Retrieval Delineation

As outlined in Section 4.1, our retrieve-then-generate variants, SFT-RTA and SFT-CCI, utilize special symbols `<RETRIEVAL>` and `</RETRIEVAL>` to explicitly mark the boundaries of the retrieval step. To examine the impact of these boundary to-

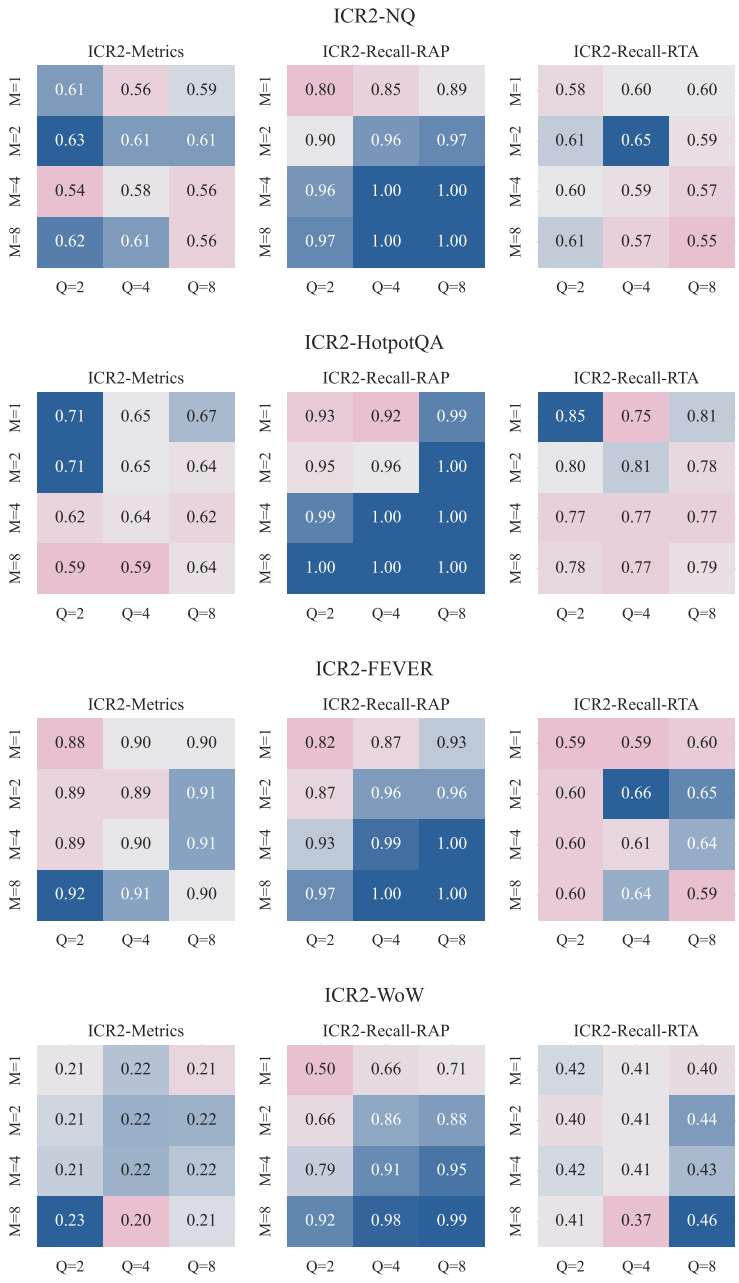

Figure 7: Effect of adjusting Q (number of attention heads for retrieval) and M (number of passages to select) when applying SFT-RTA + RAP on all tasks in ICR^2. Left column: average model performance measured by the tasks-specific metrics. Middle column: retrieval performance measured by the recall rate. Right column: retrieval performance measured by the exact match.

	Benchmark	SFT-RTA	*w/o* <RET>	SFT-CCI	*w/o* <RET>
LOFT	**NQ**	0.74	0.7 (↓ .04)	0.76	0.63 (↓ .13)
	HPQA	0.69	0.66 (↓ .03)	0.54	0.58 (↑ .04)
	MUSI.	0.33	0.24 (↓ .09)	0.35	0.24 (↓ .11)
ICR^2	**NQ**	0.60	0.56 (↓ .04)	0.63	0.61 (↓ .02)
	HPQA	0.67	0.7 (↓ .03)	0.63	0.65 (↑ .02)
	FEVER	0.91	0.92 (↑ .01)	0.89	0.86 (↓ .03)
	WoW	0.22	0.20 (↓ .02)	0.21	0.22 (↑ .01)

Table 11: Effect of removing the special boundary tokens for the retrieve-then-generate variants.

kens, we conducted an ablation study. As shown in Table 11, removing these tokens in general leads to a decline in model performance. This suggests that clearly delineating the retrieval step is crucial for the model to learn effectively.

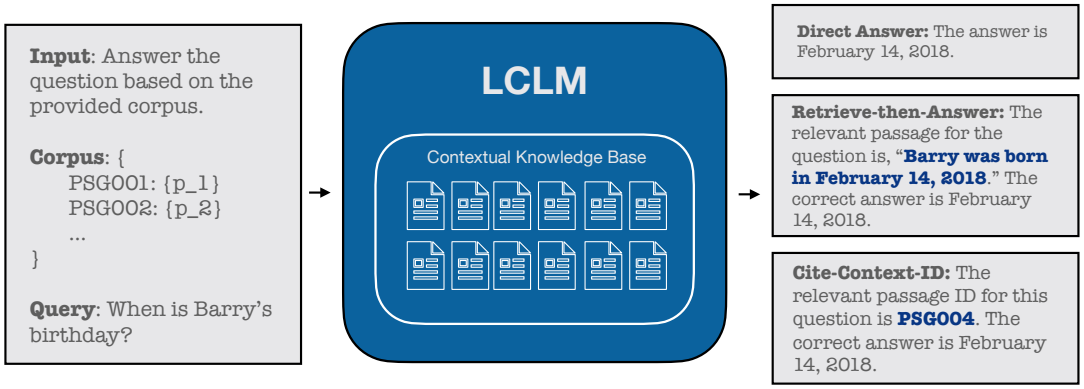

Figure 8: Templates for Direct Answer (DA), Retrieve-Then-Answer (RTA) and Cite-Context-ID (CCI).

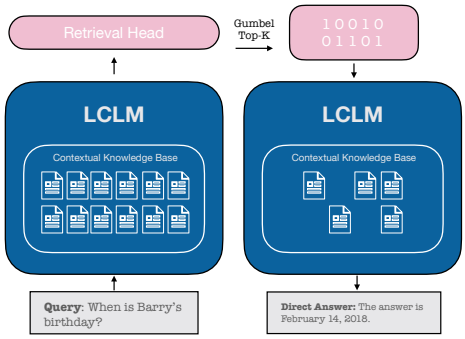

Figure 9: A retrieval head is added to predict the Top-K context for generating final responses. The head is jointly optimized with the generation head.