Retrieval-Augmented Generation for Large Language Models: A Survey

Yunfan Gao[a], Yun Xiong[b], Xinyu Gao[b], Kangxiang Jia[b], Jinliu Pan[b], Yuxi Bi[c], Yi Dai[a], Jiawei Sun[a], Meng Wang[c], and Haofen Wang [a,c]

[a]Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University
[b]Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
[c]College of Design and Innovation, Tongji University

Abstract—**Large Language Models (LLMs) showcase impressive capabilities but encounter challenges like hallucination, outdated knowledge, and non-transparent, untraceable reasoning processes. Retrieval-Augmented Generation (RAG) has emerged as a promising solution by incorporating knowledge from external databases. This enhances the accuracy and credibility of the generation, particularly for knowledge-intensive tasks, and allows for continuous knowledge updates and integration of domain-specific information. RAG synergistically merges LLMs' intrinsic knowledge with the vast, dynamic repositories of external databases. This comprehensive review paper offers a detailed examination of the progression of RAG paradigms, encompassing the Naive RAG, the Advanced RAG, and the Modular RAG. It meticulously scrutinizes the tripartite foundation of RAG frameworks, which includes the retrieval, the generation and the augmentation techniques. The paper highlights the state-of-the-art technologies embedded in each of these critical components, providing a profound understanding of the advancements in RAG systems. Furthermore, this paper introduces up-to-date evaluation framework and benchmark. At the end, this article delineates the challenges currently faced and points out prospective avenues for research and development [1].**

Index Terms—**Large language model, retrieval-augmented generation, natural language processing, information retrieval**

I. Introduction

Large language models (LLMs) have achieved remarkable success, though they still face significant limitations, especially in domain-specific or knowledge-intensive tasks [1], notably producing "hallucinations" [2] when handling queries beyond their training data or requiring current information. To overcome challenges, Retrieval-Augmented Generation (RAG) enhances LLMs by retrieving relevant document chunks from external knowledge base through semantic similarity calculation. By referencing external knowledge, RAG effectively reduces the problem of generating factually incorrect content. Its integration into LLMs has resulted in widespread adoption, establishing RAG as a key technology in advancing chatbots and enhancing the suitability of LLMs for real-world applications.

RAG technology has rapidly developed in recent years, and the technology tree summarizing related research is shown in Figure 1. The development trajectory of RAG in the era of large models exhibits several distinct stage characteristics. Initially, RAG's inception coincided with the rise of the Transformer architecture, focusing on enhancing language models by incorporating additional knowledge through Pre-Training Models (PTM). This early stage was characterized by foundational work aimed at refining pre-training techniques [3]–[5].The subsequent arrival of ChatGPT [6] marked a pivotal moment, with LLM demonstrating powerful in context learning (ICL) capabilities. RAG research shifted towards providing better information for LLMs to answer more complex and knowledge-intensive tasks during the inference stage, leading to rapid development in RAG studies. As research progressed, the enhancement of RAG was no longer limited to the inference stage but began to incorporate more with LLM fine-tuning techniques.

The burgeoning field of RAG has experienced swift growth, yet it has not been accompanied by a systematic synthesis that could clarify its broader trajectory. This survey endeavors to fill this gap by mapping out the RAG process and charting its evolution and anticipated future paths, with a focus on the integration of RAG within LLMs. This paper considers both technical paradigms and research methods, summarizing three main research paradigms from over 100 RAG studies, and analyzing key technologies in the core stages of "Retrieval," "Generation," and "Augmentation." On the other hand, current research tends to focus more on methods, lacking analysis and summarization of how to evaluate RAG. This paper comprehensively reviews the downstream tasks, datasets, benchmarks, and evaluation methods applicable to RAG. Overall, this paper sets out to meticulously compile and categorize the foundational technical concepts, historical progression, and the spectrum of RAG methodologies and applications that have emerged post-LLMs. It is designed to equip readers and professionals with a detailed and structured understanding of both large models and RAG. It aims to illuminate the evolution of retrieval augmentation techniques, assess the strengths and weaknesses of various approaches in their respective contexts, and speculate on upcoming trends and innovations.

Our contributions are as follows:

- In this survey, we present a thorough and systematic review of the state-of-the-art RAG methods, delineating its evolution through paradigms including naive RAG,

Corresponding Author.Email:haofen.wang@tongji.edu.cn

[1] Resources are available at https://github.com/Tongji-KGLLM/RAG-Survey

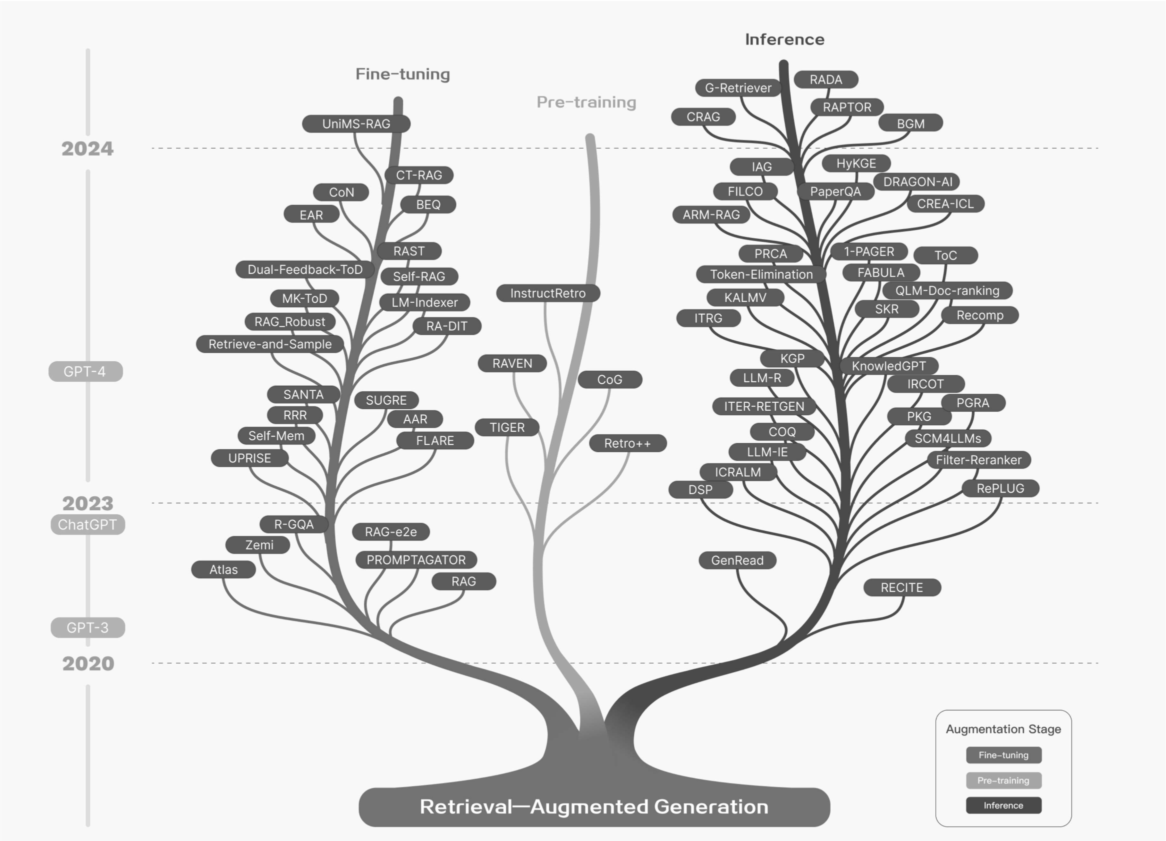


Fig. 1. Technology tree of RAG research. The stages of involving RAG mainly include pre-training, fine-tuning, and inference. With the emergence of LLMs, research on RAG initially focused on leveraging the powerful in context learning abilities of LLMs, primarily concentrating on the inference stage. Subsequent research has delved deeper, gradually integrating more with the fine-tuning of LLMs. Researchers have also been exploring ways to enhance language models in the pre-training stage through retrieval-augmented techniques.

advanced RAG, and modular RAG. This review contextualizes the broader scope of RAG research within the landscape of LLMs.

- We identify and discuss the central technologies integral to the RAG process, specifically focusing on the aspects of “Retrieval”, “Generation” and “Augmentation”, and delve into their synergies, elucidating how these components intricately collaborate to form a cohesive and effective RAG framework.
- We have summarized the current assessment methods of RAG, covering 26 tasks, nearly 50 datasets, outlining the evaluation objectives and metrics, as well as the current evaluation benchmarks and tools. Additionally, we anticipate future directions for RAG, emphasizing potential enhancements to tackle current challenges.

The paper unfolds as follows: Section II introduces the main concept and current paradigms of RAG. The following three sections explore core components—“Retrieval”, “Generation” and “Augmentation”, respectively. Section III focuses on optimization methods in retrieval,including indexing, query and embedding optimization. Section IV concentrates on post-retrieval process and LLM fine-tuning in generation. Section V analyzes the three augmentation processes. Section VI focuses on RAG’s downstream tasks and evaluation system. Section VII mainly discusses the challenges that RAG currently faces and its future development directions. At last, the paper concludes in Section VIII.

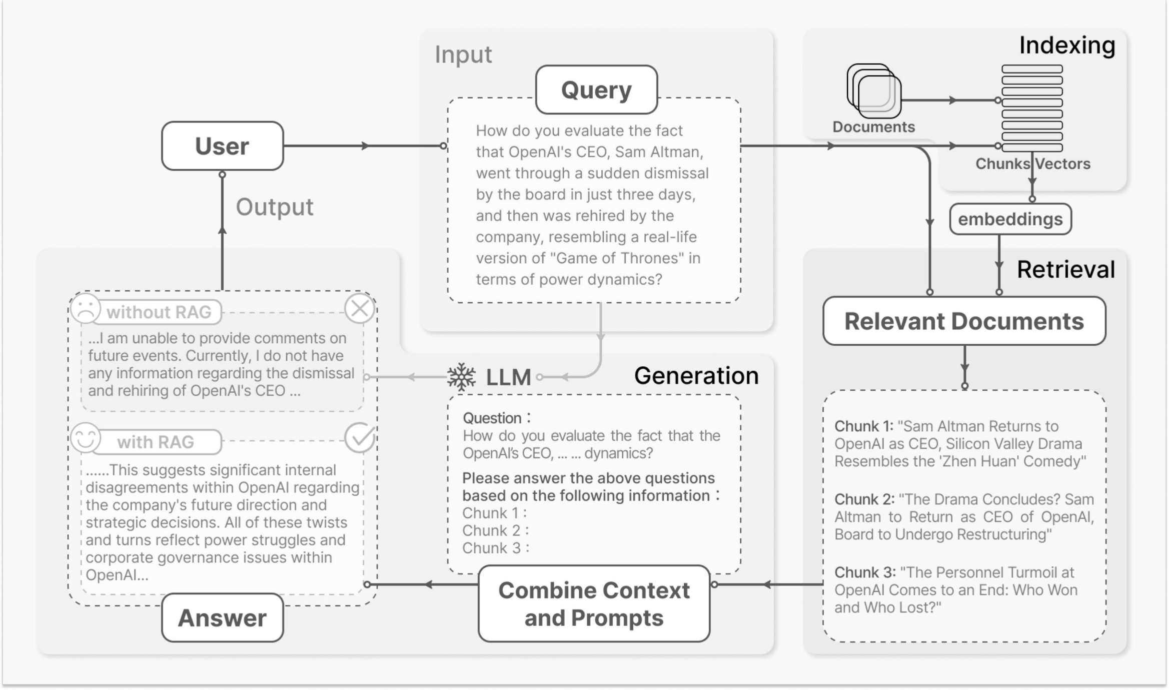
II. Overview of RAG

A typical application of RAG is illustrated in Figure 2. Here, a user poses a question to ChatGPT about a recent, widely discussed news. Given ChatGPT’s reliance on pretraining data, it initially lacks the capacity to provide updates on recent developments. RAG bridges this information gap by sourcing and incorporating knowledge from external databases. In this case, it gathers relevant news articles related to the user’s query. These articles, combined with the original question, form a comprehensive prompt that empowers LLMs to generate a well-informed answer.

The RAG research paradigm is continuously evolving, and we categorize it into three stages: Naive RAG, Advanced RAG, and Modular RAG, as showed in Figure 3. Despite RAG method are cost-effective and surpass the performance of the native LLM, they also exhibit several limitations. The development of Advanced RAG and Modular RAG is a response to these specific shortcomings in Naive RAG.

A. Naive RAG

The Naive RAG research paradigm represents the earliest methodology, which gained prominence shortly after the



How do you evaluate the fact that OpenAI's CEO, Sam Altman, went through a sudden dismissal by the board in just three days, and then was rehired by the company, resembling a real-life version of "Game of Thrones" in terms of power dynamics?

...I am unable to provide comments on future events. Currently, I do not have any information regarding the dismissal and rehiring of OpenAI's CEO ...

......This suggests significant internal disagreements within OpenAI regarding the company's future direction and strategic decisions. All of these twists and turns reflect power struggles and corporate governance issues within OpenAI...

Question：
How do you evaluate the fact that the OpenAI's CEO, ... dynamics?
Please answer the above questions based on the following information：
Chunk 1 :
Chunk 2 :
Chunk 3 :

Chunk 1: "Sam Altman Returns to OpenAI as CEO, Silicon Valley Drama Resembles the 'Zhen Huan' Comedy"

Chunk 2: "The Drama Concludes? Sam Altman to Return as CEO of OpenAI, Board to Undergo Restructuring"

Chunk 3: "The Personnel Turmoil at OpenAI Comes to an End: Who Won and Who Lost?"

Fig. 2. A representative instance of the RAG process applied to question answering. It mainly consists of 3 steps. 1) Indexing. Documents are split into chunks, encoded into vectors, and stored in a vector database. 2) Retrieval. Retrieve the Top k chunks most relevant to the question based on semantic similarity. 3) Generation. Input the original question and the retrieved chunks together into LLM to generate the final answer.

widespread adoption of ChatGPT. The Naive RAG follows a traditional process that includes indexing, retrieval, and generation, which is also characterized as a "Retrieve-Read" framework [7].

Indexing starts with the cleaning and extraction of raw data in diverse formats like PDF, HTML, Word, and Markdown, which is then converted into a uniform plain text format. To accommodate the context limitations of language models, text is segmented into smaller, digestible chunks. Chunks are then encoded into vector representations using an embedding model and stored in vector database. This step is crucial for enabling efficient similarity searches in the subsequent retrieval phase.

Retrieval. Upon receipt of a user query, the RAG system employs the same encoding model utilized during the indexing phase to transform the query into a vector representation. It then computes the similarity scores between the query vector and the vector of chunks within the indexed corpus. The system prioritizes and retrieves the top K chunks that demonstrate the greatest similarity to the query. These chunks are subsequently used as the expanded context in prompt.

Generation. The posed query and selected documents are synthesized into a coherent prompt to which a large language model is tasked with formulating a response. The model's approach to answering may vary depending on task-specific criteria, allowing it to either draw upon its inherent parametric knowledge or restrict its responses to the information contained within the provided documents. In cases of ongoing dialogues, any existing conversational history can be integrated into the prompt, enabling the model to engage in multi-turn dialogue interactions effectively.

However, Naive RAG encounters notable drawbacks:

Retrieval Challenges. The retrieval phase often struggles with precision and recall, leading to the selection of misaligned or irrelevant chunks, and the missing of crucial information.

Generation Difficulties. In generating responses, the model may face the issue of hallucination, where it produces content not supported by the retrieved context. This phase can also suffer from irrelevance, toxicity, or bias in the outputs, detracting from the quality and reliability of the responses.

Augmentation Hurdles. Integrating retrieved information with the different task can be challenging, sometimes resulting in disjointed or incoherent outputs. The process may also encounter redundancy when similar information is retrieved from multiple sources, leading to repetitive responses. Determining the significance and relevance of various passages and ensuring stylistic and tonal consistency add further complexity. Facing complex issues, a single retrieval based on the original query may not suffice to acquire adequate context information.

Moreover, there's a concern that generation models might overly rely on augmented information, leading to outputs that simply echo retrieved content without adding insightful or synthesized information.

B. *Advanced RAG*

Advanced RAG introduces specific improvements to overcome the limitations of Naive RAG. Focusing on enhancing retrieval quality, it employs pre-retrieval and post-retrieval strategies. To tackle the indexing issues, Advanced RAG refines its indexing techniques through the use of a sliding window approach, fine-grained segmentation, and the incorporation of metadata. Additionally, it incorporates several optimization methods to streamline the retrieval process [8].

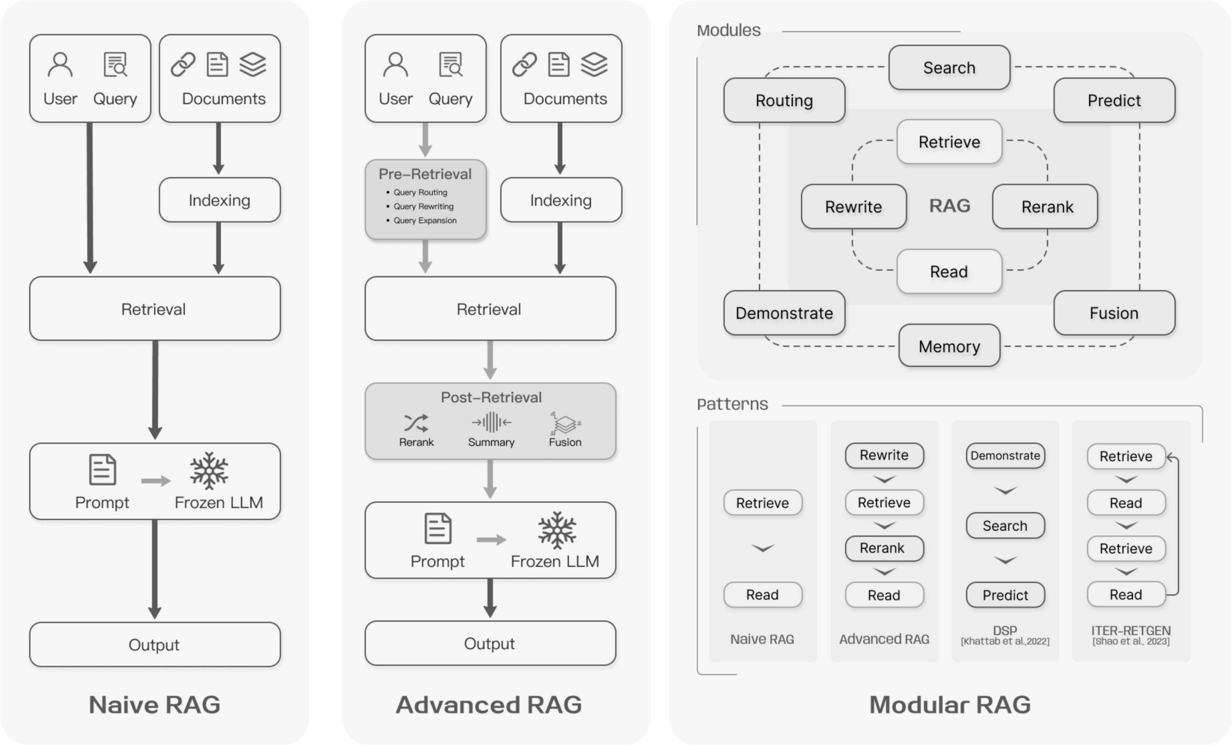


Fig. 3. Comparison between the three paradigms of RAG. (Left) Naive RAG mainly consists of three parts: indexing, retrieval and generation. (Middle) Advanced RAG proposes multiple optimization strategies around pre-retrieval and post-retrieval, with a process similar to the Naive RAG, still following a chain-like structure. (Right) Modular RAG inherits and develops from the previous paradigm, showcasing greater flexibility overall. This is evident in the introduction of multiple specific functional modules and the replacement of existing modules. The overall process is not limited to sequential retrieval and generation; it includes methods such as iterative and adaptive retrieval.

Pre-retrieval process. In this stage, the primary focus is on optimizing the indexing structure and the original query. The goal of optimizing indexing is to enhance the quality of the content being indexed. This involves strategies: enhancing data granularity, optimizing index structures, adding metadata, alignment optimization, and mixed retrieval. While the goal of query optimization is to make the user's original question clearer and more suitable for the retrieval task. Common methods include query rewriting query transformation, query expansion and other techniques [7], [9]–[11].

Post-Retrieval Process. Once relevant context is retrieved, it's crucial to integrate it effectively with the query. The main methods in post-retrieval process include rerank chunks and context compressing. Re-ranking the retrieved information to relocate the most relevant content to the edges of the prompt is a key strategy. This concept has been implemented in frameworks such as LlamaIndex[2], LangChain[3], and HayStack [12]. Feeding all relevant documents directly into LLMs can lead to information overload, diluting the focus on key details with irrelevant content.To mitigate this, post-retrieval efforts concentrate on selecting the essential information, emphasizing critical sections, and shortening the context to be processed.

C. Modular RAG

The modular RAG architecture advances beyond the former two RAG paradigms, offering enhanced adaptability and versatility. It incorporates diverse strategies for improving its components, such as adding a search module for similarity searches and refining the retriever through fine-tuning. Innovations like restructured RAG modules [13] and rearranged RAG pipelines [14] have been introduced to tackle specific challenges. The shift towards a modular RAG approach is becoming prevalent, supporting both sequential processing and integrated end-to-end training across its components. Despite its distinctiveness, Modular RAG builds upon the foundational principles of Advanced and Naive RAG, illustrating a progression and refinement within the RAG family.

1) New Modules: The Modular RAG framework introduces additional specialized components to enhance retrieval and processing capabilities. The Search module adapts to specific scenarios, enabling direct searches across various data sources like search engines, databases, and knowledge graphs, using LLM-generated code and query languages [15]. RAG-Fusion addresses traditional search limitations by employing a multi-query strategy that expands user queries into diverse perspectives, utilizing parallel vector searches and intelligent re-ranking to uncover both explicit and transformative knowledge [16]. The Memory module leverages the LLM's memory to guide retrieval, creating an unbounded memory pool that

[2] https://www.llamaindex.ai
[3] https://www.langchain.com/

aligns the text more closely with data distribution through iterative self-enhancement [17], [18]. Routing in the RAG system navigates through diverse data sources, selecting the optimal pathway for a query, whether it involves summarization, specific database searches, or merging different information streams [19]. The Predict module aims to reduce redundancy and noise by generating context directly through the LLM, ensuring relevance and accuracy [13]. Lastly, the Task Adapter module tailors RAG to various downstream tasks, automating prompt retrieval for zero-shot inputs and creating task-specific retrievers through few-shot query generation [20], [21] .This comprehensive approach not only streamlines the retrieval process but also significantly improves the quality and relevance of the information retrieved, catering to a wide array of tasks and queries with enhanced precision and flexibility.

2) New Patterns: Modular RAG offers remarkable adaptability by allowing module substitution or reconfiguration to address specific challenges. This goes beyond the fixed structures of Naive and Advanced RAG, characterized by a simple "Retrieve" and "Read" mechanism. Moreover, Modular RAG expands this flexibility by integrating new modules or adjusting interaction flow among existing ones, enhancing its applicability across different tasks.

Innovations such as the Rewrite-Retrieve-Read [7]model leverage the LLM's capabilities to refine retrieval queries through a rewriting module and a LM-feedback mechanism to update rewriting model., improving task performance. Similarly, approaches like Generate-Read [13] replace traditional retrieval with LLM-generated content, while Recite-Read [22] emphasizes retrieval from model weights, enhancing the model's ability to handle knowledge-intensive tasks. Hybrid retrieval strategies integrate keyword, semantic, and vector searches to cater to diverse queries. Additionally, employing sub-queries and hypothetical document embeddings (HyDE) [11] seeks to improve retrieval relevance by focusing on embedding similarities between generated answers and real documents.

Adjustments in module arrangement and interaction, such as the Demonstrate-Search-Predict (DSP) [23] framework and the iterative Retrieve-Read-Retrieve-Read flow of ITER-RETGEN [14], showcase the dynamic use of module outputs to bolster another module's functionality, illustrating a sophisticated understanding of enhancing module synergy. The flexible orchestration of Modular RAG Flow showcases the benefits of adaptive retrieval through techniques such as FLARE [24] and Self-RAG [25]. This approach transcends the fixed RAG retrieval process by evaluating the necessity of retrieval based on different scenarios. Another benefit of a flexible architecture is that the RAG system can more easily integrate with other technologies (such as fine-tuning or reinforcement learning) [26]. For example, this can involve fine-tuning the retriever for better retrieval results, fine-tuning the generator for more personalized outputs, or engaging in collaborative fine-tuning [27].

D. RAG vs Fine-tuning

The augmentation of LLMs has attracted considerable attention due to their growing prevalence. Among the optimization methods for LLMs, RAG is often compared with Fine-tuning (FT) and prompt engineering. Each method has distinct characteristics as illustrated in Figure 4. We used a quadrant chart to illustrate the differences among three methods in two dimensions: external knowledge requirements and model adaption requirements. Prompt engineering leverages a model's inherent capabilities with minimum necessity for external knowledge and model adaption. RAG can be likened to providing a model with a tailored textbook for information retrieval, ideal for precise information retrieval tasks. In contrast, FT is comparable to a student internalizing knowledge over time, suitable for scenarios requiring replication of specific structures, styles, or formats.

RAG excels in dynamic environments by offering real-time knowledge updates and effective utilization of external knowledge sources with high interpretability. However, it comes with higher latency and ethical considerations regarding data retrieval. On the other hand, FT is more static, requiring retraining for updates but enabling deep customization of the model's behavior and style. It demands significant computational resources for dataset preparation and training, and while it can reduce hallucinations, it may face challenges with unfamiliar data.

In multiple evaluations of their performance on various knowledge-intensive tasks across different topics, [28] revealed that while unsupervised fine-tuning shows some improvement, RAG consistently outperforms it, for both existing knowledge encountered during training and entirely new knowledge. Additionally, it was found that LLMs struggle to learn new factual information through unsupervised fine-tuning. The choice between RAG and FT depends on the specific needs for data dynamics, customization, and computational capabilities in the application context. RAG and FT are not mutually exclusive and can complement each other, enhancing a model's capabilities at different levels. In some instances, their combined use may lead to optimal performance. The optimization process involving RAG and FT may require multiple iterations to achieve satisfactory results.

III. Retrieval

In the context of RAG, it is crucial to efficiently retrieve relevant documents from the data source. There are several key issues involved, such as the retrieval source, retrieval granularity, pre-processing of the retrieval, and selection of the corresponding embedding model.

A. Retrieval Source

RAG relies on external knowledge to enhance LLMs, while the type of retrieval source and the granularity of retrieval units both affect the final generation results.

1) Data Structure: Initially, text is s the mainstream source of retrieval. Subsequently, the retrieval source expanded to include semi-structured data (PDF) and structured data (Knowledge Graph, KG) for enhancement. In addition to retrieving from original external sources, there is also a growing trend in recent researches towards utilizing content generated by LLMs themselves for retrieval and enhancement purposes.