Don't Do RAG: When Cache-Augmented Generation is All You Need for Knowledge Tasks

Brian J Chan*
Chao-Ting Chen*
Jui-Hung Cheng*
Department of Computer Science
National Chengchi University
Taipei, Taiwan
{110703065,110703038,110703007}@nccu.edu.tw

Hen-Hsen Huang
Insititue of Information Science
Academia Sinica
Taipei, Taiwan
hhhuang@iis.sinica.edu.tw

Abstract

Retrieval-augmented generation (RAG) has gained traction as a powerful approach for enhancing language models by integrating external knowledge sources. However, RAG introduces challenges such as retrieval latency, potential errors in document selection, and increased system complexity. With the advent of large language models (LLMs) featuring significantly extended context windows, this paper proposes an alternative paradigm, cache-augmented generation (CAG) that bypasses real-time retrieval. Our method involves preloading all relevant resources, especially when the documents or knowledge for retrieval are of a limited and manageable size, into the LLM's extended context and caching its runtime parameters. During inference, the model utilizes these preloaded parameters to answer queries without additional retrieval steps. Comparative analyses reveal that CAG eliminates retrieval latency and minimizes retrieval errors while maintaining context relevance. Performance evaluations across multiple benchmarks highlight scenarios where long-context LLMs either outperform or complement traditional RAG pipelines. These findings suggest that, for certain applications, particularly those with a constrained knowledge base, CAG provide a streamlined and efficient alternative to RAG, achieving comparable or superior results with reduced complexity.

CCS Concepts

• **Computing methodologies** → **Discourse, dialogue and pragmatics**; **Natural language generation**; • **Information systems** → **Specialized information retrieval**.

Keywords

Cache Augmented Generation, Retrieval Augmented Generation, Retrieval-Free Question Answering, Large Language Models

*Three authors contributed equally to this research.

WWW Companion '25, April 28-May 2, 2025, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1331-6/2025/04
https://doi.org/10.1145/3701716.3715490

ACM Reference Format:
Brian J Chan, Chao-Ting Chen, Jui-Hung Cheng, and Hen-Hsen Huang. 2025. Don't Do RAG: When Cache-Augmented Generation is All You Need for Knowledge Tasks. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3701716.3715490

1 Introduction

The advent of retrieval-augmented generation (RAG) [2, 5] has significantly enhanced the capabilities of large language models (LLMs) by dynamically integrating external knowledge sources. RAG systems have proven effective in handling open-domain questions and specialized tasks, leveraging retrieval pipelines to provide contextually relevant answers. However, RAG is not without its drawbacks. The need for real-time retrieval introduces latency, while errors in selecting or ranking relevant documents can degrade the quality of the generated responses. Additionally, integrating retrieval and generation components increases system complexity, necessitating careful tuning and adding to the maintenance overhead.

This paper proposes an alternative paradigm, cache-augmented generation (CAG), leveraging the capabilities of long-context LLMs to address these challenges. Instead of relying on a retrieval pipeline, as shown in Figure 1, our approach involves preloading the LLM with all relevant documents in advance and precomputing the key-value (KV) cache [9], which encapsulates the inference state of the LLM. The preloaded context enables the model to provide rich, contextually accurate answers without the need for additional retrieval during runtime. This approach eliminates retrieval latency, mitigates retrieval errors, and simplifies system architecture, all while maintaining high-quality responses by ensuring the model processes all relevant context holistically.

Recent advances in long-context LLMs have extended their ability to process and reason over substantial textual inputs. For example, Llama 3.1 [1] was trained with a 128K context length, and its effective context length is 32K in Llama 3.1 8B and 64K in Llama 3.1 70B [3]. This 32K to 64K context window is sufficient for storing knowledge sources such as internal company documentation, FAQs, customer support logs, and domain-specific databases, making it practical for many real-world applications. By accommodating larger context windows, these models can assimilate extensive information in a single inference step, making them well-suited for tasks like document comprehension, multi-turn dialogue, and

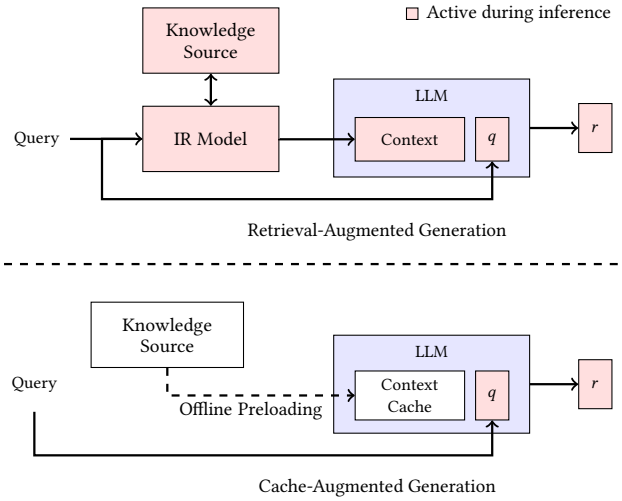

Figure 1: Comparison of Retrieval-Augmented Generation (RAG) and our Cache-Augmented Generation (CAG) Workflows: The pink-shaded components represent the processes active during real-time inference. In RAG (top section), the IR model retrieves relevant information from the knowledge source, and both the retrieved knowledge and query are processed by the LLM during inference, introducing retrieval latency. In contrast, CAG (bottom section) preloads and caches knowledge offline, allowing the LLM to process only the query during inference, eliminating retrieval overhead and ensuring a more efficient generation process.

summarization of lengthy texts. This capability eliminates the dependency on real-time retrieval, as all necessary information can be preloaded into the model. These developments create opportunities to streamline workflows for knowledge-intensive tasks, potentially reducing or even eliminating the need for traditional RAG systems.

Recent studies [4, 7] have investigated the performance of long-context models in RAG tasks, revealing that state-of-the-art models like GPT-o1, GPT-4, and Claude 3.5 can effectively process large amounts of retrieved data, outperforming traditional systems in many scenarios. Findings suggest that as long as all documents fit within the extended context length, traditional RAG systems can be replaced by these long-context models. Similarly, Lu et al. [8] has demonstrated the benefits of precomputed KV caching to improve efficiency, albeit with the need for position ID rearrangement to enable proper functioning. Nonetheless, these methods remain vulnerable to retrieval failures inherent to RAG systems.

Through a series of experiments comparing traditional RAG workflows with our proposed approach, we identify scenarios where long-context LLMs outperform RAG in both efficiency and accuracy. By addressing the technical and practical implications, this paper aims to provide insights into when and why CAG may serve as a streamlined, effective alternative to RAG, particularly for cases where the documents or knowledge for retrieval are of limited, manageable size. Our findings challenge the default reliance on RAG for knowledge integration tasks, offering a simplified, robust solution to harness the growing capabilities of long-context LLMs. Our contributions are threefold as follows:

- **Efficient Alternative to RAG**: We introduced a novel approach leveraging long-context LLMs with preloaded documents and precomputed KV caches, mitigating retrieval latency, errors, and system complexity.
- **Quantitative Analysis**: We conducted extensive experiments showing scenarios where long-context LLMs outperform traditional RAG systems, especially with manageable knowledge bases.
- **Practical Insights**: This work provided actionable insights into optimizing knowledge-intensive workflows, demonstrating the viability of retrieval-free methods for specific applications. Our CAG framework is released publicly.[1]

2 Methodology

Our CAG framework leverages the extended context capabilities of long-context LLMs to enable retrieval-free knowledge integration. By preloading external knowledge sources, such as a collection of documents $\mathcal{D} = \{d_1, d_2, \dots\}$, and precomputing the KV cache C_{KV}, we address the computational challenges and inefficiencies inherent to real-time retrieval in traditional RAG systems. The operation of our framework is divided into three phases:

(1) **External Knowledge Preloading**
In this phase, a curated collection of documents $\mathcal{D}$ relevant to the target application is preprocessed and formatted to fit within the model's extended context window. The LLM $\mathcal{M}$, with parameters θ, processes $\mathcal{D}$, transforming it into a precomputed KV cache:

$$C_{\text{KV}} = \texttt{KV-Encode}(\mathcal{D}) \tag{1}$$

This KV cache, which encapsulates the inference state of the LLM, is stored on disk or in memory for future use. The computational cost of processing $\mathcal{D}$ is incurred only once, regardless of the number of subsequent queries.

(2) **Inference**
During inference, the precomputed KV cache C_{KV} is loaded alongside the user's query q. The LLM utilizes this cached context to generate responses:

$$r = \mathcal{M}(\mathcal{D} \oplus q) = \mathcal{M}(q \mid C_{\text{KV}}) \tag{2}$$

By preloading the external knowledge, this phase eliminates retrieval latency and reduces risks of errors or omissions that arise from dynamic retrieval. The combined prompt $\mathcal{D} \oplus q$ ensures a unified understanding of both the external knowledge and the user query.

(3) **Cache Reset**
To maintain system performance across multiple inference sessions, the KV cache, stored in memory, can be reset efficiently. As the KV cache grows in an append-only manner with new tokens $q = (t_1, t_2, \dots, t_k)$ sequentially appended, resetting involves truncating these new tokens. This allows for rapid reinitialization without reloading the entire cache from disk, ensuring sustained speed and responsiveness.

[1] https://github.com/hhhuang/CAG

The proposed methodology offers several significant advantages over traditional RAG systems:

- **Reduced Inference Time**: By eliminating the need for real-time retrieval, the inference process becomes faster and more efficient, enabling quicker responses to user queries.
- **Unified Context**: Preloading the entire knowledge collection into the LLM provides a holistic and coherent understanding of the documents, resulting in improved response quality and consistency across a wide range of tasks.
- **Simplified Architecture**: By removing the need to integrate retrievers and generators, the system becomes more streamlined, reducing complexity, improving maintainability, and lowering development overhead.

Looking forward, our approach is poised to become even more powerful with the anticipated advancements in LLMs. As future models continue to expand their context length, they will be able to process increasingly larger knowledge collections in a single inference step. Additionally, the improved ability of these models to extract and utilize relevant information from long contexts will further enhance their performance. These two trends will significantly extend the usability of our approach, enabling it to handle more complex and diverse applications. Consequently, our methodology is well-positioned to become a robust and versatile solution for knowledge-intensive tasks, leveraging the growing capabilities of next-generation LLMs.

3 Experiments

3.1 Experimental Setup

To evaluate the effectiveness of our proposed method, we conducted experiments using two widely recognized question-answering benchmarks: the Stanford Question Answering Dataset (SQuAD) 1.0 [10] and the HotPotQA dataset [11]. These datasets provide complementary challenges, with SQuAD focusing on precise, context-aware answers within single passages and HotPotQA emphasizing multi-hop reasoning across multiple documents. Each of both datasets consists of documents $\mathcal{D} = \{d_1, d_2, \dots\}$ paired with questions $\mathcal{Q} = \{q_1, q_2, \dots\}$ and golden responses $\mathcal{R} = \{r_1, r_2, \dots\}$. These datasets provide a robust platform for assessing both single-context comprehension and complex multi-hop reasoning.

To investigate how different levels of reference text length impact retrieval difficulty, we created three test sets for each dataset, varying the size of the reference text. For example, in the HotPotQA-small configuration, we sampled 16 documents $\mathcal{D}_s \subset \mathcal{D}$ from the HotPotQA document set to form a long reference text. QA pairs associated with $\mathcal{D}_s$ were selected as test instances. The same methodology was applied to create test sets for SQuAD.

The dataset statistics are summarized in Table 1. As the number of documents (and hence the length of the reference text) increases, the task becomes more challenging, particularly for RAG systems. Longer reference texts increase the difficulty of accurately retrieving the correct information, which is crucial for LLMs to generate high-quality responses.

The primary task involves generating accurate and contextually relevant answers $\hat{\mathcal{R}} = \{\hat{r}_1, \hat{r}_2, \dots\}$ for the SQuAD and HotPotQA questions, based on the respective preloaded passages. By leveraging the precomputed key-value cache $C_{\text{KV}} = \texttt{KV-Encode}(\mathcal{D})$, our system generates responses $\hat{r}_i = \mathcal{M}(q_i \mid C_{\text{KV}})$ without relying on retrieval mechanisms during inference. This unified approach allows for direct performance comparisons against traditional RAG systems, highlighting the strengths and limitations of our method across diverse QA challenges.

Table 1: The SQuAD and HotPotQA test sets with varying reference text lengths, highlighting the number of documents, questions, and associated responses for each configuration.

Source	Size	# Docs	# Tokens	# QA Pairs
HotPotQA	Small	16	21k	1,392
	Medium	32	43k	1,056
	Large	64	85k	1,344
SQuAD	Small	3	21k	500
	Medium	4	32k	500
	Large	7	50k	500

The experiments were executed on Tesla V100 32G × 8 GPUs. For all experiments, we used Llama 3.1 8B [1] as the underlying LLM across all systems, including both the RAG baselines and our proposed method. This model supports input sizes of up to 128k tokens, enabling the processing of extensive contexts. For our proposed method, the context of each dataset was preloaded into the model via a precomputed KV cache. For SQuAD, the documents $\mathcal{D}_{\text{S}}$ were encoded into a KV cache $C^{\text{S}}_{\text{KV}} = \texttt{KV-Encode}(\mathcal{D}_{\text{S}})$, while for HotPotQA, the documents $\mathcal{D}_{\text{H}}$ were encoded into $C^{\text{H}}_{\text{KV}} = \texttt{KV-Encode}(\mathcal{D}_{\text{H}})$. These caches were stored offline and loaded during inference to eliminate the need for real-time retrieval, ensuring comprehensive access to all relevant information for each dataset.

Our experiments were conducted on both the SQuAD and HotPotQA datasets to evaluate the performance of different systems in terms of similarity to ground-truth answers, measured using BERTScore [12]. Each dataset—SQuAD and HotPotQA—was evaluated separately, with retrieval systems configured to fetch passages exclusively from the respective dataset to ensure focused and fair evaluation.

3.2 Baseline Systems

The baseline RAG systems were implemented using the LlamaIndex framework,[2] employing two retrieval strategies: BM25 for sparse retrieval and OpenAI Indexes for dense retrieval. The details of each baseline system are as follows:

(1) **Sparse Retrieval System (BM25)**: The first baseline system employed BM25 indexes for retrieval. BM25, a sparse retrieval algorithm, ranks documents based on term frequency-inverse document frequency (TF-IDF) and document length normalization. Given a query q_i, BM25 retrieves the top-k passages $\mathcal{P}_k = \{p_1, p_2, \dots, p_k\}$ from the indexed collection $\mathcal{D}$. These passages were then passed to the generator, $\mathcal{M}$, to synthesize answers:

$$\hat{r}_i = \mathcal{M}(q_i \mid \mathcal{P}_k) \tag{3}$$

[2] https://www.llamaindex.ai/framework

Table 2: Experimental Results

Size	System	Top-k	HotPotQA BERT-Score	SQuAD BERT-Score
Small	Sparse RAG	1	0.6788	0.7214
		3	0.7626	0.7616
		5	0.7676	0.7608
		10	0.7521	0.7584
	Dense RAG	1	0.7164	0.6216
		3	0.7582	0.7106
		5	0.7481	0.7334
		10	0.7576	0.7586
	CAG (Ours)		**0.7951**	**0.7695**
Medium	Sparse RAG	1	0.6592	0.6902
		3	0.7546	0.7301
		5	0.7633	0.7298
		10	0.7458	0.7262
	Dense RAG	1	0.6973	0.5871
		3	0.7432	0.6702
		5	0.7322	0.6890
		10	0.7308	0.7310
	CAG (Ours)		**0.7821**	**0.7383**
Large	Sparse RAG	1	0.6616	0.7254
		3	0.7463	0.7634
		5	**0.7535**	0.7658
		10	0.7345	0.7613
	Dense RAG	1	0.7020	0.6070
		3	0.7409	0.7018
		5	0.7234	0.7286
		10	0.7374	0.7590
	CAG (Ours)		0.7407	**0.7734**

BM25 provides a robust and interpretable retrieval mechanism, suited for tasks involving keyword matching.

(2) **Dense Retrieval System (OpenAI Indexes)**: The second baseline utilized OpenAI indexes,[3] which employ dense embeddings to represent both documents and queries in a shared semantic space. For a query q_i, dense retrieval selects the top-k passages $\mathcal{P}_k$ that semantically align with the query, offering improved contextual understanding compared to sparse methods. These passages were similarly passed to the generator for answer synthesis as Equation 3. This system is particularly effective for questions requiring nuanced contextual matching beyond exact term overlap.

For the RAG baselines, the top-1, top-3, top-5, and top-10 retrieved passages were used for inference. In contrast, our CAG utilized the preloaded context specific to each dataset to generate answers without retrieval constraints.

3.3 Results

As shown in Table 2, the experimental results highlight key distinctions between our proposed CAG approach and RAG systems. CAG consistently achieved the highest BERTScore in most cases,

[3]https://cookbook.openai.com/examples/evaluation/evaluate_rag_with_llamaindex

Table 3: Response Time (Seconds) Comparison on HotPotQA

Size	System	Retrieval	Generation
Small	Sparse RAG, Top-3	0.0008	0.7406
	Sparse RAG, Top-10	0.0012	1.5595
	Dense RAG, Top-3	0.4849	1.0093
	Dense RAG, Top-10	0.3803	2.6608
	CAG	-	0.8512
	In-Context Learning	-	9.3197
Medium	Sparse RAG, Top-3	0.0008	0.7148
	Sparse RAG, Top-10	0.0012	1.5306
	Dense RAG, Top-3	0.4140	0.9566
	Dense RAG, Top-10	0.4171	2.6361
	CAG	-	1.4078
	In-Context Learning	-	26.3717
Large	Sparse RAG, Top-3	0.0008	0.6667
	Sparse RAG, Top-10	0.0012	1.5175
	Dense RAG, Top-3	0.4123	0.9331
	Dense RAG, Top-10	0.4100	2.6447
	CAG	-	2.2631
	In-Context Learning	-	92.0824

outperforming both sparse and dense RAG methods. By preloading the entire reference text from the test set, our method is immune to retrieval errors, ensuring holistic reasoning over all relevant information. This advantage is particularly evident in scenarios where RAG systems struggle with retrieving incomplete or irrelevant passages, leading to suboptimal answer generation.

However, as the data size increases, the performance gap between CAG and RAG narrows slightly, aligning with prior findings that long-context LLMs may experience degradation when handling very long contexts [6]. Additionally, the fact that sparse RAG outperforms dense RAG suggests that the datasets may not be sufficiently challenging, allowing traditional sparse retrieval to effectively capture most relevant information without requiring deeper semantic retrieval. Despite these factors, the results underscore the robustness and efficiency of CAG, particularly for tasks that require a unified understanding of the source material. By fully leveraging the long-context capabilities of Llama 3.1, our approach bypasses retrieval challenges and maintains superior performance in retrieval-free knowledge integration.

Table 3 and Figure 2 show the retrieval and generation time across different HotPotQA knowledge sizes for various RAG methods and CAG. CAG eliminates retrieval time entirely, whereas sparse and dense retrieval-based RAG systems require additional retrieval steps, with dense retrieval incurring higher latency. Sparse RAG exhibits insignificant retrieval latency in the experiments, but still requires retrieval before generation. As the knowledge size increases, generation time grows across all methods, including CAG, highlighting the computational cost of handling longer contexts. However, CAG remains more efficient than dense RAG, as it avoids retrieval overhead while maintaining comparable or superior response times. Table 3 also compares our CAG approach with standard in-context learning, where the reference text is provided

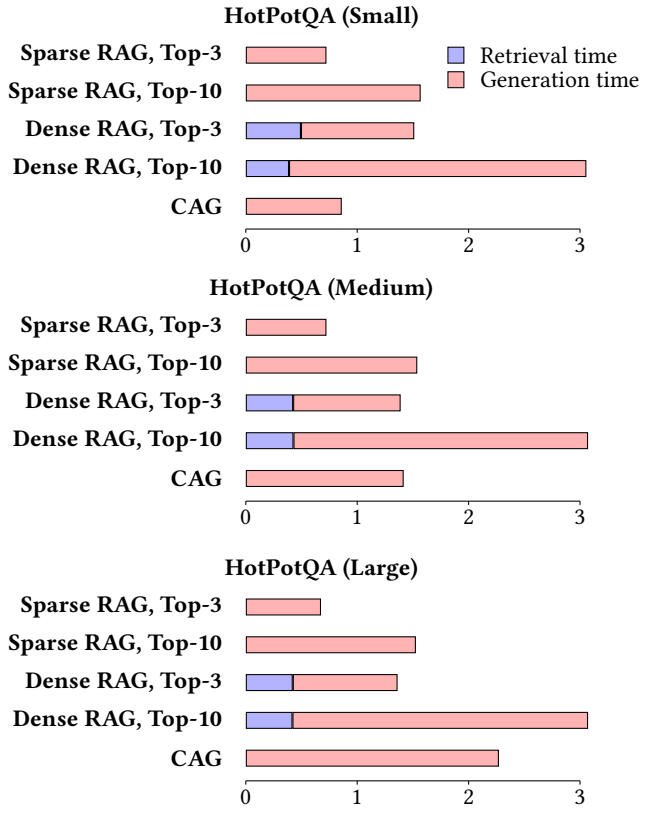

Figure 2: Response Time Comparison on HotPotQA (Seconds). The x-axis represents response time in seconds across different knowledge sizes. CAG eliminates retrieval overhead, while dense RAG incurs longer retrieval and generation times due to retrieving and feeding longer text chunks into the LLM. Sparse RAG retrieves shorter text spans, resulting in faster generation. As the knowledge size increases, generation time grows for all methods, but CAG remains competitive while bypassing retrieval completely.

dynamically during inference, requiring real-time KV-cache computation. The results demonstrate that CAG dramatically reduces generation time, particularly as the reference text length increases. This efficiency stems from preloading the KV-cache, which eliminates the need to process the reference text on the fly.

4 Conclusion

As long-context LLMs evolve, we present a compelling case for rethinking traditional RAG workflows. While our work emphasizes eliminating retrieval latency, there is potential for hybrid approaches that combine preloading with selective retrieval. For example, a system could preload a foundation context and use retrieval only to augment edge cases or highly specific queries. This would balance the efficiency of preloading with the flexibility of retrieval, making it suitable for scenarios where context completeness and adaptability are equally important.

Limitations

Our method requires loading all relevant documents into the models context, making it well-suited for use cases such as internal knowledge bases of small companies, FAQs, and call centers, where the knowledge source is of a manageable size. However, this approach becomes impractical for significantly larger datasets. Fortunately, as LLMs continue to expand their context lengths and hardware capabilities advance, this limitation is expected to diminish, enabling broader applicability in the future.

Acknowledgments

This work was partially supported by National Science and Technology Council (NSTC), Taiwan, under the grant 112-2221-E-001-016-MY3, by Academia Sinica, under the grant 236d-1120205, and by National Center for High-performance Computing (NCHC), National Applied Research Laboratories (NARLabs), and NSTC under the project "Trustworthy AI Dialog Engine, TAIDE." We thank Discover AI[4] and the many individuals who have introduced, shared, and discussed our work, contributing to its broader visibility.

References

[1] 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* (2024).

[2] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).

[3] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the Real Context Size of Your Long-Context Language Models?. In *First Conference on Language Modeling*. https://openreview.net/forum?id=kIoBbc76Sy

[4] Quinn Leng, Jacob Portes, Sam Havens, Matei Zaharia, and Michael Carbin. 2024. Long Context RAG Performance of Large Language Models. *arXiv preprint arXiv:2411.03538* (2024).

[5] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[6] Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context LLMs Struggle with Long In-context Learning. arXiv:2404.02060 [cs.CL] https://arxiv.org/abs/2404.02060

[7] Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Retrieval Augmented Generation or Long-Context LLMs? A Comprehensive Study and Hybrid Approach. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Association for Computational Linguistics, Miami, Florida, US, 881–893. https://doi.org/10.18653/v1/2024.emnlp-industry.66

[8] Songshuo Lu, Hua Wang, Yutian Rong, Zhi Chen, and Yaohua Tang. 2024. TurboRAG: Accelerating Retrieval-Augmented Generation with Precomputed KV Caches for Chunked Text. arXiv:2410.07590 [cs.CV] https://arxiv.org/abs/2410.07590

[9] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems* 5 (2023), 606–624.

[10] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Jian Su, Kevin Duh, and Xavier Carreras (Eds.). Association for Computational Linguistics, Austin, Texas, 2383–2392. https://doi.org/10.18653/v1/D16-1264

[11] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[12] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. [n. d.]. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

[4] https://www.youtube.com/watch?v=NaEf_uiFX6o