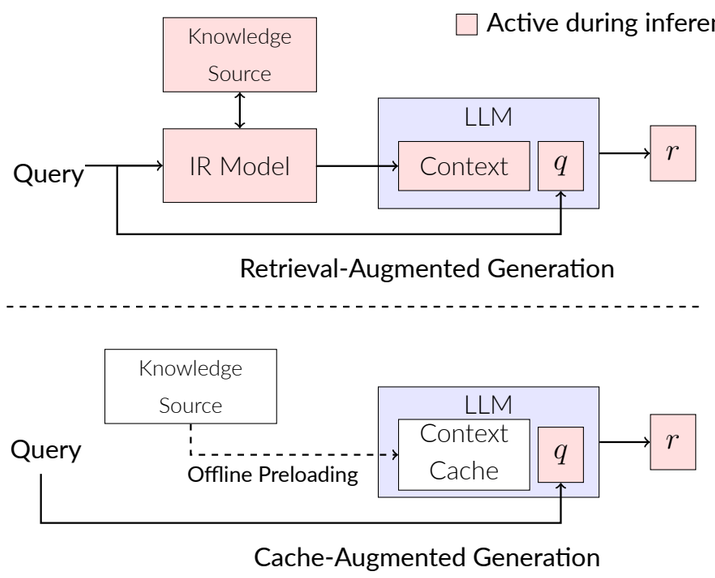
Don't Do RAG: When Cache-Augmented Generation is All You Need for Knowledge Tasks

Brian J Chan Chao-Ting Chen Jui-Hung Cheng * Hen-Hsen Huang

Department of Computer Science, National Chengchi University, Taipei, Taiwan
*Insititue of Information Science, Academia Sinica, Taipei, Taiwan

Introduction

Recent advancements in large language models (LLMs) with extended context windows open new possibilities for retrieval-free knowledge integration. This work introduces Cache-Augmented Generation (CAG) — a streamlined alternative to Retrieval-Augmented Generation (RAG). By preloading relevant documents and caching key-value inference states, CAG eliminates retrieval latency, reduces errors, and simplifies system design. Experiments on SQuAD and HotPotQA demonstrate that CAG achieves comparable or superior accuracy and efficiency, especially when the knowledge base is of manageable size. Our findings challenge the default reliance on RAG, offering a alternative solution for knowledge-intensive tasks.



Methodology

By preloading external knowledge sources, such as a collection of documents $D = d_1, d_2, ...$, and precomputing the KV cache C_{KV}, we address the computational challenges and inefficiencies inherent to real-time retrieval in traditional RAG systems.

- **External Knowledge Preloading:** The LLM M, with parameters θ, processes D, transforming it into a precomputed KV cache: $C_{KV} = KV - Encode(D)$
- **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: $r = M(D \oplus q) = M(q|C_{KV})$
- **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 $q = (t_1, t_2, ..., t_k$ sequentially appended, resetting involves truncating these new tokens.

Comparison

1. **Cost of RAG**: Retrieval-Augmented Generation (RAG) systems come with several inherent costs. One major challenge is retrieval latency, as dense information retrieval (IR) methods tend to be slow and consume significant computational resources. Additionally, retrieval errors can propagate through the system, with sparse IR approaches often missing critical relevance information. Furthermore, RAG introduces system complexity due to the need for seamless integration between a search engine and a large language model (LLM), which can complicate deployment and maintenance.
2. **Advantages of CAG**: As the context window and capability of large language models continues to expand, CAG (Cached-Augmented Generation) can gradually replace traditional RAG (Retrieval-Augmented Generation) applications. By eliminating real-time retrieval, CAG significantly reduces latency and enables faster inference while maintaining strong contextual relevance. With longer context windows, models can directly process larger amounts of background information, reducing the need for external retrieval systems. This not only simplifies system design but also enhances reliability, making CAG an increasingly effective and efficient alternative to RAG in a growing number of use cases.

Evaluation

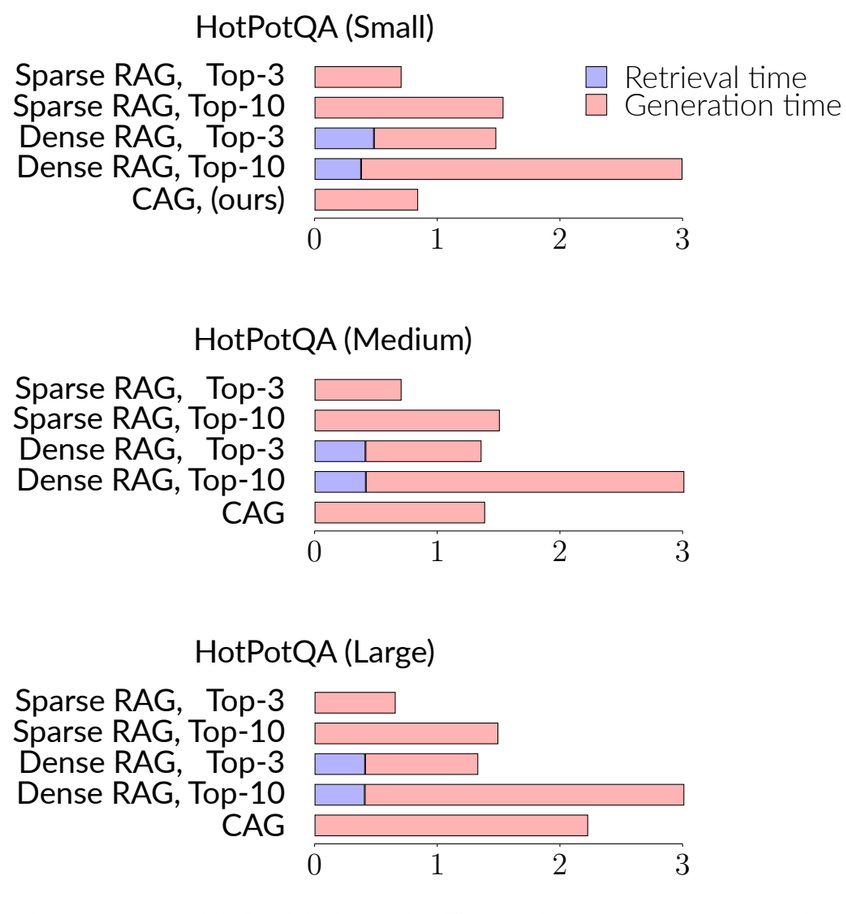
The SQuAD and HotPotQA test sets with varying reference text lengths, highlighting the number of documents, questions, and associated responses for each configuration.

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

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

Results



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.