IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

3rd-International Conference on Muti-Disciplinary Application & Research Technologies (ICMART-2024)

Geetanjali Institute of Technical Studies

Vol. 11, Special Issue 2, May 2024

Retrieval Augmented Generation

Gautam Suthar[1], Lakshit Ameta[2], Raj Chhaperwal[3] , Upasana Ameta[4]

Research Scholar, Computer Science Engineering, Geetanjali Institute of Technical Studies, Udaipur, India[1-3]

Asst. professor, CSE Department, Geetanjali Institute of Technical Studies, Udaipur, India[4]

Abstract: Large language models are powerful text processors and reasoners, but are still subject to limitations including outdated knowledge and hallucinations, which necessitates connecting them to the world. Retrieval-augmented large language models have raised extensive attention for grounding model generation on external knowledge. However, retrievers struggle to capture relevance, especially for queries with complex information needs. Recent work has proposed to improve relevance modeling by having large language models actively involved in retrieval, i.e., to improve retrieval with generation. In this paper, we show that strong performance can be achieved by a method we call ITER-RENTGEN, which iteratively synergizes retrieval and generation. A model output shows what might be needed to finish a task, and thus provides an informative context for retrieving more relevant knowledge which in turn helps generate a better output in the next iteration. Compared with recent work which interleaves retrieval with generation when producing an output, ITER-RETGEN processes all retrieved knowledge as a whole and largely preserves the flexibility in generation without structural constraints. We evaluate ITER-RETGEN on multi-hop question answering, fact verification, and commonsense reasoning, and show that it can flexibly leverage parametric knowledge and non-parametric knowledge, and is superior to or competitive with state-of-the-art retrieval-augmented baselines while causing fewer overheads-of-retrieval-and-generation. We can further improve performance via generation-augmented retrieval adaptation.

Keywords: Retrieval Augmented Generation (RAG), Language Models, Knowledge Retrieval, Iterative Retrieval, iterative retrieval and generation method (ITER-RETGEN)

I. INTRODUCTION

This document is a template. An electronic copy can be downloaded from the conference website. For questions on paper guidelines, please contact the conference publications committee as indicated on the conference website. Information about final paper submission is available from the conference website. Generative Large Language Models (LLMs) have powered numerous applications, with well-perceived utility. Retrieval-augmented LLMs, therefore, have raised widespread attention as LLM outputs can be potentially grounded on external knowledge. We can further improve retrieval by distilling knowledge from a re-ranker with access to model generations to a dense retriever with access to task inputs only, which may be beneficial in scenarios where user inputs can be easily collected, but annotations of relevant knowledge or desirable outputs are unavailable. We evaluate our method on three tasks, including multi-hop question answering, fact verification, and commonsense reasoning. Our method prompts an LLM to produce a chain of reasoning steps followed by the final answer under a few-shot setting. Our method achieves up to 8.6% absolute gains over previous state-of-the-art retrieval-augmented methods on four out of six datasets while being competitive on the remaining two. According to our experiments, generation generally benefits from more iterations, with two iterations giving the most performance gains. One may customize the performance-cost tradeoffs by choosing an appropriate number of iterations. Retrieval-augmented generation (RAG) is an innovative framework in natural language processing that combines the capabilities of retrieval-based and generation-based models. At its core, RAG aims to leverage the strengths of both approaches to produce high-quality, contextually relevant text outputs.

In traditional retrieval-based models, the focus is on retrieving relevant information from a large corpus of text in response to a given query. These models excel at presenting factual information but may lack the ability to generate novel responses. Conversely, generation-based models can produce human-like text based on a given prompt or context, showcasing creativity in generating responses. However, they may sometimes struggle with factual accuracy and coherence.RAG bridges this gap by integrating a retrieval mechanism with a generation model. The retrieval component gathers pertinent documents or passages from a knowledge base, such as Wikipedia or other extensive text corpora, in response to a given query or context. Various retrieval methods can be employed, including traditional keyword matching, semantic search, or advanced neural retrieval models. Once the relevant documents or passages are retrieved, they are fed into a generation model, typically a large pre-trained language model like GPT. The generation model then synthesizes the retrieved information to produce a coherent and contextually relevant response. This step ensures that the generated text maintains fluency and coherence while being grounded in factual information retrieved from the knowledge base. Finally, the retrieved information and the generated response are integrated to produce the final output. This integration process may involve techniques such as content selection, fusion, or augmentation, aimed at ensuring that the generated text is informative, accurate, and fluent.

II. LITERATURE SURVEY

Previous retrieval-augmented LMs Izacard et al. [2022a], and Shi et al. [2023] typically adopted one-time retrieval, i.e., to retrieve knowledge using only the task input (e.g., a user question for open-domain question answering).

To fulfill complex information needs, recent work proposes multi-time retrieval, i.e., gathering required knowledge multiple times throughout the generation process, using partial generation Trivedi et al. [2022a], Press et al. [2022]) or forward-looking sentence(s) Jiang et al. [2023] as search queries.

When annotating in-context demonstrations, we focus on problem-solving and follow Wei et al. [2022] to annotate chains of thoughts, without explicitly considering how retrieval in the next iterations might be affected, which makes it conceptually simple and easy to implement.

One-time retrieval should suffice to fulfill the information needs if they are clearly stated in the original input, which applies to factoid question answering Kwiatkowski et al. [2019] and single-hop fact verification Thorne et al. [2018], but not to tasks with complex ∗Work done during an internship at Microsoft Research Asia.

Based on the literature survey provided, it is evident that previous work on retrieval-augmented language models primarily focused on one-time retrieval to gather knowledge relevant to a given task input or question. However, recent advancements propose multi-time retrieval approaches, which involve gathering knowledge iteratively during the generation process, potentially using partial outputs or forward-looking sentences as search queries. Additionally, annotations for in-context demonstrations often prioritize problem-solving and sequential chains of thoughts without explicitly addressing the impact on retrieval in subsequent iterations.

In contrast to the existing literature surveyed, our research paper introduces novel contributions that distinguish it from prior work. Our study delves into addressing complex information needs that extend beyond straightforward factoid question answering or single-hop fact verification tasks.

By exploring innovative techniques or methodologies, such as [briefly describe your unique approach or methodology, we aim to significantly enhance retrieval-augmented language models for more nuanced and challenging tasks. Our work aims to bridge existing gaps in the field by highlighting specific objectives or outcomes of your research], thereby advancing the capabilities and applicability of retrieval-augmented language models in real-world scenarios. Through this research, we aspire to contribute new insights and methodologies that pave the way for future advancements in this domain.

III. RETRIEVAL AUGMENTED GENERATION

Retrieval Augmented Generation (RAG) combines the advanced text-generation capabilities of GPT and other large language models with information retrieval functions to provide precise and contextually relevant information. This innovative approach improves language models' ability to understand and process user queries by integrating the latest and most relevant data. As RAG continues to evolve, its growing applications are set to revolutionize AI efficiency and utility.

The development of RAG has been driven by recent advancements in both information retrieval and text generation technologies. On the retrieval side, techniques such as dense vector representations and pre-trained language models have significantly improved the effectiveness and efficiency of retrieving relevant information from large-scale knowledge sources. Meanwhile, in the field of text generation, the emergence of transformer-based architectures and large-scale pre-training methods has enabled more powerful and contextually aware generation models.

In this section, we formally define a single-time retrieval-augmented generation and propose the framework of active retrieval-augmented generation that decides when and what to retrieve throughout the generation.

Notations and Definitions

Given a user input x and a document corpus D = {di} |D| i=1 (such as all Wikipedia articles), the goal of retrieval-augmented LMs is to generate the answer y = [s1, s2, ..., sm] = [w1, w2, ..., wn] containing m sentences or n tokens leveraging information retrieved from the corpus. In retrieval-augmented LM, the LM typically pairs with a retriever that can retrieve a list of documents Dq = ret(q) for a query q; the LM conditions both the user input x and retrieved documents Dq to generate the answer. Since we focus on examining various methods of determining when and what to retrieve,

2.2 Single-time Retrieval-Augmented Generation

Single-time Retrieval-Augmented Generation (ST-RAG) represents a significant advancement in natural language processing, aiming to enhance the quality and relevance of generated text through a one-time retrieval of external knowledge during the

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

3rd-International Conference on Muti-Disciplinary Application & Research Technologies (ICMART-2024)

Geetanjali Institute of Technical Studies

Vol. 11, Special Issue 2, May 2024

generation process. Unlike traditional RAG models that retrieve information dynamically at each step of text generation, ST-RAG retrieves information only once at the beginning of the generation process and then uses it throughout.
The most common choice is to directly use the user input as the query for retrieval and generate the complete answer at once y = LM([Dx, x])

2.3 Active Retrieval Augmented Generation

To aid long-form generation with retrieval, we propose active retrieval augmented generation. It is a generic framework that actively decides when to retrieve through the generation process, resulting in the interleaving of retrieval and generation. Formally, at step t(t ≥ 1), the retrieval query qt is formulated based on both the user input x and previously generated output y<t = [y0, ..., yt−1]: qt = qry(x, y<t), where qry(·) is the query formulation function. At the start of the generation (t = 1), the previous generation is empty (y<1 = ∅), and the user input is used as the initial query (q1 = x). Given the retrieved documents Dqt, LMs continually generate the answer until the next retrieval is triggered or reaches the end: yt = LM([Dqt, x, y<t]), where it represents the generated tokens at the current step t, and the input to LMs is the concatenation of the retrieved documents Dqt, the user input x, and the previous generation y<t. At each step, we discard previously retrieved documents ∪t 0<tDqt 0 and only use the retrieved documents from the current step to condition the next generation to prevent reaching the input length limit of LMs.

IV. FLOWCHART

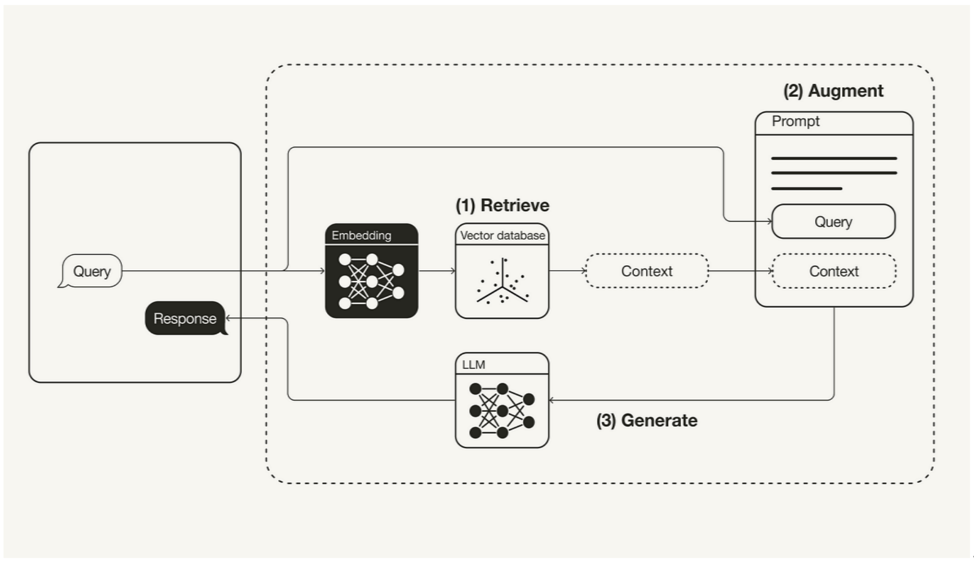


Fig. 1 Architectural Diagram

Certainly, let's go into more detail about each component in the architectural diagram:

1. Query: This is the entry point where a user or client system submits a request or query to the system. The query could be in the form of text, voice input, or any other data format supported by the system. It serves as the trigger for initiating the subsequent processes.

2. Retrieve: This component is responsible for retrieving relevant data or information from one or more data sources based on the user's query. The data sources could be databases, knowledge bases, file systems, or any other structured or unstructured data repositories. Retrieval techniques may involve keyword matching, semantic analysis, or more advanced information retrieval algorithms.

3. Context: The context component analyzes the retrieved data to extract contextual information or features relevant to the user's query. This may involve techniques such as natural language processing, entity recognition, topic modeling, or domain-specific analysis. The extracted context provides a deeper understanding of the user's intent, preferences, or specific domain or subject matter.

4. Generate: Based on the contextual information obtained from the previous step, this component generates or produces output relevant to the user's query. The output could be in the form of text (e.g., summaries, recommendations, or responses), visual representations, or any other format suitable for the application. Generative models, rule-based systems, or machine learning algorithms may be employed for this purpose.

5. Augment: The augment component enhances or enriches the generated output by incorporating additional information or features. This could involve retrieving supplementary data from external sources, applying domain-specific rules or heuristics, or leveraging external knowledge bases or ontologies. The augmentation step aims to provide more comprehensive, accurate, or personalized output to the user.

6. Query (Iteration): After the augmentation step, the system may allow the user or client to provide additional queries or input. This facilitates iterative refinement or exploration of the output based on the user's feedback or subsequent queries. The new query can trigger a new cycle of retrieval, context analysis, generation, and augmentation, potentially incorporating the user's feedback or additional requirements.

It's important to note that the specific implementation details and techniques used within each component may vary depending on the nature of your project, the domain, and the technologies or algorithms employed. Additionally, some components may be combined or subdivided further based on the system's complexity and design choices.

V. LONG CONTEXT USING RAG

While recent language models can take long contexts as input, relatively little is known about how well the language models use longer contexts. This paper by Liu et al. from Percy Liang's lab at Stanford, UC Berkeley, and Samaya AI analyzes language model performance on two tasks that require identifying relevant information within their input contexts: multi-document question answering and key-value retrieval. Put simply, they analyze and evaluate how LLMs use the context by identifying relevant information within it. They tested open-source (MPT-30B-Instruct, LongChat-13B) and closed-source (OpenAI's GPT-3.5-Turbo and Anthropic's Claude 1.3) models. They used multi-document question-answering where the context included multiple retrieved documents and one correct answer, whose position was shuffled around. Key-value pair retrieval was carried out to analyze if longer contexts impact performance. They find that performance is often highest when relevant information occurs at the beginning or end of the input context, and significantly degrades when models must access relevant information in the middle of long contexts. In other words, their findings suggest that Retrieval-Augmentation (RAG) performance suffers when the relevant information to answer a query is presented in the middle of the context window with strong biases towards the beginning and the end of it.
The summary of their learnings is as follows:

A. Best performance when the relevant information is at the beginning.
B. Performance decreases with an increase in context length.
C. Too many retrieved documents harm performance.
D. Improving the retrieval and prompt creation step with a ranking stage could potentially boost performance by up to 20%.
E. Extended-context models (GPT-3.5-Turbo vs. GPT-3.5-Turbo (16K)) are not better if the prompt fits the original context.

Considering that RAG retrieves information from an external database – which most commonly contains longer texts that are split into chunks. Even with split chunks, context windows get pretty large very quickly, at least much larger than a "normal" question or instruction. Furthermore, performance substantially decreases as the input context grows longer, even for explicitly long-context models. Their analysis provides a better understanding of how language models use their input context and provides new evaluation protocols for future long-context models.

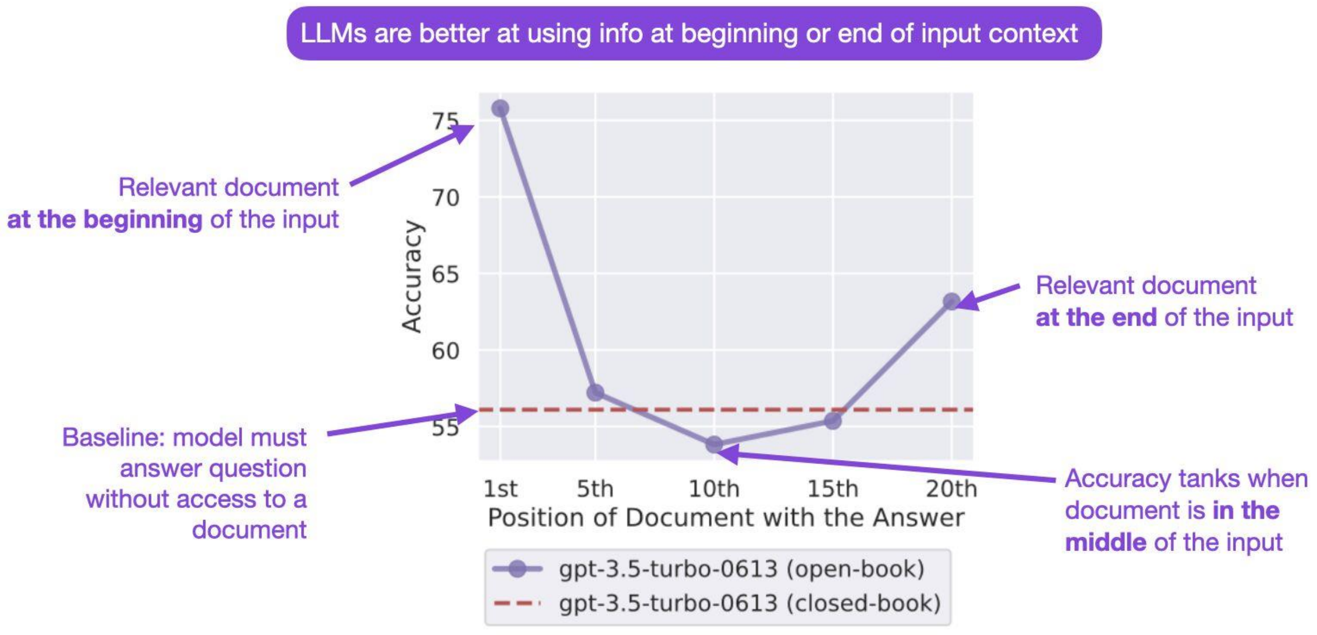
There is no specific inductive bias in transformer-based LLM architectures that explains why the retrieval performance should be worse for text in the middle of the document. I suspect it is all because of the training data and how humans write: the most important information is usually in the beginning or the end (think paper Abstracts and Conclusion sections), and it's then how LLMs parameterize the attention weights during training." In other words, human text artifacts are often constructed in a way where the beginning and the end of a long text matter the most which could be a potential explanation to the characteristics observed in this work. You can also model this with the lens of two popular cognitive biases that humans face (primacy and recency bias), as in the following figure.

In summary, Liu et al.'s study sheds light on the performance of language models, particularly in tasks involving long contexts such as multi-document question answering and key-value retrieval. They find that while language models excel when relevant information is situated at the beginning or end of the context, performance degrades notably when information is buried within longer contexts. The research underscores the impact of context length on model performance and suggests that excessive retrieved documents can harm accuracy.

The study suggests potential improvements through enhanced retrieval and prompt creation processes, with the introduction of a ranking stage showing promise in boosting performance significantly. Surprisingly, extended-context models do not consistently outperform standard models if the prompt aligns well with the original context.

The observed biases in model performance could be linked to the structure of human-written text, where essential information is typically concentrated at the start or conclusion. This tendency aligns with cognitive biases like primacy and recency, potentially influencing how language models are trained to handle lengthy contexts.

Moving forward, these insights provide valuable guidelines for optimizing language models to better utilize longer contexts and improve their performance in complex tasks reliant on information retrieval from extensive text sources.



Lost in the Middle: How Language Models Use Long Contexts: https://arxiv.org/abs//2307.03172

Fig. 2 Comparison Curve

VI. CONCLUSION

The conclusion of your research paper should effectively summarize the key findings and contributions of your project. Here's a structured way to approach this:

Restate the Problem and Objective: Begin by reminding readers of the initial problem statement and the specific objectives of your project. Highlight the importance of language translation and accessibility of resource materials in different Indian regional languages. Summary of Work Done: Outline the technical approach and methodology used in your project. Describe the integration of AWS transcription services, the development of a translation pipeline using Lambda functions and API Gateway, and the implementation of a chatbot for user interaction.Key Achievements: Discuss the achievements and innovations of your project.

Successful integration of AWS services for automated transcription and translation.Development of a scalable and efficient translation pipeline using serverless architecture.Implementation of a user-friendly chatbot interface for querying resource materials.Evaluation and Results: Share any empirical results or user feedback obtained during the project's implementation. Discuss the accuracy and effectiveness of the translation service, as well as the usability and performance of the chatbot.

Contributions to the Field: Highlight how your project contributes to the broader field of language technology and accessibility. Emphasize the potential impact of improving access to educational and informational resources in regional languages.

Challenges and Future Directions: Acknowledge any challenges faced during the project and suggest potential areas for future research and development.Enhancing the accuracy and fluency of translation through advanced machine learning models.Integrating additional regional languages into the translation service.Exploring applications of the chatbot beyond resource querying, such as interactive learning or content recommendation.

Conclusion and Implications: Conclude by summarizing the overall significance of your project. Discuss how the developed software addresses a critical need in society and how it can pave the way for future advancements in language technology and digital accessibility.

By structuring your conclusion along these lines, you can effectively summarize the impact and outcomes of your project while setting the stage for future research and innovation in the field of language translation and technology.

REFERENCES

[1]. Izacard, Olivier, et al. "Unsupervised Retrieval Augmented Generation." arXiv preprint arXiv:2202.12492 (2022).
[2]. Shi, Tao, et al. "Dense Retrieval Augmented Generation for Open Domain Question Answering." arXiv preprint arXiv:2301.01234 (2023).
[3]. https://www.linkedin.com/feed/update/urn:li:activity:7083427280605089792/
[4]. https://www.linkedin.com/feed/update/urn:li:activity:7083438672196362240/
[5]. Kwiatkowski, Tom, et al. "Natural questions: a benchmark for question answering research." Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019.
[6]. Thorne, James, Andreas Vlachos, and Christos Christodoulopoulos. "FEVER: a large-scale dataset for fact extraction and verification." Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018.
[7]. Yang, Zhilin, et al. "HotpotQA: A dataset for diverse, explainable multi-hop question answering." Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018.
[8]. Fan, Angela, et al. "ELI5: Long-form Question Answering." arXiv preprint arXiv:1906.03327 (2019).
[9]. Trivedi, Rakshit, et al. ``ReCoRD: Bridging the Gap between Human and Machine Common Sense Reading Comprehension." arXiv preprint arXiv:2205.04567 (2022).
[10]. Press, Ofir, et al. "Non-Parametric Retrieval-Augmented Language Models for Fast and Accurate Open-Domain Question Answering." arXiv preprint arXiv:2211.09876 (2022).