Retrieval-Augmented Generation for Large Language Models: A Survey

Yunfan Gao [1], **Yun Xiong** [2], **Xinyu Gao** [2], **Kangxiang Jia** [2], **Jinliu Pan** [2], **Yuxi Bi** [3], **Yi Dai**[1], **Jiawei Sun**[1], **Qianyu Guo**[4], **Meng Wang** [3] and **Haofen Wang** [1,3] *

[1] Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University
[2] Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
[3] College of Design and Innovation, Tongji University
[4] School of Computer Science, Fudan University

Abstract

Large Language Models (LLMs) demonstrate significant capabilities but face challenges such as hallucination, outdated knowledge, and non-transparent, untraceable reasoning processes.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 models, 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 the metrics and benchmarks for assessing RAG models, along with the most up-to-date evaluation framework. In conclusion, the paper delineates prospective avenues for research, including the identification of challenges, the expansion of multi-modalities, and the progression of the RAG infrastructure and its ecosystem. [1].

1 Introduction

Large language models (LLMs) such as the GPT series [Brown *et al.*, 2020, OpenAI, 2023] and the LLama series [Touvron *et al.*, 2023], along with other models like Gemini [Google, 2023], have achieved remarkable success in natural language processing, demonstrating superior performance on various benchmarks including SuperGLUE [Wang *et al.*, 2019], MMLU [Hendrycks *et al.*, 2020], and BIG-bench [Srivastava *et al.*, 2022]. Despite these advancements, LLMs exhibit notable limitations, particularly in handling domain-specific or highly specialized queries [Kandpal *et al.*, 2023]. A common issue is the generation of incorrect information, or "hallucinations" [Zhang *et al.*, 2023b], especially when queries extend beyond the model's training data or necessitate up-to-date information. These shortcomings underscore the impracticality of deploying LLMs as black-box solutions in real-world production environments without additional safeguards. One promising approach to mitigate these limitations is Retrieval-Augmented Generation (RAG), which integrates external data retrieval into the generative process, thereby enhancing the model's ability to provide accurate and relevant responses.

RAG, introduced by Lewis et al. [Lewis *et al.*, 2020] in mid-2020, stands as a paradigm within the realm of LLMs, enhancing generative tasks. Specifically, RAG involves an initial retrieval step where the LLMs query an external data source to obtain relevant information before proceeding to answer questions or generate text. This process not only informs the subsequent generation phase but also ensures that the responses are grounded in retrieved evidence, thereby significantly enhancing the accuracy and relevance of the output. The dynamic retrieval of information from knowledge bases during the inference phase allows RAG to address issues such as the generation of factually incorrect content, commonly referred to as "hallucinations." The integration of RAG into LLMs has seen rapid adoption and has become a pivotal technology in refining the capabilities of chatbots and rendering LLMs more viable for practical applications.

The evolutionary trajectory of RAG unfolds across four distinctive phases, as illustrated in Figure 1. In its inception in 2017, aligned with the emergence of the Transformer architecture, the primary thrust was on assimilating additional knowledge through Pre-Training Models (PTM) to augment language models. This epoch witnessed RAG's foundational efforts predominantly directed at optimizing pre-training methodologies.

Following this initial phase, a period of relative dormancy ensued before the advent of chatGPT, during which there was minimal advancement in related research for RAG. The subsequent arrival of chatGPT marked a pivotal moment in the

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

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

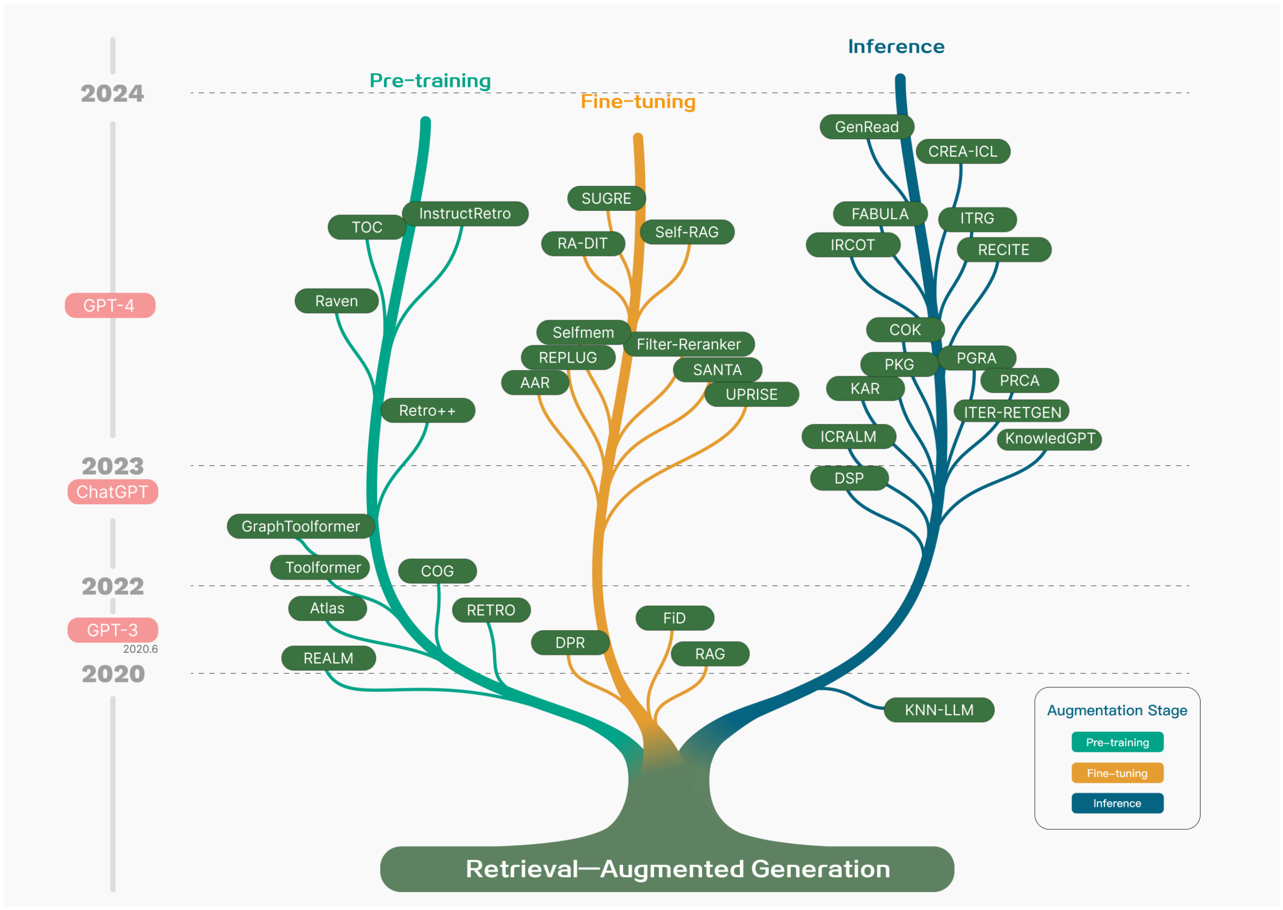

Figure 1: Technology tree of RAG research development featuring representative works

trajectory, propelling LLMs into the forefront. The community's focal point shifted towards harnessing the capabilities of LLMs to attain heightened controllability and address evolving requirements. Consequently, the lion's share of RAG endeavors concentrated on inference, with a minority dedicated to fine-tuning processes. As LLM capabilities continued to advance, especially with the introduction of GPT-4, the landscape of RAG technology underwent a significant transformation. The emphasis evolved into a hybrid approach, combining the strengths of RAG and fine-tuning, alongside a dedicated minority continuing the focus on optimizing pre-training methodologies.

Despite the rapid growth of RAG research, there has been a lack of systematic consolidation and abstraction in the field, which poses challenges in understanding the comprehensive landscape of RAG advancements. This survey aims to outline the entire RAG process and encompass the current and future directions of RAG research, by providing a thorough examination of retrieval augmentation in LLMs.

Therefore, this paper aims to comprehensively summarize and organize the technical principles, developmental history, content, and, in particular, the relevant methods and applications after the emergence of LLMs, as well as the evaluation methods and application scenarios of RAG. It seeks to provide a comprehensive overview and analysis of existing RAG technologies and offer conclusions and prospects for future development methods. This survey intends to furnish readers and practitioners with a thorough and systematic comprehension of large models and RAG, elucidate the progression and key technologies of retrieval augmentation, clarify the merits and limitations of various technologies along with their suitable contexts, and forecast potential future developments.

Our contributions are as follows:

- We present a thorough and systematic review of the state-of-the-art RAG, delineating its evolution through paradigms including naive RAG, 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", "Generator" and "Augmentation", and delve into their synergies, elucidating how these components intricately collaborate to form a cohesive and effective RAG framework.
- We construct a thorough evaluation framework for RAG, outlining the evaluation objectives and metrics. Our comparative analysis clarifies the strengths and weaknesses of RAG compared to fine-tuning from various

perspectives. Additionally, we anticipate future directions for RAG, emphasizing potential enhancements to tackle current challenges, expansions into multi-modal settings, and the development of its ecosystem.

The paper unfolds as follows: Section 2 and 3 define RAG and detail its developmental process. Section 4 through 6 explore core components—Retrieval, "Generation" and "Augmentation"—highlighting diverse embedded technologies. Section 7 focuses on RAG's evaluation system. Section 8 compare RAG with other LLM optimization methods and suggest potential directions for its evolution. The paper concludes in Section 9.

2 Definition

The definition of RAG can be summarized from its workflow. Figure 2 depicts a typical RAG application workflow. In this scenario, a user inquires ChatGPT about a recent high-profile event (i.e., the abrupt dismissal and reinstatement of OpenAI's CEO) which generated considerable public discourse. ChatGPT as the most renowned and widely utilized LLM, constrained by its pretraining data, lacks knowledge of recent events. RAG addresses this gap by retrieving up-to-date document excerpts from external knowledge bases. In this instance, it procures a selection of news articles pertinent to the inquiry. These articles, alongside the initial question, are then amalgamated into an enriched prompt that enables ChatGPT to synthesize an informed response. This example illustrates the RAG process, demonstrating its capability to enhance the model's responses with real-time information retrieval.

Technologically, RAG has been enriched through various innovative approaches addressing pivotal questions such as "what to retrieve" "when to retrieve" and "how to use the retrieved information". For "what to retrieve" research has progressed from simple token [Khandelwal *et al.*, 2019] and entity retrieval [Nishikawa *et al.*, 2022] to more complex structures like chunks [Ram *et al.*, 2023] and knowledge graph [Kang *et al.*, 2023], with studies focusing on the granularity of retrieval and the level of data structuring. Coarse granularity brings more information but with lower precision. Retrieving structured text provides more information while sacrificing efficiency. The question of "when to retrieve" has led to strategies ranging from single [Wang *et al.*, 2023e, Shi *et al.*, 2023] to adaptive [Jiang *et al.*, 2023b, Huang *et al.*, 2023] and multiple retrieval [Izacard *et al.*, 2022] methods. High frequency of retrieval brings more information and lower efficiency. As for "how to use" the retrieved data, integration techniques have been developed across various levels of the model architecture, including the input [Khattab *et al.*, 2022], intermediate [Borgeaud *et al.*, 2022], and output layers [Liang *et al.*, 2023]. Although the "intermediate" and "output layers" are more effective, there are problems with the need for training and low efficiency.

RAG is a paradigm that enhances LLMs by integrating external knowledge bases. It employs a synergistic approach, combining information retrieval mechanisms and In-Context Learning (ICL) to bolster the LLM's performance. In this framework, a query initiated by a user prompts the retrieval of pertinent information via search algorithms. This information is then woven into the LLM's prompts, providing additional context for the generation process. RAG's key advantage lies in its obviation of the need for retraining of LLMs for task-specific applications. Developers can instead append an external knowledge repository, enriching the input and thereby refining the model's output precision. RAG has become one of the most popular architectures in LLMs' systems, due to its high practicality and low barrier to entry, with many conversational products being built almost entirely on RAG.

The RAG workflow comprises three key steps. First, the corpus is partitioned into discrete chunks, upon which vector indices are constructed utilizing an encoder model. Second, RAG identifies and retrieves chunks based on their vector similarity to the query and indexed chunks. Finally, the model synthesizes a response conditioned on the contextual information gleaned from the retrieved chunks. These steps form the fundamental framework of the RAG process, underpinning its information retrieval and context-aware generation capabilities. Next, we will provide an introduction to the RAG research framework.

3 RAG Framework

The RAG research paradigm is continuously evolving, and this section primarily delineates its progression. We categorize it into three types: Naive RAG, Advanced RAG, and Modular RAG. While RAG were cost-effective and surpassed the performance of the native LLM, they also exhibited several limitations. The development of Advanced RAG and Modular RAG was a response to these specific shortcomings in Naive RAG.

3.1 Naive RAG

The Naive RAG research paradigm represents the earliest methodology, which gained prominence shortly after the widespread adoption of ChatGPT. The Naive RAG follows a traditional process that includes indexing, retrieval, and generation. It is also characterized as a "Retrieve-Read" framework [Ma *et al.*, 2023a].

Indexing

The indexing process is a crucial initial step in data preparation that occurs offline and involves several stages. It begins with data indexing, where original data is cleansed and extracted, and various file formats such as PDF, HTML, Word, and Markdown are converted into standardized plain text. In order to fit within the context limitations of language models, this text is then segmented into smaller, more manageable chunks in a process known as chunking. These chunks are subsequently transformed into vector representations through an embedding model, chosen for its balance between inference efficiency and model size. This facilitates similarity comparisons during the retrieval phase. Finally, an index is created to store these text chunks and their vector embeddings as key-value pairs, which allows for efficient and scalable search capabilities.

Retrieval

Upon receipt of a user query, the system employs the same encoding model utilized during the indexing phase to transcode

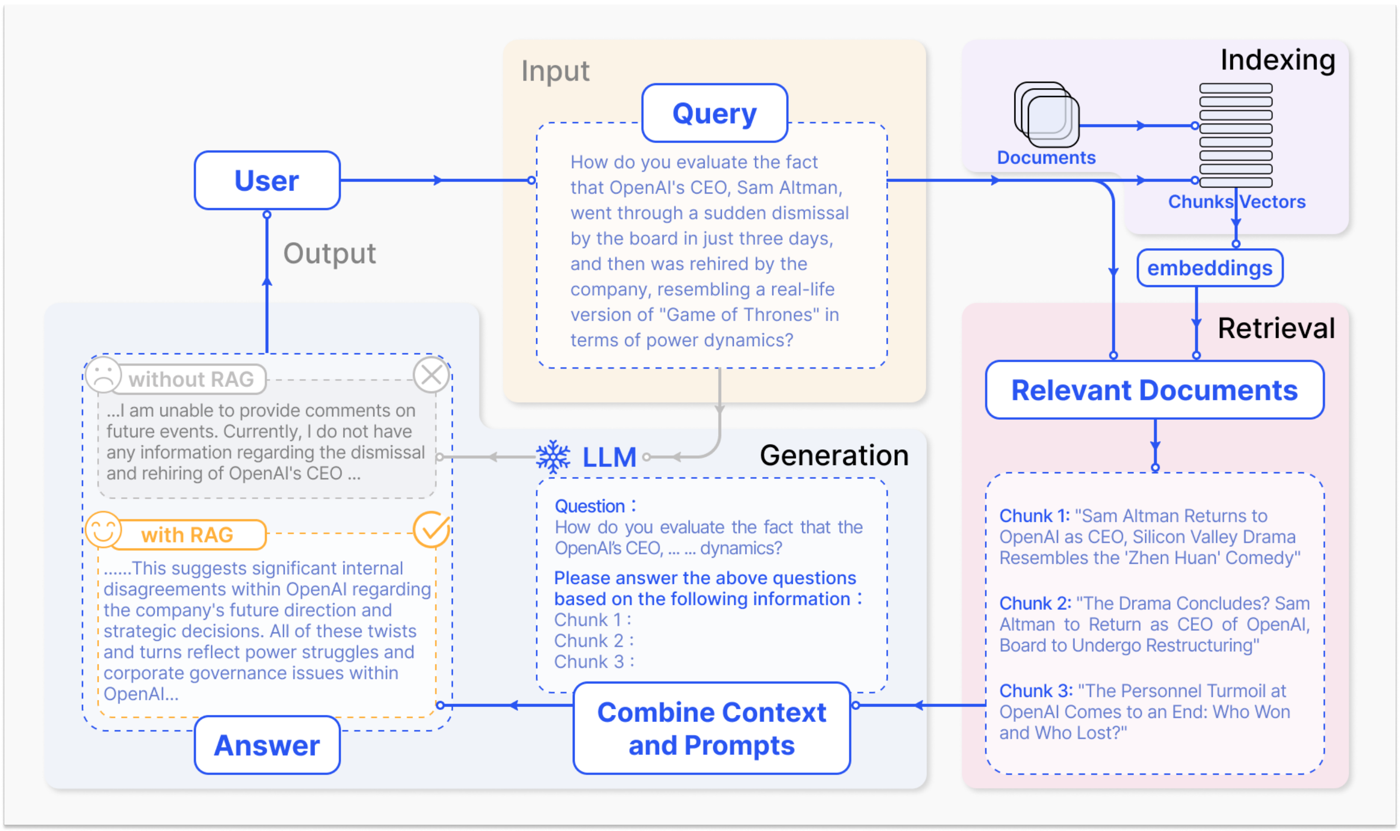
Input
Query
How do you evaluate the fact that OpenAI's CEO, Sam Altman, went through a sudden dismissal by the board in just three days, and then was rehired by the company, resembling a real-life version of "Game of Thrones" in terms of power dynamics?
Indexing
Documents
Chunks Vectors
embeddings
User
Output
Retrieval
Relevant Documents
without RAG
...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 ...
with RAG
......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...
LLM
Generation
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?"
Answer
Combine Context and Prompts

Figure 2: A representative instance of the RAG process applied to question answering

the input into a vector representation. It then proceeds to compute the similarity scores between the query vector and the vectorized 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 contextual basis for addressing the user's request.

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.

Drawbacks in Naive RAG

Naive RAG faces significant challenges in three key areas: "Retrieval," "Generation," and "Augmentation".

Retrieval quality poses diverse challenges, including low precision, leading to misaligned retrieved chunks and potential issues like hallucination or mid-air drop. Low recall also occurs, resulting in the failure to retrieve all relevant chunks, thereby hindering the LLMs' ability to craft comprehensive responses. Outdated information further compounds the problem, potentially yielding inaccurate retrieval results.

Response generation quality presents hallucination challenge, where the model generates answers not grounded in the provided context, as well as issues of irrelevant context and potential toxicity or bias in the model's output.

The augmentation process presents its own challenges in effectively integrating context from retrieved passages with the current generation task, potentially leading to disjointed or incoherent output. Redundancy and repetition are also concerns, especially when multiple retrieved passages contain similar information, resulting in repetitive content in the generated response.

Discerning the importance and relevance of multiple retrieved passages to the generation task is another challenge, requiring the proper balance of each passage's value. Additionally, reconciling differences in writing styles and tones to ensure consistency in the output is crucial.

Lastly, there's a risk of generation models overly depending on augmented information, potentially resulting in outputs that merely reiterate the retrieved content without providing new value or synthesized information.

3.2 Advanced RAG

Advanced RAG has been developed with targeted enhancements to address the shortcomings of Naive RAG. In terms of retrieval quality, Advanced RAG implements pre-retrieval and post-retrieval strategies. To address the indexing challenges experienced by Naive RAG, Advanced RAG has refined its indexing approach using techniques such as sliding window, fine-grained segmentation, and metadata. It has also introduced various methods to optimize the retrieval process [ILIN, 2023].

Pre-Retrieval Process

Optimizing Data Indexing.The goal of optimizing data indexing is to enhance the quality of the content being indexed. This involves five primary strategies: enhancing data granularity, optimizing index structures, adding metadata, alignment optimization, and mixed retrieval.

Enhancing data granularity aims to elevate text standardization, consistency, factual accuracy, and rich context to improve the RAG system's performance. This includes removing irrelevant information, dispelling ambiguity in entities and terms, confirming factual accuracy, maintaining context, and updating outdated documents.

Optimizing index structures involves adjusting the size of chunks to capture relevant context, querying across multiple index paths, and incorporating information from the graph structure to capture relevant context by leveraging relationships between nodes in a graph data index.

Adding metadata information involves integrating referenced metadata, such as dates and purposes, into chunks for filtering purposes, and incorporating metadata like chapters and subsections of references to improve retrieval efficiency.

Alignment optimization addresses alignment issues and disparities between documents by introducing "hypothetical questions" [Li *et al.*, 2023d] into documents to rectify alignment issues and differences.

Retrieval

During the retrieval stage, the primary focus is on identifying the appropriate context by calculating the similarity between the query and chunks. The embedding model is central to this process. In the advanced RAG, there is potential for optimization of the embedding models.

Fine-tuning Embedding. Fine-tuning embedding models significantly impact the relevance of retrieved content in RAG systems. This process involves customizing embedding models to enhance retrieval relevance in domain-specific contexts, especially for professional domains dealing with evolving or rare terms. The BGE embedding model [BAAI, 2023], such as BGE-large-EN developed by BAAI[2], is an example of a high-performance embedding model that can be fine-tuned to optimize retrieval relevance. Training data for fine-tuning can be generated using language models like GPT-3.5-turbo to formulate questions grounded on document chunks, which are then used as fine-tuning pairs.

Dynamic Embedding adapts to the context in which words are used, unlike static embedding, which uses a single vector for each word [Karpukhin *et al.*, 2020]. For example, in transformer models like BERT, the same word can have varied embeddings depending on surrounding words. OpenAI's embeddings-ada-02 model[3], built upon the principles of LLMs like GPT, is a sophisticated dynamic embedding model that captures contextual understanding. However, it may not exhibit the same sensitivity to context as the latest full-size language models like GPT-4.

[2]https://huggingface.co/BAAI/bge-large-en

[3]https://platform.openai.com/docs/guides/embeddings

Post-Retrieval Process

After retrieving valuable context from the database, it is essential to merge it with the query as an input into LLMs while addressing challenges posed by context window limits. Simply presenting all relevant documents to the LLM at once may exceed the context window limit, introduce noise, and hinder the focus on crucial information. Additional processing of the retrieved content is necessary to address these issues.

Re-Ranking. 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[4], LangChain[5], and HayStack [Blagojevi, 2023]. For example, Diversity Ranker[6] prioritizes reordering based on document diversity, while LostInTheMiddleRanker alternates placing the best document at the beginning and end of the context window. Additionally, approaches like cohereAI rerank [Cohere, 2023], bge-rerank[7], and LongLLMLingua [Jiang *et al.*, 2023a] recalculate the semantic similarity between relevant text and the query, addressing the challenge of interpreting vector-based simulated searches for semantic similarity.

Prompt Compression. Research indicates that noise in retrieved documents adversely affects RAG performance. In post-processing, the emphasis lies in compressing irrelevant context, highlighting pivotal paragraphs, and reducing the overall context length. Approaches such as Selective Context and LLMLingua [Litman *et al.*, 2020, Anderson *et al.*, 2022] utilize small language models to calculate prompt mutual information or perplexity, estimating element importance. Recomp [Xu *et al.*, 2023a] addresses this by training compressors at different granularities, while Long Context [Xu *et al.*, 2023b] and "Walking in the Memory Maze" [Chen *et al.*, 2023a] design summarization techniques to enhance LLM's key information perception, particularly in dealing with extensive contexts.

3.3 Modular RAG

The modular RAG structure diverges from the traditional Naive RAG framework, providing greater versatility and flexibility. It integrates various methods to enhance functional modules, such as incorporating a search module for similarity retrieval and applying a fine-tuning approach in the retriever [Lin *et al.*, 2023]. Restructured RAG modules [Yu *et al.*, 2022] and iterative methodologies like [Shao *et al.*, 2023] have been developed to address specific issues. The modular RAG paradigm is increasingly becoming the norm in the RAG domain, allowing for either a serialized pipeline or an end-to-end training approach across multiple modules. The comparison of three RAG paradigms

[4]https://www.llamaindex.ai

[5]https://www.langchain.com/

[6]https://haystack.deepset.ai/blog/enhancing-rag-pipelines-in-haystack

[7]https://huggingface.co/BAAI/bge-reranker-large

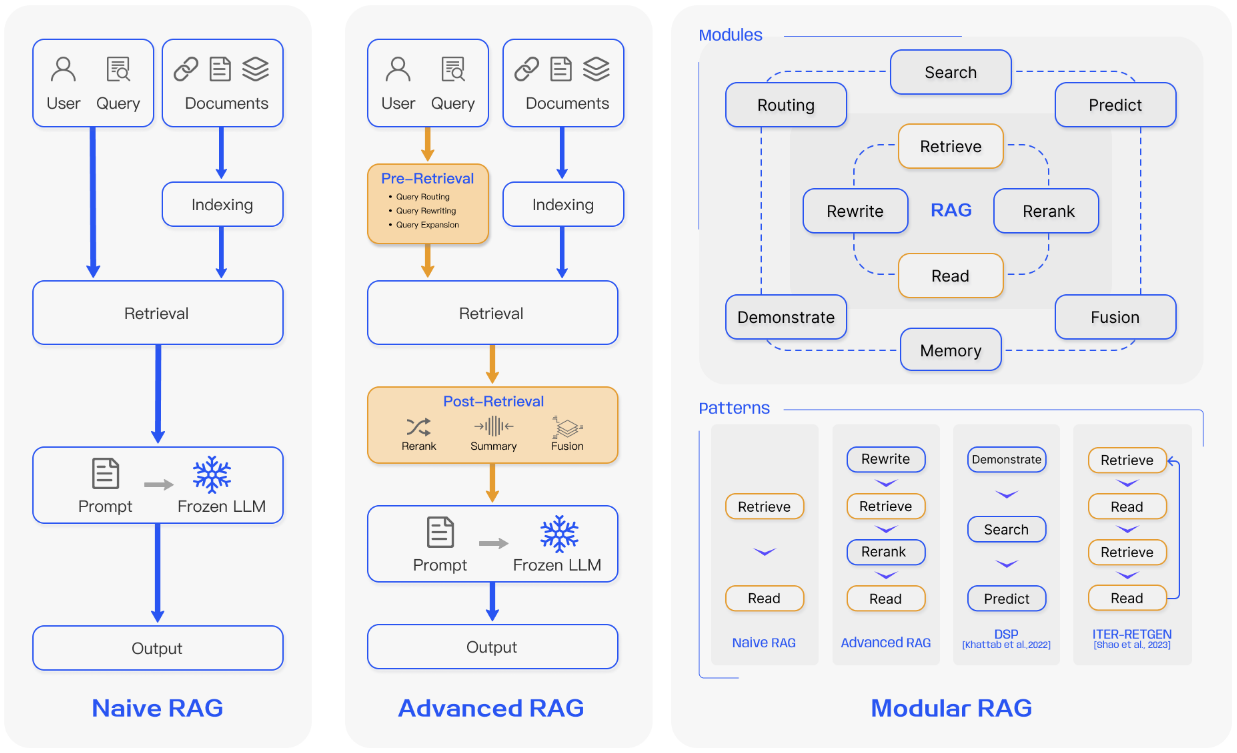

Figure 3: Comparison between the three paradigms of RAG

is depicted in Figure 3. However, Modular RAG is not standalone. Advanced RAG is a specialized form of modular RAG, and further, Naive RAG itself is a special case of Advanced RAG. The relationship among the three paradigms is one of inheritance and development.

New Modules

Search Module. In contrast to the similarity retrieval in Naive/Advanced RAG, the Search Module is tailored to specific scenarios and incorporates direct searches on additional corpora. This integration is achieved using code generated by the LLM, query languages such as SQL or Cypher, and other custom tools. The data sources for these searches can include search engines, text data, tabular data, and knowledge graphs [Wang *et al.*, 2023d].

Memory Module. This module harnesses the memory capabilities of the LLM to guide retrieval. The approach involves identifying memories most similar to the current input. Selfmem [Cheng *et al.*, 2023b] utilizes a retrieval-enhanced generator to create an unbounded memory pool iteratively, combining the "original question" and "dual question". By employing a retrieval-enhanced generative model that uses its own outputs to improve itself, the text becomes more aligned with the data distribution during the reasoning process. Consequently, the model's own outputs are utilized instead of the training data [Wang *et al.*, 2022a].

Fusion. RAG-Fusion [Raudaschl, 2023]enhances traditional search systems by addressing their limitations through a multi-query approach that expands user queries into multiple, diverse perspectives using an LLM. This approach not only captures the explicit information users seek but also uncovers deeper, transformative knowledge. The fusion process involves parallel vector searches of both original and expanded queries, intelligent re-ranking to optimize results, and pairing the best outcomes with new queries. This sophisticated method ensures search results that align closely with both the explicit and implicit intentions of the user, leading to more insightful and relevant information discovery.

Routing. The RAG system's retrieval process utilizes diverse sources, differing in domain, language, and format, which can be either alternated or merged based on the situation [Li *et al.*, 2023b]. Query routing decides the subsequent action to a user's query, with options ranging from summarization, searching specific databases, or merging different pathways into a single response. The query router also chooses the appropriate data store for the query, which may include various sources like vector stores, graph databases, or relational databases, or a hierarchy of indices—for instance, a summary index and a document block vector index for multi-document storage. The query router's decision-making is predefined and executed via LLMs calls, which direct the query to the chosen index.

Predict . It addresses the common issues of redundancy and noise in retrieved content. Instead of directly retrieving from a data source, this module utilizes the LLM to generate the necessary context [Yu *et al.*, 2022]. The content produced by the LLM is more likely to contain pertinent information compared to that obtained through direct retrieval.

Task Adapter. This module focuses on adapting RAG to a variety of downstream tasks. UPRISE automates the retrieval of prompts for zero-shot task inputs from a pre-constructed data pool, thereby enhancing universality across tasks and models [Cheng *et al.*, 2023a]. Meanwhile, PROMPTAGATOR [Dai *et al.*, 2022] utilizes LLM as a few-shot query generator and, based on the generated data, creates task-specific retrievers. By leveraging the generalization capability of LLMs, it enables the development of task-specific end-to-end retrievers with minimal examples.

New Patterns

The organizational structure of Modular RAG is highly adaptable, allowing for the substitution or rearrangement of modules within the RAG process to suit specific problem contexts.

Naive RAG and Advanced RAG can both be considered as being composed of some fixed modules. As illustrated in the figure 3, Naive RAG primarily consists of the "Retrieve" and "Read" modules. A typical pattern of Advanced RAG builds upon the foundation of Naive RAG by adding "Rewrite" and "Rerank" modules. However, on the whole, modular RAG enjoys greater diversity and flexibility.

Current research primarily explores two organizational paradigms. The first involves adding or replacing modules, while the second focuses on adjusting the organizational flow between modules. This flexibility enables tailoring the RAG process to effectively address a wide array of tasks.

Adding or Replacing Modules.The strategy of introducing or substituting modules involves maintaining the core structure of the Retrieval-Read process while integrating additional modules to enhance specific functionalities. The RRR model [Ma *et al.*, 2023a] introduces the Rewrite-Retrieve-Read process, utilizing the LLM performance as a reinforcement learning incentive for a rewriting module. This enables the rewriter to fine-tune retrieval queries, thereby improving the downstream task performance of the reader.

Similarly, modules can be selectively swapped in methodologies like Generate-Read [Yu *et al.*, 2022], where the LLM's generation module takes the place of the retrieval module. The Recite-Read approach [Sun *et al.*, 2022] transforms external retrieval into retrieval from model weights, requiring the LLM to initially memorize task-specific information and subsequently produce output capable of handling knowledge-intensive natural language processing tasks.

Adjusting the Flow between Modules. zheIn the realm of module flow adjustment, there is a focus on enhancing the interaction between language models and retrieval models. DSP [Khattab *et al.*, 2022] introduces the Demonstrate-Search-Predict framework, treating the context learning system as an explicit program rather than a final task prompt, leading to more effective handling of knowledge-intensive tasks. The ITER-RETGEN [Shao *et al.*, 2023] approach utilizes generated content to guide retrieval, iteratively implementing "retrieval-enhanced generation" and "generation-enhanced retrieval" within a Retrieve-Read-Retrieve-Read flow. This method demonstrates an innovative way of using one module's output to improve the functionality of another.

Optimizing the RAG Pipeline

The optimization of the retrieval process aims to enhance the efficiency and quality of information in RAG systems. Current research focuses on integrating diverse search technologies, refining retrieval steps, incorporating cognitive backtracking, implementing versatile query strategies, and leveraging embedding similarity. These efforts collectively strive to achieve a balance between retrieval efficiency and the depth of contextual information in RAG systems.

Hybrid Search Exploration. The RAG system optimizes its performance by intelligently integrating various techniques, including keyword-based search, semantic search, and vector search. This approach leverages the unique strengths of each method to accommodate diverse query types and information needs, ensuring consistent retrieval of highly relevant and context-rich information. The use of hybrid search serves as a robust supplement to retrieval strategies, thereby enhancing the overall efficacy of the RAG pipeline.

Recursive Retrieval and Query Engine. Recursive retrieval involves acquiring smaller chunks during the initial retrieval phase to capture key semantic meanings. Subsequently, larger chunks containing more contextual information are provided to the LLM in later stages of the process. This two-step retrieval method helps to strike a balance between efficiency and the delivery of contextually rich responses.

StepBack-prompt approach encourages the LLM to move away from specific instances and engage in reasoning around broader concepts and principles [Zheng *et al.*, 2023]. Experimental results demonstrate a significant performance increase in various challenging, inference-based tasks when backward prompts are used, highlighting their natural adaptability to the RAG process. These retrieval-enhancing steps can be applied both in generating responses to backward prompts and in the final question-answering process.

Sub-Queries. Depending on the scenario, various query strategies can be employed, such as using query engines provided by frameworks like LlamaIndex, leveraging tree queries, utilizing vector queries, or executing simple sequential querying of chunks.

Hypothetical Document Embeddings. HyDE operates on the belief that the answers generated might be closer in the embedding space than a direct query. Using the LLM, HyDE creates a hypothetical document (answer) in response to a query, embeds this document, and uses the resulting embedding to retrieve real documents similar to the hypothetical one. Instead of seeking embedding similarity based on the query, this approach focuses on the embedding similarity from one answer to another [Gao *et al.*, 2022]. However, it might not consistently produce desirable outcomes, especially when the language model is unfamiliar with the subject matter, potentially leading to more instances with errors.

4 Retrieval

In the context of RAG, it is crucial to efficiently retrieve relevant documents from the data source. However, creating a proficient retriever presents significant challenges. This sectionelves into three fundamental questions: 1) How can we achieve accurate semantic representations? 2) What methods can align the semantic spaces of queries and documents? 3) How can the retriever's output be aligned with the preferences of the Large Language Model?

4.1 Enhancing Semantic Representations

In RAG, the semantic space is essential as it involves the multidimensional mapping of queries and documents. Retrieval accuracy in this semantic space significantly impacts RAG outcomes. This section will present two methods for building accurate semantic spaces.

Chunk optimization

When managing external documents, the initial step involves breaking them down into smaller chunks to extract fine-grained features, which are then embedded to represent their semantics. However, embedding overly large or excessively small text chunks may lead to sub-optimal outcomes. Therefore, identifying the optimal chunk size for documents within the corpus is crucial to ensuring the accuracy and relevance of the retrieved results.

Choosing an appropriate chunking strategy requires careful consideration of several vital factors, such as the nature of the indexed content, the embedding model and its optimal block size, the expected length and complexity of user queries, and the specific application's utilization of the retrieved results. For instance, the selection of a chunking model should be based on the content's length—whether it is longer or shorter. Additionally, different embedding models demonstrate distinct performance characteristics at varying block sizes. For example, sentence-transformer performs better with single sentences, while text-embedding-ada-002 excels with blocks containing 256 or 512 tokens.

Additionally, factors like the length and complexity of user input questions, and the specific needs of the application (e.g., semantic search or question answering), have effect on the choice of a chunking strategy. This choice can be directly influenced by the token limits of the selected LLMs, requiring adjustments to the block size. In reality, getting precise query results involves flexibly applying different chunking strategies. There is no one-size-fits-all "best" strategy, only the most appropriate one for a particular context.

Current research in RAG explores various block optimization techniques aimed at improving both retrieval efficiency and accuracy. One such approach involves the use of sliding window technology, enabling layered retrieval by merging globally related information across multiple retrieval processes. Another strategy, known as the "small2big" method, utilizes small text blocks during the initial search phase and subsequently provides larger related text blocks to the language model for processing.

The abstract embedding technique prioritizes top K retrieval based on document abstracts (or summaries), offering a comprehensive understanding of the entire document context. Additionally, the metadata filtering technique leverages document metadata to enhance the filtering process. An innovative approach, the graph indexing technique, transforms entities and relationships into nodes and connections, significantly improving relevance, particularly in the context of multi-hop problems.

The combination of these diverse methods has led to notable advancements, resulting in enhanced retrieval outcomes and improved performance for RAG.

Fine-tuning Embedding Models

Once the appropriate size of chunks is determined, the next crucial step involves embedding these chunks and the query into the semantic space using an embedding model. The effectiveness of the embedding is critical as it impacts the model's ability to represent the corpus. Recent research has introduced prominent embedding models such as AngIE, Voyage, BGE,etc [Li and Li, 2023, VoyageAI, 2023, BAAI, 2023]. These models have undergone pre-training on extensive corpora. However, their capability to accurately capture domain-specific information may be limited when applied to specialized domains.

Moreover, task-specific fine-tuning of embedding models is essential to ensure that the model comprehends the user query in terms of content relevance. A model without fine-tuning may not adequately address the requirements of a specific task. Consequently, fine-tuning an embedding model becomes crucial for downstream applications. There are two primary paradigms in embedding fine-tuning methods.

Domain Knowledge Fine-tuning. To ensure that an embedding model accurately captures domain-specific information, it is imperative to utilize domain-specific datasets for fine-tuning. This process diverges from standard language model fine-tuning, chiefly in the nature of the datasets involved. Typically, the dataset for embedding model fine-tuning encompasses three principal elements: queries, a corpus, and relevant documents. The model employs these queries to identify pertinent documents within the corpus. The efficacy of the model is then gauged based on its ability to retrieve these relevant documents in response to the queries. The dataset construction, model fine-tuning, and evaluation phases each present distinct challenges. The LlamaIndex [Liu, 2023] introduces a suite of pivotal classes and functions designed to enhance the embedding model fine-tuning workflow, thereby simplifying these intricate processes. By curating a corpus infused with domain knowledge and leveraging the methodologies offered, one can adeptly fine-tune an embedding model to align closely with the specific requirements of the target domain.

Fine-tuning for Downstream Tasks. Fine-tuning embedding models for downstream tasks is a critical step in enhancing model performance. In the realm of utilizing RAG for these tasks, innovative methods have emerged to fine-tune embedding models by harnessing the capabilities of LLMs. For example, PROMPTAGATOR [Dai *et al.*, 2022] utilizes the LLM as a few-shot query generator to create task-specific retrievers, addressing challenges in supervised fine-tuning, particularly in data-scarce domains. Another approach, LLM-Embedder [Zhang *et al.*, 2023a], exploits LLMs to generate reward signals for data across multiple downstream tasks. The retriever is fine-tuned with two types of supervised signals: hard labels for the dataset and soft rewards from the LLMs. This dual-signal approach fosters a more effective fine-tuning process, tailoring the embedding model to diverse downstream applications.

While these methods improve semantic representation by incorporating domain knowledge and task-specific fine-tuning, retrievers may not always exhibit optimal compatibility with certain LLMs. To address this, some researchers have explored direct supervision of the fine-tuning process using feedback from LLMs. This direct supervision seeks to align the retriever more closely with the LLM, thereby improving performance on downstream tasks. A more comprehensive discussion on this topic is presented in Section 4.3.

4.2 Aligning Queries and Documents

In the context of RAG applications, retrievers may utilize a single embedding model for encoding both the query and the documents, or employ separate models for each. Additionally, the user's original query may suffer from imprecise phrasing and lack of semantic information. Therefore, it is crucial to align the semantic space of the user's query with those of the documents. This section introduces two fundamental techniques aimed at achieving this alignment.

Query Rewriting

Query rewriting is a fundamental approach for aligning the semantics of a query and a document. Methods such as Query2Doc and ITER-RETGEN leverage LLMs to create a pseudo-document by combining the original query with additional guidance [Wang *et al.*, 2023c, Shao *et al.*, 2023]. HyDE constructs query vectors using textual cues to generate a "hypothetical" document capturing essential patterns [Gao *et al.*, 2022]. RRR introduces a framework that reverses the traditional retrieval and reading order, focusing on query rewriting [Ma *et al.*, 2023a]. STEP-BACKPROMPTING enables LLMs to perform abstract reasoning and retrieval based on high-level concepts [Zheng *et al.*, 2023]. Additionally, the multi-query retrieval method utilizes LLMs to generate and execute multiple search queries simultaneously, advantageous for addressing complex problems with multiple sub-problems.

Embedding Transformation

Beyond broad strategies such as query rewriting, there exist more granular techniques specifically designed for embedding transformations. LlamaIndex [Liu, 2023] exemplifies this by introducing an adapter module that can be integrated following the query encoder. This adapter facilitates fine-tuning, thereby optimizing the representation of query embeddings to map them into a latent space that is more closely aligned with the intended tasks.

The challenge of aligning queries with structured external documents, particularly when addressing the incongruity between structured and unstructured data, is addressed by SANTA [Li *et al.*, 2023d]. It enhances the retriever's sensitivity to structured information through two pre-training strategies: first, by leveraging the intrinsic alignment between structured and unstructured data to inform contrastive learning in a structured-aware pre-training scheme; and second, by implementing Masked Entity Prediction. The latter utilizes an entity-centric masking strategy that encourages language models to predict and fill in the masked entities, thereby fostering a deeper understanding of structured data.

The issue of aligning queries with structured external documents, especially when dealing with the disparity between structured and unstructured data, is tackled by SANTA [Li *et al.*, 2023d]. This approach improves the retriever's ability to recognize structured information through two pre-training strategies: firstly, by utilizing the inherent alignment between structured and unstructured data to guide contrastive learning in a structured-aware pre-training scheme; and secondly, by employing Masked Entity Prediction. The latter uses an entity-centric masking strategy to prompt language models to predict and complete the masked entities, thus promoting a more profound comprehension of structured data.

4.3 Aligning Retriever and LLM

In the RAG pipeline, enhancing retrieval hit rate through various techniques may not necessarily improve the final outcome, as the retrieved documents may not align with the specific requirements of the LLMs. Therefore, this section introduces two methods aimed at aligning the retriever outputs with the preferences of the LLMs.

Fine-tuning Retrievers

Several studies utilize feedback signals from LLMs to refine retrieval models. For instance, AAR [Yu *et al.*, 2023b] introduces supervisory signals for a pre-trained retriever using an encoder-decoder architecture. This is achieved by identifying the LM's preferred documents through FiD cross-attention scores. Subsequently, the retriever undergoes fine-tuning with hard negative sampling and standard cross-entropy loss. Ultimately, the refined retriever can be directly applied to enhance unseen target LMs, resulting in improved performance in the target task. Additionally, it is suggested that LLMs may have a preference for focusing on readable rather than information-rich documents.

REPLUG [Shi *et al.*, 2023] utilizes a retriever and an LLM to calculate the probability distributions of the retrieved documents and then performs supervised training by computing the KL divergence. This straightforward and effective training method enhances the performance of the retrieval model by using an LM as the supervisory signal, eliminating the need for specific cross-attention mechanisms.

UPRISE [Cheng *et al.*, 2023a] also employs frozen LLMs to fine-tune the prompt retriever. Both the LLM and the retriever take prompt-input pairs as inputs and utilize the scores provided by the LLM to supervise the retriever's training, effectively treating the LLM as a dataset labeler. In addition, Atlas [Izacard *et al.*, 2022] proposes four methods of supervised fine-tuning embedding models:

- *Attention Distillation*. This approach employs cross-attention scores generated by the LLM during output to distill the model's knowledge.
- *EMDR2*. By using the Expectation-Maximization algorithm, this method trains the model with retrieved documents as latent variables.
- *Perplexity Distillation* directly trains the model using the perplexity of generated tokens as an indicator.

- *LOOP*. This method presents a novel loss function based on the impact of document deletion on LLM prediction, offering an efficient training strategy to better adapt the model to specific tasks.

These approaches aim to improve the synergy between the retriever and the LLM, leading to enhanced retrieval performance and more accurate responses to user inquiries.

Adapters

Fine-tuning models may present challenges, such as integrating functionality through an API or addressing constraints arising from limited local computational resources. Consequently, some approaches opt to incorporate an external adapter to aid in alignment.

PRCA trains the adapter through a context extraction phase and a reward-driven phase. The retriever's output is then optimized using a token-based autoregressive strategy [Yang *et al.*, 2023b]. The token filtering approach employs cross-attention scores to efficiently filter tokens, selecting only the highest-scoring input tokens [Berchansky *et al.*, 2023].RECOMP introduces both extractive and generative compressors for summary generation. These compressors either select relevant sentences or synthesize document information, creating summaries tailored to multi-document queries [Xu *et al.*, 2023a].

Furthermore, PKG introduces an innovative method for integrating knowledge into white-box models via directive fine-tuning [Luo *et al.*, 2023]. In this approach, the retriever module is directly substituted to generate relevant documents according to a query. This method assists in addressing the difficulties encountered during the fine-tuning process and enhances model performance.

5 Generation

A crucial component of RAG is its generator, which is responsible for converting retrieved information into coherent and fluent text. Unlike traditional language models, RAG's generator sets itself apart by improving accuracy and relevance via the incorporation of retrieved data. In RAG, the generator's input encompasses not only typical contextual information but also relevant text segments obtained through the retriever. This comprehensive input enables the generator to gain a deep understanding of the question's context, resulting in more informative and contextually relevant responses.

Furthermore, the generator is guided by the retrieved text to ensure coherence between the generated content and the obtained information. The diverse input data has led to targeted efforts during the generation phase, all aimed at refining the adaptation of the large model to the input data derived from queries and documents. In the following subsections, we will explore the introduction of the generator by delving into aspects of post-retrieval processing and fine-tuning.

5.1 Post-retrieval with Frozen LLM

In the realm of untunable LLMs , many studies rely on well-established models like GPT-4 [OpenAI, 2023] to harness their comprehensive internal knowledge for systematically synthesizing retrieved information from various documents. However, challenges persist with these large models, including limitations on context length and susceptibility to redundant information. To tackle these issues, certain research endeavors have turned their focus to post-retrieval processing.

Post-retrieval processing involves treating, filtering, or optimizing the relevant information retrieved by the retriever from a large document database. Its main goal is to enhance the quality of retrieval results, aligning them more closely with user needs or subsequent tasks. It can be viewed as a reprocessing of the documents obtained during the retrieval phase. Common operations in post-retrieval processing typically include information compression and result reranking.

Information Compression

The retriever excels at retrieving relevant information from a vast knowledge base, but managing the substantial amount of information within retrieval documents is a challenge. Ongoing research aims to extend the context length of large language models to tackle this issue. However, current large models still struggle with context limitations. Therefore, there are scenarios where condensing information becomes necessary. Information condensation is significant for reducing noise, addressing context length restrictions, and enhancing generation effects.

PRCA tackled this issue by training an information extractor [Yang *et al.*, 2023b]. In the context extraction phase, when provided with an input text S_{input}, it is capable of producing an output sequence $C_{extracted}$ that represents the condensed context from the input document. The training process is designed to minimize the difference between $C_{extracted}$ and the actual context C_{truth}.

Similarly, RECOMP adopts a comparable approach by training an information condenser using contrastive learning [Xu *et al.*, 2023a]. Each training data point consists of one positive sample and five negative samples, and the encoder undergoes training using contrastive loss throughout this process [Karpukhin *et al.*, 2020] .

Another study has taken a different approach by aiming to reduce the number of documents in order to improve the accuracy of the model's answers. In the study by [Ma *et al.*, 2023b], they propose the “Filter-Reranker” paradigm, which combines the strengths of LLMs and Small Language Models (SLMs). In this paradigm, SLMs serve as filters, while LLMs function as reordering agents. The research shows that instructing LLMs to rearrange challenging samples identified by SLMs leads to significant improvements in various Information Extraction (IE) tasks.

Reranking

The re-ranking model is pivotal in optimizing the document set retrieved from the retriever. Language models often face performance declines when additional context is introduced, and re-ranking effectively addresses this issue. The core concept involves rearranging document records to prioritize the most relevant items at the top, thereby limiting the total number of documents. This not only resolves the challenge of context window expansion during retrieval but also enhances retrieval efficiency and responsiveness.

The re-ranking model assumes a dual role throughout the information retrieval process, functioning as both an optimizer and a refiner. It provides more effective and accurate input for subsequent language model processing [Zhuang *et al.*, 2023].

Contextual compression is incorporated into the reordering process to offer more precise retrieval information. This method entails reducing the content of individual documents and filtering the entire document, with the ultimate goal of presenting the most relevant information in the search results for a more focused and accurate display of pertinent content.

5.2 Fine-tuning LLM for RAG

Optimizing the generator within the RAG model is a critical aspect of its architecture. The generator's role is to take the retrieved information and produce relevant text, forming the final output of the model. The optimization of the generator aims to ensure that the generated text is both natural and effectively leverages the retrieved documents to better meet the user's query needs.

In standard LLMs generation tasks, the input typically consists of a query. RAG stands out by incorporating not only a query but also various retrieved documents (structured/unstructured) by the retriever into the input. This additional information can significantly influence the model's understanding, particularly for smaller models. In such cases, fine-tuning the model to adapt to the input of both query and retrieved documents becomes crucial. Before presenting the input to the fine-tuned model, post-retrieval processing usually occurs for the documents retrieved by the retriever. It is essential to note that the fine-tuning method for the generator in RAG aligns with the general fine-tuning approach for LLMs. In the following, we will briefly describe some representative works involving data (formatted/unformatted) and optimization functions.

General Optimization Process

As part of the general optimization process, the training data typically consists of input-output pairs, aiming to train the model to produce the output y given the input x. In the work of Self-Mem [Cheng *et al.*, 2023b], a traditional training process is employed, where given the input x, relevant documents z are retrieved (selecting Top-1 in the paper), and after integrating (x, z), the model generates the output y. The paper utilizes two common paradigms for fine-tuning, namely Joint-Encoder and Dual-Encoder [Arora *et al.*, 2023, Wang *et al.*, 2022b, Lewis *et al.*, 2020, Xia *et al.*, 2019, Cai *et al.*, 2021, Cheng *et al.*, 2022].

In the Joint-Encoder paradigm, a standard model based on an encoder-decoder is used. Here, the encoder initially encodes the input, and the decoder, through attention mechanisms, combines the encoded results to generate tokens in an autoregressive manner. On the other hand, in the Dual-Encoder paradigm, the system sets up two independent encoders, with each encoder encoding the input (query, context) and the document, respectively. The resulting outputs undergo bidirectional cross-attention processing by the decoder in sequence. Both architectures utilize the Transformer [Vaswani *et al.*, 2017] as the foundational block and optimize with Negative Log-Likelihood loss.

Utilizing Contrastive Learning

In the phase of preparing training data for language models, interaction pairs of input and output are usually created. This traditional method can lead to "exposure bias," where the model is only trained on individual, correct output examples, thus restricting its exposure to a range of possible outputs citesequence. This limitation can hinder the model's real-world performance by causing it to overfit to the particular examples in the training set, thereby reducing its ability to generalize across various contexts.

To mitigate exposure bias, SURGE [Kang *et al.*, 2023] proposes the use of graph-text contrastive learning. This method includes a contrastive learning objective that prompts the model to produce a range of plausible and coherent responses, expanding beyond the instances encountered in the training data. This approach is crucial in reducing overfitting and strengthening the model's ability to generalize.

For retrieval tasks that engage with structured data, the SANTA framework [Li *et al.*, 2023d] implements a tripartite training regimen to effectively encapsulate both structural and semantic nuances. The initial phase focuses on the retriever, where contrastive learning is harnessed to refine the query and document embeddings.

Subsequently, the generator's preliminary training stage employs contrastive learning to align the structured data with its unstructured document descriptions. In a further stage of generator training, the model acknowledges the critical role of entity semantics in the representation learning of textual data for retrieval, as highlighted by [Sciavolino *et al.*, 2021, Zhang *et al.*, 2019]. This process commences with the identification of entities within the structured data, followed by the application of masks over these entities within the generator's input data, thus setting the stage for the model to anticipate and predict these masked elements.

The training regimen progresses with the model learning to reconstruct the masked entities by leveraging contextual information. This exercise cultivates the model's comprehension of the textual data's structural semantics and facilitates the alignment of pertinent entities within the structured data. The overarching optimization goal is to train the language model to accurately restore the obscured spans, thereby enriching its understanding of entity semantics [Ye *et al.*, 2020].

6 Augmentation in RAG

This section is structured around three key aspects: the augmentation stage, sources of augmentation data, and the augmentation process. These facets elucidate the critical technologies pivotal to RAG's development. A taxonomy of RAG's core components is presented in Figure 4.

6.1 RAG in Augmentation Stages

RAG, a knowledge-intensive endeavor, incorporates a variety of technical methodologies across the pre-training, fine-tuning, and inference stages of language model training.

Pre-training Stage

During the pre-training stage, researchers have investigated methods to bolster PTMs for open-domain QA through

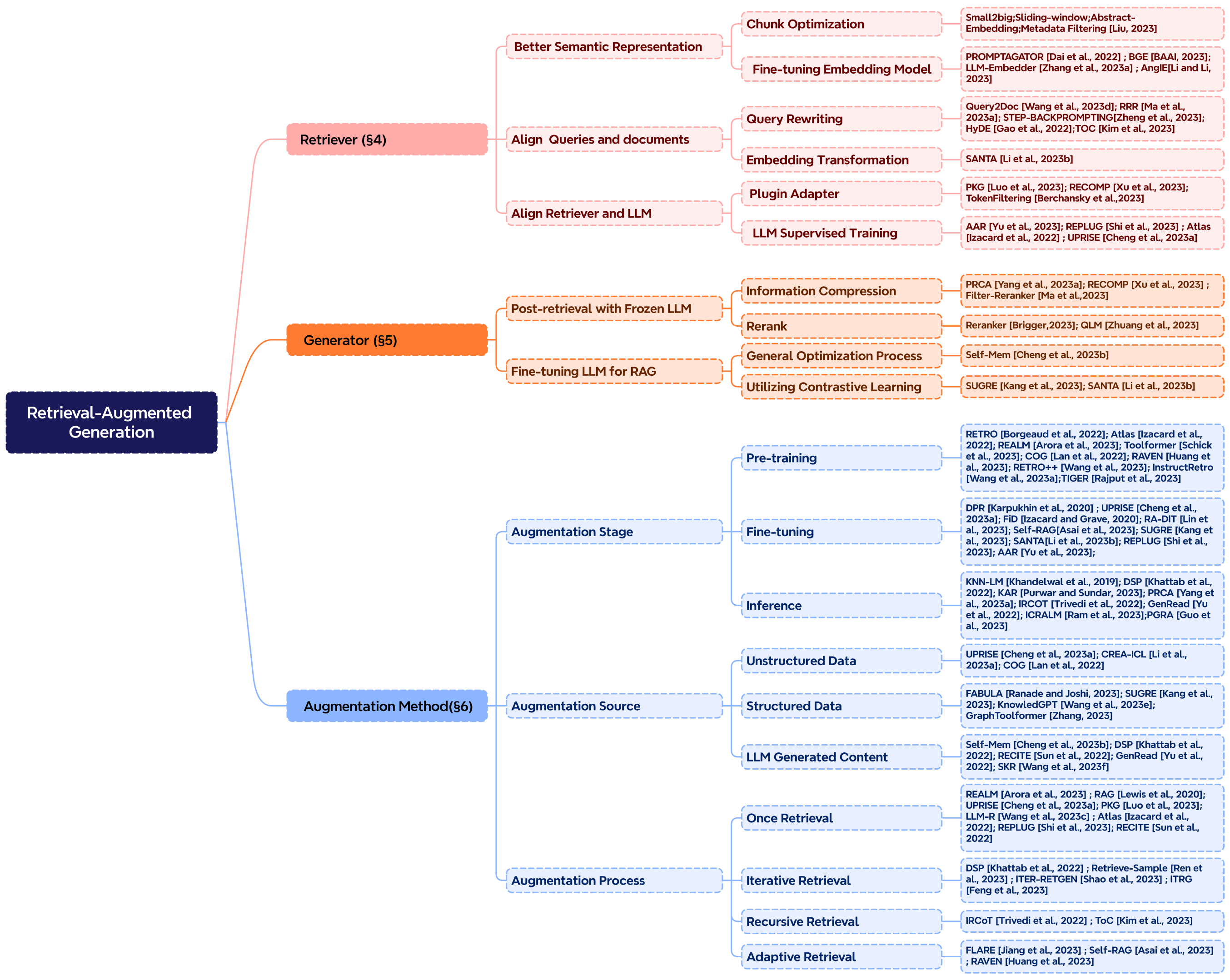
Retrieval-Augmented Generation
Retriever (§4)
Better Semantic Representation
Chunk Optimization
Small2big;Sliding-window;Abstract-Embedding;Metadata Filtering [Liu, 2023]
Fine-tuning Embedding Model
PROMPTAGATOR [Dai et al., 2022] ; BGE [BAAI, 2023]; LLM-Embedder [Zhang et al., 2023a] ; AnglE[Li and Li, 2023]
Align Queries and documents
Query Rewriting
Query2Doc [Wang et al., 2023d]; RRR [Ma et al., 2023a]; STEP-BACKPROMPTING[Zheng et al., 2023]; HyDE [Gao et al., 2022];TOC [Kim et al., 2023]
Embedding Transformation
SANTA [Li et al., 2023b]
Align Retriever and LLM
Plugin Adapter
PKG [Luo et al., 2023]; RECOMP [Xu et al., 2023]; TokenFiltering [Berchansky et al.,2023]
LLM Supervised Training
AAR [Yu et al., 2023]; REPLUG [Shi et al., 2023] ; Atlas [Izacard et al., 2022] ; UPRISE [Cheng et al., 2023a]
Generator (§5)
Post-retrieval with Frozen LLM
Information Compression
PRCA [Yang et al., 2023a]; RECOMP [Xu et al., 2023] ; Filter-Reranker [Ma et al.,2023]
Rerank
Reranker [Brigger,2023]; QLM [Zhuang et al., 2023]
Fine-tuning LLM for RAG
General Optimization Process
Self-Mem [Cheng et al., 2023b]
Utilizing Contrastive Learning
SUGRE [Kang et al., 2023]; SANTA [Li et al., 2023b]
Augmentation Method(§6)
Augmentation Stage
Pre-training
RETRO [Borgeaud et al., 2022]; Atlas [Izacard et al., 2022]; REALM [Arora et al., 2023]; Toolformer [Schick et al., 2023]; COG [Lan et al., 2022]; RAVEN [Huang et al., 2023]; RETRO++ [Wang et al., 2023]; InstructRetro [Wang et al., 2023a];TIGER [Rajput et al., 2023]
Fine-tuning
DPR [Karpukhin et al., 2020] ; UPRISE [Cheng et al., 2023a]; FiD [Izacard and Grave, 2020]; RA-DIT [Lin et al., 2023]; Self-RAG[Asai et al., 2023]; SUGRE [Kang et al., 2023]; SANTA[Li et al., 2023b]; REPLUG [Shi et al., 2023]; AAR [Yu et al., 2023];
Inference
KNN-LM [Khandelwal et al., 2019]; DSP [Khattab et al., 2022]; KAR [Purwar and Sundar, 2023]; PRCA [Yang et al., 2023a]; IRCOT [Trivedi et al., 2022]; GenRead [Yu et al., 2022]; ICRALM [Ram et al., 2023];PGRA [Guo et al., 2023]
Augmentation Source
Unstructured Data
UPRISE [Cheng et al., 2023a]; CREA-ICL [Li et al., 2023a]; COG [Lan et al., 2022]
Structured Data
FABULA [Ranade and Joshi, 2023]; SUGRE [Kang et al., 2023]; KnowledGPT [Wang et al., 2023e]; GraphToolformer [Zhang, 2023]
LLM Generated Content
Self-Mem [Cheng et al., 2023b]; DSP [Khattab et al., 2022]; RECITE [Sun et al., 2022]; GenRead [Yu et al., 2022]; SKR [Wang et al., 2023f]
Augmentation Process
Once Retrieval
REALM [Arora et al., 2023] ; RAG [Lewis et al., 2020]; UPRISE [Cheng et al., 2023a]; PKG [Luo et al., 2023]; LLM-R [Wang et al., 2023c] ; Atlas [Izacard et al., 2022]; REPLUG [Shi et al., 2023]; RECITE [Sun et al., 2022]
Iterative Retrieval
DSP [Khattab et al., 2022] ; Retrieve-Sample [Ren et al., 2023] ; ITER-RETGEN [Shao et al., 2023] ; ITRG [Feng et al., 2023]
Recursive Retrieval
IRCoT [Trivedi et al., 2022] ; ToC [Kim et al., 2023]
Adaptive Retrieval
FLARE [Jiang et al., 2023] ; Self-RAG [Asai et al., 2023] ; RAVEN [Huang et al., 2023]

Figure 4: Taxonomy of RAG's core components

retrieval-based strategies. The REALM model adopts a structured, interpretable method for knowledge embedding, framing pre-training, and fine-tuning as a retrieve-then-predict workflow within the masked language model (MLM) framework [Arora *et al.*, 2023] .

RETRO [Borgeaud *et al.*, 2022] leverages retrieval augmentation for large-scale pre-training from scratch, achieving a reduction in model parameters while surpassing standard GPT models in terms of perplexity. RETRO distinguishes itself with an additional encoder designed to process features of entities retrieved from an external knowledge base, building on the foundational structure of GPT models.

Atlas[Izacard *et al.*, 2022] also incorporates a retrieval mechanism into the T5 architecture [Raffel *et al.*, 2020] in both the pre-training and fine-tuning stages. It uses a pre-trained T5 to initialize the encoder-decoder language model and a pre-trained Contriever for the dense retriever, improving its efficiency for complex language modeling tasks.

Furthermore, COG [Lan *et al.*, 2022] introduces a novel text generation methodology that emulates copying text fragments from pre-existing collections. Utilizing efficient vector search tools, COG computes and indexes contextually meaningful representations of text fragments, demonstrating superior performance in domains such as question-answering and domain adaptation when compared to RETRO.

The advent of scaling laws has catalyzed the growth of model parameters, propelling autoregressive models into the mainstream. Researchers are expanding the RAG approach to pretrained larger models, with RETRO++ exemplifying this trend by scaling up the model parameters while preserving or enhancing performance [Wang *et al.*, 2023b].

Empirical evidence underscores marked improvements in text generation quality, factual accuracy, reduced toxicity, and downstream task proficiency, especially in knowledge-intensive applications like open-domain QA. These results imply that integrating retrieval mechanisms into the pre-training of autoregressive language models constitutes a promising avenue, marrying sophisticated retrieval techniques with expansive language models to yield more precise and efficient language generation.

The benefits of augmented pre-training include a robust foundational model that outperforms standard GPT models in perplexity, text generation quality, and task-specific performance, all while utilizing fewer parameters. This method is particularly adept at handling knowledge-intensive tasks and facilitates the development of domain-specific models through training on specialized corpora.

Nonetheless, this approach faces challenges such as the necessity for extensive pre-training datasets and resources, as well as diminished update frequencies with increasing model sizes. Despite these hurdles, the approach offers significant advantages in model resilience. Once trained, retrieval-enhanced models can operate independently of external libraries, enhancing generation speed and operational efficiency. The potential gains identified render this methodology a compelling subject for ongoing investigation and innovation in artificial intelligence and machine learning.

Fine-tuning Stage

RAG and Fine-tuning are powerful tools for enhancing LLMs, and combining the two can meet the needs of more specific scenarios. On one hand, fine-tuning allows for the retrieval of documents with a unique style, achieving better semantic expression and aligning the differences between queries and documents. This ensures that the output of the retriever is more aptly suited to the scenario at hand. On the other hand, fine-tuning can fulfill the generation needs of making stylized and targeted adjustments. Furthermore, fine-tuning can also be used to align the retriever and generator for improved model synergy.

The main goal of fine-tuning the retriever is to improve the quality of semantic representations, achieved by directly fine-tuning the Embedding model using a corpus [Liu, 2023]. By aligning the retriever's capabilities with the preferences of the LLMs through feedback signals, both can be better coordinated [Yu *et al.*, 2023b, Izacard *et al.*, 2022, Yang *et al.*, 2023b, Shi *et al.*, 2023]. Fine-tuning the retriever for specific downstream tasks can lead to improved adaptability [cite]. The introduction of task-agnostic fine-tuning aims to enhance the retriever's versatility in multi-task scenarios [Cheng *et al.*, 2023a].

Fine-tuning generator can result in outputs that are more stylized and customized. On one hand, it allows for specialized adaptation to different input data formats. For example, fine-tuning LLMs to fit the structure of knowledge graphs [Kang *et al.*, 2023], the structure of text pairs [Kang *et al.*, 2023, Cheng *et al.*, 2023b], and other specific structures [Li *et al.*, 2023d]. On the other hand, by constructing directive datasets, one can demand LLMs to generate specific formats content. For instance, in adaptive or iterative retrieval scenarios, LLMs are fine-tuned to generate content that will help determine the timing for the next step of action [Jiang *et al.*, 2023b, Asai *et al.*, 2023].

By synergistically fine-tuning both the retriever and the generator, we can enhance the model's generalization capabilities and avoid overfitting that may arise from training them separately. However, joint fine-tuning also leads to increased resource consumption. RA-DIT [Lin *et al.*, 2023] presents a lightweight, dual-instruction tuning framework that can effectively add retrieval capabilities to any LLMs. The retrieval-enhanced directive fine-tuning updates the LLM, guiding it to make more efficient use of the information retrieved and to disregard distracting content.

Despite its advantages, fine-tuning has limitations, including the need for specialized datasets for RAG fine-tuning and the requirement for significant computational resources. However, this stage allows for customizing models to specific needs and data formats, potentially reducing resource usage compared to the pre-training phase while still being able to fine-tune the model's output style.

In summary, the fine-tuning stage is essential for the adaptation of RAG models to specific tasks, enabling the refinement of both retrievers and generators. This stage enhances the model's versatility and adaptability to various tasks, despite the challenges presented by resource and dataset requirements. The strategic fine-tuning of RAG models is therefore a critical component in the development of efficient and effective retrieval-augmented systems.

Inference Stage

The inference stage in RAG models is crucial, as it involves extensive integration with LLMs. Traditional RAG approaches, also known as Naive RAG, involve incorporating retrieval content at this stage to guide the generation process.

To overcome the limitations of Naive RAG, advanced techniques introduce more contextually rich information during inference. The DSP framework [Khattab *et al.*, 2022] utilizes a sophisticated exchange of natural language text between fronzen LMs and retrieval models (RMs), enriching the context and thereby improving generation outcomes. The PKG [Luo *et al.*, 2023] method equips LLMs with a knowledge-guided module that allows for the retrieval of pertinent information without modifying the LMs' parameters, enabling more complex task execution. CREA-ICL [Li *et al.*, 2023b] employs a synchronous retrieval of cross-lingual knowledge to enhance context, while RECITE [Sun *et al.*, 2022] generates context by sampling paragraphs directly from LLMs.

Further refinement of the RAG process during inference is seen in approaches that cater to tasks necessitating multi-step reasoning. ITRG [Feng *et al.*, 2023] iteratively retrieves information to identify the correct reasoning paths, thereby improving task adaptability. ITER-RETGEN [Shao *et al.*, 2023] follows an iterative strategy, merging retrieval and generation in a cyclical process that alternates between "retrieval-enhanced generation" and "generation-enhanced retrieval". For non-knowledge-intensive (NKI) tasks, PGRA [Guo *et al.*, 2023] proposes a two-stage framework, starting with a task-agnostic retriever followed by a prompt-guided reranker to select and prioritize evidence. In contrast, IRCOT [Trivedi *et al.*, 2022] combines RAG with Chain of Thought (CoT) methodologies, alternating CoT-guided retrievals with retrieval-informed CoT processes, significantly boosting GPT-3's performance across various question-answering tasks.

In essence, these inference-stage enhancements provide lightweight, cost-effective alternatives that leverage the capabilities of pre-trained models without necessitating further training. The principal advantage is maintaining static LLM parameters while supplying contextually relevant information to meet specific task demands. Nevertheless, this approach is not without limitations, as it requires meticulous data processing and optimization, and is bound by the foundational model's intrinsic capabilities. To address diverse task requirements effectively, this method is often paired with procedural optimization techniques such as step-wise reasoning, iterative retrieval, and adaptive retrieval strategies.

6.2 Augmentation Source

The effectiveness of RAG models is heavily impacted by the selection of data sources for augmentation. Different levels of knowledge and dimensions require distinct processing techniques. They are categorized as unstructured data, structured data, and content generated by LLMs. The technology tree of representative RAG research with different augmentation aspects is depicted in Figure 5. The leaves, colored in three different shades, represent enhancements using various types of data: unstructured data, structured data, and content generated by LLMs. The diagram clearly shows that initially, augmentation was mainly achieved through unstructured data, such as pure text. This approach later expanded to include the use of structured data (e.g. knowledge graph) for further improvement. More recently, there has been a growing trend in research that utilizes content generated by the LLMs themselves for retrieval and augmentation purposes.

Augmented with Unstructured Data

Unstructured text, is gathered from corpora, such as prompt data for fine-tuning large models [Cheng *et al.*, 2023a] and cross-lingual data [Li *et al.*, 2023b]. Retrieval units vary from tokens (e.g., kNN-LM [Khandelwal *et al.*, 2019]) to phrases (e.g., NPM, COG [Lee *et al.*, 2020, Lan *et al.*, 2022]) and document paragraphs, with finer granularities offering precision at the cost of increased retrieval complexity.

FLARE [Jiang *et al.*, 2023b] introduces an active retrieval approach, triggered by the LM's generation of low-probability words. It creates a temporary sentence for document retrieval, then regenerates the sentence with the retrieved context to predict subsequent sentences. RETRO uses the previous chunk to retrieve the nearest neighbor at the chunk level, combined with the previous chunk's context, it guides the generation of the next chunk. To preserve causality, the generation of the next block C_i only utilizes the nearest neighbor of the previous block $N(C_{i-1})$ and not $N(C_i)$.

Augmented with Structured Data

Structured data, such as knowledge graphs (KGs), provide high-quality context and mitigate model hallucinations. RET-LLMs [Modarressi *et al.*, 2023] constructs a knowledge graph memory from past dialogues for future reference. SUGRE [Kang *et al.*, 2023] employs Graph Neural Networks (GNNs) to encode relevant KG subgraphs, ensuring consistency between retrieved facts and generated text through multi-modal contrastive learning. KnowledGPT [Wang *et al.*, 2023d] generates KB search queries and stores knowledge in a personalized base, enhancing the RAG model's knowledge richness and contextuality.

LLMs-Generated Content in RAG

Addressing the limitations of external auxiliary information in RAG, some research has focused on exploiting LLMs' internal knowledge. SKR [Wang *et al.*, 2023e] classifies questions as known or unknown, applying retrieval enhancement selectively. GenRead [Yu *et al.*, 2022] replaces the retriever with an LLM generator, finding that LLM-generated contexts often contain more accurate answers due to better alignment with the pre-training objectives of causal language modeling. Selfmem [Cheng *et al.*, 2023b] iteratively creates an unbounded memory pool with a retrieval-enhanced generator, using a memory selector to choose outputs that serve as dual problems to the original question, thus self-enhancing the generative model.

These methodologies underscore the breadth of innovative data source utilization in RAG, striving to improve model performance and task effectiveness.

6.3 Augmentation Process

In the domain of RAG, the standard practice often involves a singular retrieval step followed by generation, which can lead to inefficiencies. A notable issue, termed the "lost in the middle" phenomenon, arises when a single retrieval yields redundant content that may dilute or contradict essential information, thereby degrading the generation quality [Liu *et al.*, 2023a]. Furthermore, such singular retrieval is typically insufficient for complex problems demanding multi-step reasoning, as it provides a limited scope of information [Yoran *et al.*, 2023].

As illustrated in Figure 5, to circumvent these challenges, contemporary research has proposed methods for refining the retrieval process: iterative retrieval, recursive retrieval and adaptive retrieval. Iterative retrieval allows the model to engage in multiple retrieval cycles, enhancing the depth and relevance of the information obtained. Recursive retrieval process where the results of one retrieval operation are used as the input for the subsequent retrieval. It helps to delve deeper into relevant information, particularly when dealing with complex or multi-step queries. Recursive retrieval is often used in scenarios where a gradual approach is needed to converge on a final answer, such as in academic research, legal case analysis, or certain types of data mining tasks. Adaptive retrieval, on the other hand, offers a dynamic adjustment mechanism, tailoring the retrieval process to the specific demands of varying tasks and contexts.

Iterative Retrieval

Iterative retrieval in RAG models is a process where documents are repeatedly collected based on the initial query and the text generated thus far, providing a more comprehensive knowledge base for LLMs [Borgeaud *et al.*, 2022, Arora *et al.*, 2023]. This approach has been shown to enhance the robustness of subsequent answer generation by offering additional contextual references through multiple retrieval iterations. However, it may suffer from semantic discontinuity and the accumulation of irrelevant information, as

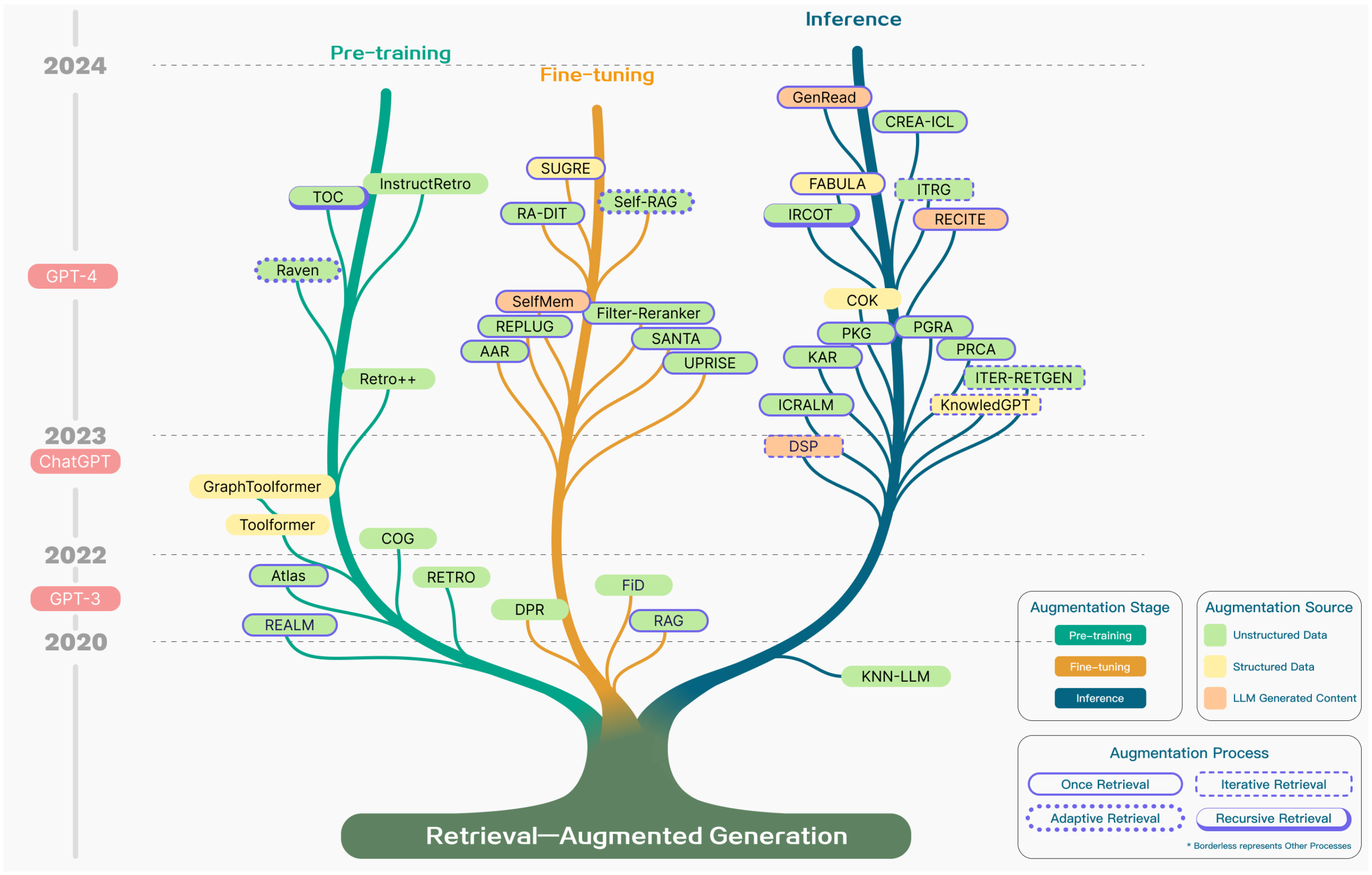

Figure 5: Technology tree of representative RAG research with different augmentation aspects

it typically relies on a sequence of n tokens to demarcate the boundaries between generated text and retrieved documents.

To address specific data scenarios, recursive retrieval and multi-hop retrieval techniques are utilized. Recursive retrieval involves a structured index to process and retrieve data in a hierarchical manner, which may include summarizing sections of a document or lengthy PDF before performing a retrieval based on this summary. Subsequently, a secondary retrieval within the document refines the search, embodying the recursive nature of the process. In contrast, multi-hop retrieval is designed to delve deeper into graph-structured data sources, extracting interconnected information [Li *et al.*, 2023c].

Additionally, some methodologies integrate the steps of retrieval and generation. ITER-RETGEN [Shao *et al.*, 2023] employs a synergistic approach that leverages "retrieval-enhanced generation" alongside "generation-enhanced retrieval" for tasks that necessitate the reproduction of specific information. The model harnesses the content required to address the input task as a contextual basis for retrieving pertinent knowledge, which in turn facilitates the generation of improved responses in subsequent iterations.

Recursive Retrieval

Recursive Retrieval is often used in information retrieval and NLP to improve the depth and relevance of search results. The process involves iteratively refining search queries based on the results obtained from previous searches. Recursive Retrieval aims to enhance the search experience by gradually converging on the most pertinent information through a feedback loop. IRCoT [Trivedi *et al.*, 2022] uses chain-of-thought to guide the retrieval process and refines the CoT with the obtained retrieval results. ToC [Kim *et al.*, 2023] creates a clarification tree that systematically optimizes the ambiguous parts in the Query. It can be particularly useful in complex search scenarios where the user's needs are not entirely clear from the outset or where the information sought is highly specialized or nuanced. The recursive nature of the process allows for continuous learning and adaptation to the user's requirements, often resulting in improved satisfaction with the search outcomes.

Adaptive Retrieval

Adaptive retrieval methods, exemplified by Flare and Self-RAG [Jiang *et al.*, 2023b, Asai *et al.*, 2023], refine the RAG framework by enabling LLMs to actively determine the optimal moments and content for retrieval, thus enhancing the efficiency and relevance of the information sourced.

These methods are part of a broader trend wherein LLMs employ active judgment in their operations, as seen in model agents like AutoGPT, Toolformer, and Graph-Toolformer [Yang *et al.*, 2023c, Schick *et al.*, 2023, Zhang, 2023]. Graph-Toolformer, for instance, divides its retrieval process into distinct steps where LLMs proactively use retrievers, apply Self-Ask techniques, and employ few-shot prompts to initiate search queries. This proactive stance allows LLMs to decide when to search for necessary information, akin to how an agent utilizes tools.

WebGPT [Nakano *et al.*, 2021] integrates a reinforcement learning framework to train the GPT-3 model in autonomously using a search engine during text generation. It navigates this process using special tokens that facilitate actions such as search engine queries, browsing results, and citing references, thereby expanding GPT-3's capabilities through the use of external search engines.

Flare automates timing retrieval by monitoring the confidence of the generation process, as indicated by the probability of generated terms [Jiang *et al.*, 2023b]. When the probability falls below a certain threshold would activates the retrieval system to collect relevant information, thus optimizing the retrieval cycle.

Self-RAG [Asai *et al.*, 2023] introduces "reflection tokens" that allow the model to introspect its outputs. These tokens come in two varieties: "retrieve" and "critic". The model autonomously decides when to activate retrieval, or alternatively, a predefined threshold may trigger the process. During retrieval, the generator conducts a fragment-level beam search across multiple paragraphs to derive the most coherent sequence. Critic scores are used to update the subdivision scores, with the flexibility to adjust these weights during inference, tailoring the model's behavior. Self-RAG's design obviates the need for additional classifiers or reliance on Natural Language Inference (NLI) models, thus streamlining the decision-making process for when to engage retrieval mechanisms and improving the model's autonomous judgment capabilities in generating accurate responses.

LLM optimization has received significant attention due to its increasing prevalence. Techniques such as prompt engineering, Fine-Tuning (FT), and RAG each have distinct characteristics, visually represented in Figure 6. While prompt engineering leverages a model's inherent capabilities, optimizing LLMs often requires the application of both RAG and FT methods. The choice between RAG and FT should be based on the specific requirements of the scenario and the inherent properties of each approach. A detailed comparison of RAG and FT is presented in Table 1.

6.4 RAG vs Fine-Tuning

RAG is like giving a model a textbook for tailored information retrieval, perfect for specific queries. On the other hand, FT is like a student internalizing knowledge over time, better for replicating specific structures, styles, or formats. FT can improve model performance and efficiency by reinforcing base model knowledge, adjusting outputs, and teaching complex instructions. However, it is not as good for integrating new knowledge or rapidly iterating new use cases.

The two methods, RAG and FT, are not mutually exclusive and can be complementary, augmenting a model's capabilities at different levels. In some cases, their combined use may yield optimal performance. The optimization process involving RAG and FT can necessitate multiple iterations to achieve satisfactory results.

7 RAG Evaluation

The rapid advancement and growing adoption of RAG in the field of Natural Language Processing (NLP) have propelled the evaluation of RAG models to the forefront of research in the LLMs community. The primary objective of this evaluation is to comprehend and optimize the performance of RAG models across diverse application scenarios.

Historically, RAG models assessments have centered on their execution in specific downstream tasks. These evaluations employ established metrics suitable to the tasks at hand. For instance, question answering evaluations might rely on EM and F1 scores [Wang *et al.*, 2023a, Shi *et al.*, 2023, Feng *et al.*, 2023, Ma *et al.*, 2023a], whereas fact-checking tasks often hinge on accuracy as the primary metric [Lewis *et al.*, 2020, Izacard *et al.*, 2022, Shao *et al.*, 2023]. Tools like RALLE, designed for the automatic evaluation of RAG applications, similarly base their assessments on these task-specific metrics [Hoshi *et al.*, 2023]. Despite this, there is a notable paucity of research dedicated to evaluating the distinct characteristics of RAG models, with only a handful of related studies.

The following section shifts the focus from task-specific evaluation methods and metrics to provide a synthesis of the existing literature based on their unique attributes. This exploration covers the objectives of RAG evaluation, the aspects along which these models are assessed, and the benchmarks and tools available for such evaluations. The aim is to offer a comprehensive overview of RAG model evaluation, outlining the methodologies that specifically address the unique aspects of these advanced generative systems.

7.1 Evaluation Targets

The assessment of RAG models mainly revolves around two key components: the retrieval and generation modules. This division ensures a thorough evaluation of both the quality of context provided and the quality of content produced.

Retrieval Quality

Evaluating the retrieval quality is crucial for determining the effectiveness of the context sourced by the retriever component. Standard metrics from the domains of search engines, recommendation systems, and information retrieval systems are employed to measure the performance of the RAG retrieval module. Metrics such as Hit Rate, MRR, and NDCG are commonly utilized for this purpose [Liu, 2023, Nguyen, 2023].

Generation Quality

The assessment of generation quality centers on the generator's capacity to synthesize coherent and relevant answers from the retrieved context. This evaluation can be categorized based on the content's objectives: unlabeled and labeled content. For unlabeled content, the evaluation encompasses the faithfulness, relevance, and non-harmfulness of the generated answers. In contrast, for labeled content, the focus is on the accuracy of the information produced by the