Translated on December 22, 2023 Published on December 18, 2023

Retrieval-augmented generation techniques for large language models: A review

原文：Retrieval-Augmented Generation for Large Language Models: A Survey

Main authors: Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, with special thanks to **Jiawei Sun** and **Haofen Wang** Affiliations: 1. Shanghai Institute for Intelligent and Autonomous Systems, Tongji University; 2. Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University; 3. School of Design and Innovation, Tongji University Contact email: **gaoyunfan1602@gmail.com**

In this review, we focus on retrieval-enhanced generation techniques for large language models. This technique enhances the ability of large language models to handle complex queries and generate more accurate information by combining retrieval mechanisms. We start from relevant research teams from Tongji University and Fudan University and comprehensively analyze the latest progress and future trends in this field.

Summary

Although large language models (LLMs) have shown powerful capabilities, they still face challenges in practical applications, such as accuracy, knowledge update speed, and answer transparency. Retrieval-Augmented Generation (RAG) refers to retrieving relevant information from an external knowledge base before answering questions using a large language model.

RAG 被证明能显著提升答案的准确性，并特别是在知识密集型任务上减少模型的错误输出。通过引用信息来源，用户可以核实答案的准确性，从而增强对模型输出的信任。

Furthermore, RAGs facilitate rapid knowledge updating and the introduction of expertise in specific areas.

RAG effectively combines the parameterized knowledge of large language models with non-parameterized external knowledge bases, becoming one of the key methods for implementing large language models. This article outlines the development model of RAG in the era of large language models and summarizes three models: elementary RAG, advanced RAG, and modular RAG. Next, this article sorts out the three main components of RAG: retriever, generator, and enhancement method, as well as the key technologies of each part. At the same time, this article discusses how to evaluate the effectiveness of the RAG model, introduces two evaluation methods, emphasizes key evaluation indicators and capabilities, and demonstrates the latest automatic evaluation framework. Finally, the article introduces possible future research directions from three aspects: vertical optimization, horizontal scalability, and RAG's technology stack and ecosystem.

1 Introduction

Large Language Models (LLMs) have outperformed any previous models in the field of Natural Language Processing (NLP).

Large language models such as the GPT series of models [Brown *et al.* , 2020, OpenAI, 2023], the LLama series of models [Touvron *et al.* , 2023], and Gemini [Google, 2023] have demonstrated excellent language mastery and knowledge comprehension capabilities on multiple evaluation benchmarks, even surpassing multiple human evaluation benchmarks [Wang *et al.* , 2019, Hendrycks *et al.* , 2020, Srivastava *et al.* , 2022].

However, large language models also have many shortcomings.

For example, they may produce inaccurate information [Zhang *et al.* , 2023b] and exhibit knowledge gaps when processing domain-specific or highly specialized queries [Kandpal *et al.* , 2023]. Large language models may not be able to provide accurate answers when the required information is beyond the scope of the model training data or when fresh data is required. This limitation is particularly challenging when deploying generative AI into real-world production environments, as relying solely on black-box large language models may not be sufficient.

Neural networks usually parameterize knowledge by fine-tuning the model to adapt to specific domains or proprietary information. Although this approach has achieved remarkable results, it consumes a lot of computing resources, is costly, and requires professional technical knowledge, making it difficult to adapt to the ever-changing information environment. In this process, parameterized knowledge and non-parametric knowledge each play their own roles. Parametric knowledge is obtained by training a large language model (LLM) and stored in the weights of the neural network. It represents the model's understanding and generalization ability of the training data and is the basis for generating responses. Non-parametric knowledge is stored in external knowledge sources, such as vector databases, and is not directly incorporated into the model, but rather serves as an updateable supplementary information. Non-parametric knowledge enables the large language model to access and utilize the latest or domain-specific information, improving the accuracy and relevance of responses.

Purely parametric language models (LLMs) store the world knowledge they learn from a large corpus in the model parameters. However, this model has limitations. First, it is difficult to retain all the knowledge in the training corpus, especially for less common and specific knowledge. Second, since the model parameters cannot be updated dynamically, the parameterized knowledge may become outdated over time. Finally, the increase in parameters will lead to an increase in the computational cost of training and inference. To address the limitations of purely parametric models, language models can take a semi-parametric approach, combining a non-parametric corpus database with a parametric model. This approach is called Retrieval-Augmented Generation (RAG).

The term "Retrieval-Augmented Generation" (RAG) was first proposed by [Lewis *et al.* , 2020]. It combines a pre-trained retriever and a pre-trained sequence-to-sequence model (generator) through end-to-end fine-tuning to capture knowledge in a more interpretable and modular way. Before the emergence of large models, RAG mainly focused on directly optimizing end-to-end models. For example, it is common to use vector-based dense passage retrieval (DPR) [Karpukhin *et al.* , 2020] on the retrieval side and train smaller models on the generation side.

Due to the small number of overall parameters, the retriever and generator are often trained or fine-tuned synchronously end-to-end [Izacard *et al.* , 2022].

With the emergence of large language models such as ChatGPT, generative language models have shown excellent performance in various language tasks and have received increasing attention and application [Bai *et al.* , 2022, OpenAI, 2023, Touvron *et al.* , 2023, Google, 2023].

However, large language models (LLMs) still face challenges such as hallucination errors [Yao *et al.* , 2023, Bang *et al.* , 2023], knowledge updating, and data correlation issues.

These issues affect the reliability of large language models and pose challenges in some serious task scenarios, especially in knowledge-intensive tasks that require extensive knowledge, such as open-domain question answering [Chen and Yih, 2020, Reddy *et al.* , 2019, Kwiatkowski *et al.* , 2019] and commonsense reasoning [Clark *et al.* , 2019, Bisk *et al.* , 2020].

The implicit knowledge in the model parameters may be incomplete or insufficient.

Subsequent studies have found that the above problems can be effectively alleviated by introducing RAG into the contextual learning (ICL) of large models. This method has obvious and easy-to-implement effects. During the reasoning process, RAG dynamically retrieves information from external knowledge sources and uses these retrieved data as a reference for organizing answers. This greatly improves the accuracy and relevance of the answers, and effectively solves the problem of hallucination errors in large language models. This technology has rapidly attracted attention since the emergence of large language models and has become one of the cutting-edge technologies for improving the effectiveness of chatbots and enhancing the practicality of large language models. By separating factual knowledge from the training parameters of large language models, RAG cleverly combines the power of the generative model and the flexibility of the retrieval module, providing an effective solution to the problems of incomplete and insufficient knowledge inherent in purely parameterized models.

This paper systematically reviews and analyzes the existing research methods and future development paths of retrieval augmented generation (RAG), and summarizes them into three paradigms: primary RAG, advanced RAG, and modular RAG. Then, the paper provides a comprehensive overview of the three core components: retrieval, augmentation, and generation, emphasizing the improvement direction and current technical features of RAG. In the chapter on augmentation methods, the existing work is divided into three aspects: the augmentation stage of RAG, augmented data sources, and augmentation process. In addition, the paper also outlines the evaluation system and applicable scenarios related to RAG. By reading this article, readers can have a more comprehensive and systematic understanding of the concepts of large language models and retrieval augmented generation, and have an in-depth understanding of the development history and key technologies of knowledge retrieval augmentation, so as to be able to distinguish the advantages and disadvantages of different technologies, find applicable scenarios, and explore typical application cases in practice. It is worth mentioning that Feng et al. [2023b] systematically reviewed the methods, applications, and future trends of combining large language models with knowledge in their research, with a special focus on knowledge editing and retrieval augmentation methods. Zhu et al. [2023] present recent progress in enhancing retrieval systems for large language models, with a particular focus on the retrieval system itself.

Meanwhile, Asai et al. [2023a] analyzed and explained the key steps of retrieval-based language models in terms of “what”, “when”, and “how”. In contrast, the purpose of this paper is to systematically outline the entire process of retrieval-augmented generation (RAG), with a special focus on the research of enhancing the generation of large language models through knowledge retrieval.

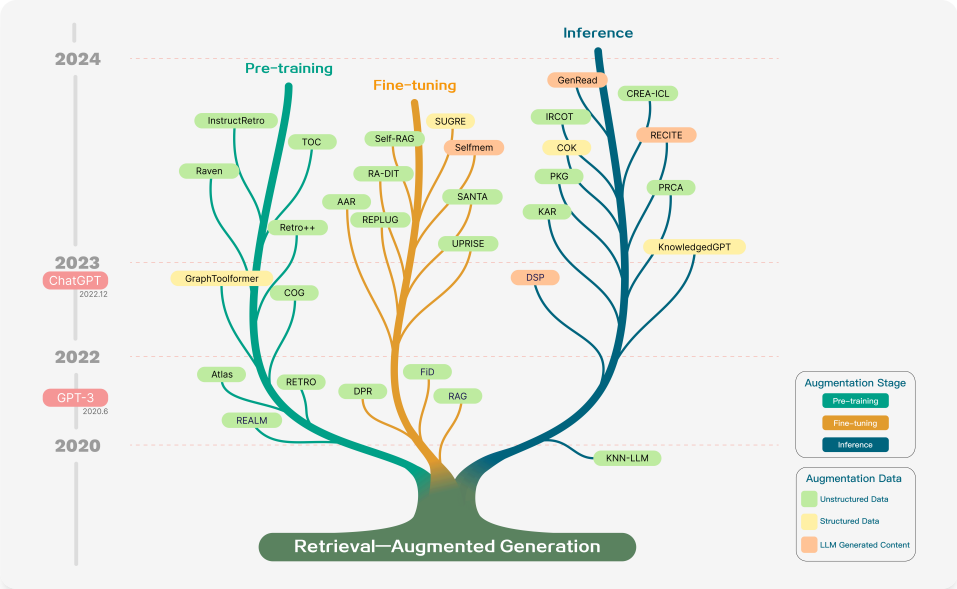

Figure 1: Timeline of existing RAG studies. The timeline is primarily determined by publication dates.

Figure 1 shows the development of RAG algorithms and models. In terms of the timeline, most of the RAG-related research appeared after 2020, especially after the release of ChatGPT in December 2022, which became an important turning point. After the release of ChatGPT, research in the field of natural language processing entered the era of large models. Primary RAG technology quickly gained attention, and the number of related studies surged. In terms of enhancement strategies, since the concept of RAG was proposed, reinforcement research in the pre-training and supervised fine-tuning stages has been ongoing. However, in the era of large language models, reinforcement research in the reasoning stage has begun to increase. This is mainly because the training cost of high-performance large models is high. Researchers have tried to integrate external knowledge into model generation in a cost-effective manner by adding RAG modules in the reasoning stage.

In exploring the use of augmented data, early RAG focused on the application of unstructured data, especially in the open domain question-answering environment. Over time, the sources of knowledge retrieved by RAG have become more extensive, including high-quality data. As a source of knowledge, these data effectively avoid problems such as large models mistakenly adopting wrong information and making wrong assumptions (i.e. "hallucinations"). It is worth mentioning that RAG has also begun to use structured knowledge, such as knowledge graphs. Recently, self-retrieval has become a hot topic, which refers to using the knowledge base of a large language model itself to improve its performance.

The following chapters of this paper are organized as follows: Chapter 2 introduces the background knowledge of RAG. Chapter 3 discusses the mainstream mode of RAG. Chapter 4 analyzes the retriever function in RAG. Chapter 5 focuses on how the generator in RAG works. Chapter 6 emphasizes the data augmentation method in RAG. Chapter 7 explains the evaluation system of RAG. Chapter 8 looks forward to the future development direction of RAG. Finally, in Chapter 9, we summarize the main contents of this survey.

2 Background

In this section, we define RAG, a model optimization technique, and compare it with other optimization techniques such as fine-tuning.

2.1 Definition

Against the backdrop of technological advances, the concept of RAG has also expanded. In the field of Large Language Models, RAG specifically refers to a mode in which the model first looks for relevant information from a vast document library when answering questions or generating text. The model then uses this found information to generate answers or text, thereby improving the accuracy of its predictions. RAG's approach eliminates the need for developers to retrain the entire large model for each specific task. They can simply add a knowledge base to the model, thereby increasing the information input to the model and thus improving the accuracy of the answer. RAG is particularly suitable for tasks that require a lot of knowledge. In short, the RAG system consists of two main stages:

1. Use encoding models (such as BM25, DPR, ColBERT, etc.) to find relevant documents based on the question [Robertson *et al.* , 2009, Karpukhin *et al.* , 2020, Khattab and Zaharia, 2020].

2. Generation phase: Based on the found context, the system generates text.

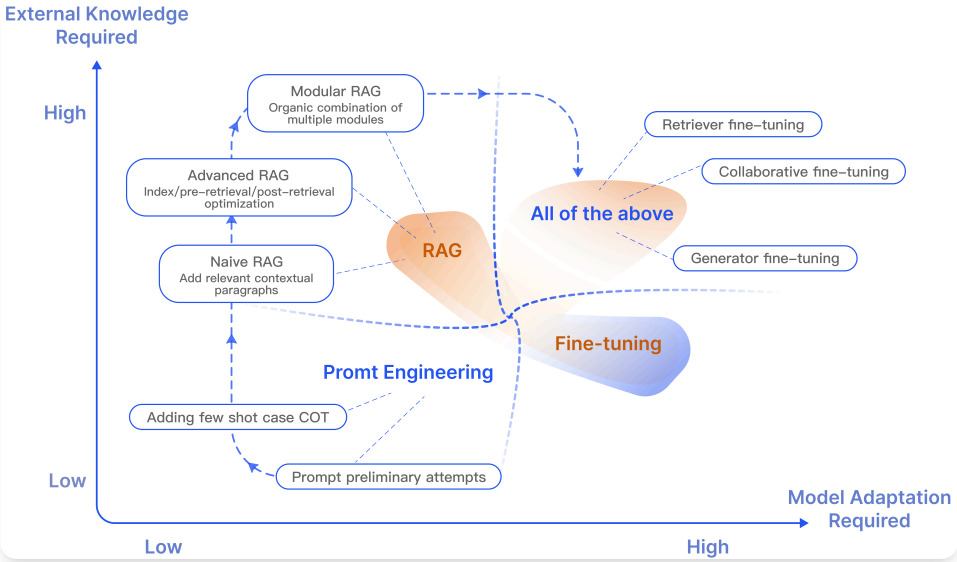

Figure 2: Comparison of RAG and other model optimization methods

2.2 RAG and fine-tuning

In the optimization process of large language models, in addition to RAG, fine-tuning is also an important technology.

Think of RAG as giving the model a textbook and asking it to find information based on a specific question. This approach works well when the model needs to answer a specific question or perform a specific information retrieval task. But RAG is not suitable for teaching models to understand a wide range of domains or learn new languages, formats, or styles.

Whereas fine-tuning is more like allowing students to absorb knowledge through extensive learning.

Fine-tuning is useful when a model needs to mimic a specific structure, style, or format. It can improve the performance of an un-fine-tuned model and make interactions more efficient.

Fine-tuning is particularly useful for reinforcing the model's existing knowledge, adjusting or customizing the model's output, and giving the model complex instructions. However, fine-tuning is not suitable for adding new knowledge to the model or for use in situations where new scenarios need to be iterated quickly.

The process of fine-tuning is like allowing students to absorb knowledge through deep and persistent learning. This method is suitable when the model needs to accurately imitate a special structure, artistic style or format. Fine-tuning can make the model perform better than the un-fine-tuned model and improve the interaction efficiency. It is particularly suitable for highlighting the existing knowledge in the model's basic knowledge base, adjusting or customizing the model output, and training the model with complex guidance. However, fine-tuning is not suitable for adding completely new knowledge to the model or dealing with situations that require rapid iteration of new scenarios. A specific comparison between RAG (Retrieval-Augmented Generation) and fine-tuning can be seen in Table 1.

RAG and fine-tuning can complement each other rather than exclude each other, thereby enhancing the capabilities of the model at different levels. In certain cases, combining these two methods can achieve the best state of model performance. The entire process of optimization using RAG and fine-tuning may require multiple rounds of iterations to achieve satisfactory results.

目前的研究已经表明，检索增强生成 (Retrieval-Augmented Generation, RAG) 在优化大语言模型 (Large Language Model) 方面，相较于其他方法具有显著的优势【Shuster *et al.*, 2021; Yasunaga *et al.*, 2022; Wang *et al.*, 2023c; Borgeaud *et al.*, 2022】：

- RAG improves the accuracy of answers by associating external knowledge, effectively reducing false information in the language model and making the generated answers more accurate and credible.
- The use of retrieval technology can identify the latest information, which gives RAG a clear advantage in maintaining the timeliness and accuracy of answers compared to traditional language models that only rely on training data.
- One of the strengths of RAG is its transparency. By citing the source of the information, users can verify the accuracy of the answers, which increases trust in the model output.
- RAG is highly customizable. By indexing text corpora related to specific fields, RAG can provide professional knowledge support for different fields.
- In terms of security and privacy management, RAG enables greater control over data usage through roles and security controls set in the database. In contrast, a fine-tuned model may be less explicit in managing data access permissions.
- RAG is more scalable in processing large-scale datasets. It does not need to update all parameters and create new training sets, so it has more advantages in terms of economic efficiency.
- Finally, RAG provides more trustworthy results. RAG extracts deterministic results from the latest data, while fine-tuned models may produce misinformation and inaccuracies when processing dynamic data, lacking transparency and credibility.

3 RAG Framework

The RAG research paradigm is constantly evolving. This chapter focuses on the development of the RAG research paradigm. We divide it into three types: naive RAG, advanced RAG, and modular RAG. Although early RAG performed well in terms of cost-effectiveness and outperformed traditional large language models (LLMs), it still faces many challenges. Advanced RAG and modular RAG are designed to address specific deficiencies of the original RAG (Naive RAG).

Feature Comparison	RAG	Fine-tuning
Knowledge Update	Directly update the retrieval knowledge base to ensure that information is continuously updated without the need for frequent retraining, which is very suitable for dynamically changing data environments.	Storing static data requires retraining for knowledge and data updates.
External knowledge	Skilled in leveraging external resources, particularly suited to working with documents or other structured/unstructured databases.	It can be used to keep externally learned knowledge from pre-training consistent with a large language model, but may not be practical for frequently changing data sources.
data processing	The requirements for data processing and operation are extremely low.	依赖于构建高质量的数据集，有限的数据集可能无法显著提高性能。
Model customization	Focuses on information retrieval and incorporating external knowledge, but may not adequately customize model behavior or writing style.	Allows tailoring LLM behavior to specific styles or terminology, writing style, or specific domain knowledge.
Explainability	The answers can be traced back to the specific data source, providing higher explainability and traceability.	Like a black box, it is not always clear why the model reacts in a certain way, and interpretability is relatively low.
Computing resources	Computing resources are needed to support search strategies and database-related technologies. Integration and updating of external data sources need to be maintained.	It is necessary to prepare and organize high-quality training datasets, determine fine-tuning targets, and provide corresponding computing resources.
Delay Requirements	Because it involves data retrieval, it may cause higher latency.	The fine-tuned Large Language Model (LLM) can respond directly without retrieval, reducing latency.

Reduce hallucinations	Since each response is based on actual evidence retrieved, it is inherently less prone to false or fictitious responses.	Training a model on domain-specific data can help reduce hallucinations, but hallucinations may still occur when faced with inputs that it was not trained on.
Ethical and Privacy Issues	Storing and retrieving text from external databases may raise ethical and privacy concerns.	Sensitive content in training data may raise ethical and privacy issues.

Table 1: Comparison between RAG and fine-tuning

3.1 Naive RAG

Naive RAG represents an early research method that quickly gained prominence after ChatGPT became widely used. The Naive RAG pipeline includes traditional indexing, retrieval, and generation steps. Naive RAG is also summarized as a "retrieval"-"reading" framework [Ma *et al.* , 2023a].

index

Refers to the process of obtaining data from the data source and building an index in an offline state. Specifically, building a data index includes the following steps:

1. **Data indexing:** includes cleaning and extracting raw data, converting files in different formats such as PDF, HTML, Word, Markdown, etc. into plain text.
2. **Chunking:** Split the loaded text into smaller pieces. Since language models have limited ability to process context, the text needs to be divided into the smallest possible chunks.
3. **Embedding and indexing:** This phase involves encoding text into vectors through a language model. The resulting vectors will be used in subsequent retrieval processes to calculate their similarity with the question vector. Since a large amount of text needs to be encoded and questions are encoded in real time when users ask questions, the embedding model requires high-speed reasoning capabilities, and the model's parameter scale should not be too large. After the embedding is completed, the next step is to create an index to store the original corpus chunks and embeddings as key-value pairs for fast and frequent searches in the future.

Search:

Based on the user input, the query content is converted into a vector using the same encoding model as in the first stage. The system calculates the similarity between the question vector and the document block vector in the corpus, and selects the top K most relevant document blocks as supplementary background information for the current question based on the similarity level.

generate:

The given question and the relevant documents are combined into a new prompt. Subsequently, the Large Language Model (LLM) is tasked with answering the question based on the provided information. Depending on the needs of different tasks, the model can be asked to rely on its own knowledge base or answer questions based only on the given information. If there is historical dialogue information, it can also be incorporated into the prompt to support multi-round dialogue.

Challenges of Simple RAG:

Naive RAG faces challenges in three main aspects: retrieval quality, response generation quality, and the enhancement process.

- **Retrieval quality:** There are many problems in this area. The most important problem is low precision, that is, not all document blocks in the retrieval set are relevant to the query content, which may lead to incorrect or incoherent information. The second is the low recall problem, that is, the failure to retrieve all relevant document blocks makes it impossible for the large language model to obtain enough background information to synthesize the answer. In addition, outdated information is also a challenge, because redundant or outdated data may lead to inaccurate retrieval results.
- **Response generation quality:** The issues here are equally varied. The most prominent issue is misinformation, where the model fabricates answers without sufficient context. Another issue is irrelevant responses, where the model generates answers that are not specific to the query. Furthermore, generating harmful or biased responses is also a problem.
- **Enhancement process:** Ultimately, the enhancement process faces several important challenges. Of particular importance is how to effectively incorporate the context of the retrieved text into the current generation task. If not handled properly, the generated content may appear disorganized. When multiple retrieved texts contain similar information, redundancy and repetition become problems, which may lead to duplication of generated content. In addition, how to judge the importance or relevance of multiple retrieved texts to the generation task is very challenging, and the enhancement process needs to properly evaluate the value of each text. The retrieved content may have different writing styles or tones, and the enhancement process needs to reconcile these differences to ensure the consistency of the final output. Finally, the generation model may rely too much on the enhanced information, resulting in the generated content simply repeating the retrieved information without new value or comprehensive information.

3.2 Advanced RAG

To overcome the limitations of Naive RAG, Advanced RAG has made targeted improvements. In terms of retrieval generation quality, Advanced RAG introduces pre-retrieval and post-retrieval methods. It also optimizes indexes through sliding windows, fine-grained segmentation, and metadata to solve the indexing problems encountered by Naive RAG. At the same time, Advanced RAG also proposes a variety of methods to optimize the retrieval process. In terms of specific implementation, Advanced RAG can be adjusted in a pipelined or end-to-end manner.

Pre-search processing

- **Optimizing data indexing** Optimizing data indexing aims to improve the quality of indexed content. Currently, five main strategies are used: improving index data

granularity, optimizing index structure, adding metadata, alignment optimization, and hybrid retrieval.

1. **Improve data granularity:** Pre-indexing optimization is to improve the standardization and consistency of text, ensure accurate facts and rich context, and thus guarantee the performance of the RAG system. Text standardization aims to remove irrelevant information and special characters to improve retrieval efficiency. In terms of ensuring consistency, the main focus is to eliminate ambiguity in terms and entities, remove duplicate or redundant information, and simplify the retrieval process. The accuracy of facts is critical, and each piece of data should be verified as much as possible. In terms of maintaining context, the system is adapted to the interactive context of the real world by adding domain-specific annotations and continuously updating through user feedback loops. Considering time sensitivity, outdated documents should be updated. In general, the focus of optimizing indexed data is on clarity, context, and correctness to improve the efficiency and reliability of the system. Here are some best practices.
2. **Optimize index structure:** This can be achieved by adjusting the chunk size, changing the index path, and incorporating graph structure information. The approach to adjusting the chunk size is to collect as much relevant context as possible while minimizing interference. Chunk size is a key parameter when building a RAG system. Different evaluation frameworks are used to compare different chunk sizes. **LlamaIndex** uses GPT4 to evaluate the fidelity and relevance of data, and the LLaMA [Touvron *et al.* , 2023] index can automatically evaluate the effects of different chunking methods. Querying across multiple index paths is closely related to previous metadata filtering and chunking methods, and may involve querying in different indexes at the same time. Standard indexes can be used for specific queries, or independent indexes can be used to search or filter based on metadata keywords (such as the "date" index). Introducing a graph structure is to convert entities into nodes and the relationships between them into associations, which can improve the accuracy of multi-hop problems by leveraging the relationships between nodes. Using graph data indexes can improve the relevance of retrieval.
3. **Add metadata information:** The core of this strategy is to embed citation metadata, such as date and purpose (for filtering), into the data block. Adding metadata such as chapters and reference subsections is helpful for improving retrieval efficiency. When the index is divided into multiple blocks, how to search efficiently becomes the key. Preliminary screening through metadata can improve the efficiency and accuracy of retrieval.
4. **Alignment optimization:** This strategy focuses on alignment issues and differences between different documents. The core idea of alignment is to introduce "hypothetical questions", that is, to create questions that are suitable for answering with each document and combine these questions with the documents. This approach helps to solve alignment issues and inconsistencies between documents.
5. **混合检索：** 混合检索的优势在于它结合了不同检索技术的长处。它智能地融合了关键词搜索、语义搜索和向量搜索等多种技术，适应不同类型的查询需求，确保能够一致地检索到最相关和内容丰富的信息。混合检索作为检索策略的重要补充，能够显著提升 RAG 流程的整体性能。

Embedding

- **微调嵌入：** 微调嵌入模型的调整直接影响到 RAG 的有效性。微调的目的是让检索到的内容与查询之间的相关性更加紧密。微调嵌入的作用可以比作在语音生成前对“听觉”进

行调整，优化检索内容对最终输出的影响。通常，微调嵌入的方法可以分为针对特定领域上下文的嵌入调整和检索步骤的优化。特别是在处理不断变化或罕见术语的专业领域，这些定制化的嵌入方法能够显著提高检索的相关性。BGE[BAAI, 2023]嵌入模型是一个经过微调的高性能嵌入模型，例如由 BAAI 3 开发的 BGE-large-EN。为了对 BGE 模型进行微调，首先使用诸如 gpt-3.5-turbo 这样的大语言模型（LLM）根据文档块制定问题，其中问题和答案（文档块）构成了微调过程中的训练对。

- **Dynamic Embedding** : Unlike static embedding, dynamic embedding adjusts according to the context in which the word appears, providing different vector representations for each word. For example, in Transformer models such as BERT, the embedding of the same word will change depending on the surrounding vocabulary. Studies have found that in OpenAI's text-embeddingada-002 model, unexpectedly high cosine similarities often occur when the text length is less than 5 tokens. Ideal embeddings should contain enough context to ensure good results. OpenAI's embeddings-ada-02 is developed based on the principles of large language models (such as GPT). It is more complex than traditional static embedding models and can capture a certain degree of context. Although it performs well in contextual understanding, it may not be as context-sensitive as the latest large language models (such as GPT-4).

Post-retrieval process

After retrieving valuable context from the database, merging it with the query content into the large language model (LLM) can be challenging. Showing all relevant documents to the LLM at once may exceed the context window it can handle. Concatenating multiple documents into a lengthy query is not only inefficient, but also introduces noise that affects the LLM's focus on key information. Therefore, additional processing of the retrieved content is required to address these issues.

- **ReRank** : Reranking, placing the most relevant information at the front and back edges of the prompt, is a straightforward approach. This idea has been used in frameworks such as LlamaIndex, LangChain, and HayStack [Blagojevi, 2023]. For example, Diversity Ranker reranks documents based on their diversity, while LostInTheMiddleRanker alternates between placing the best documents at the beginning and end of the context window. Meanwhile, to address the challenges of vector-based semantic similarity simulation search, methods such as cohereAI rerank [Cohere, 2023], bgererank5, or LongLLMLingua [Jiang *et al.* , 2023a] recalculate the semantic similarity between relevant text and the query.
- **Prompt compression** : Studies have shown that noise in retrieved documents can adversely affect RAG performance. In the later stages of processing, we focus on compressing insignificant context, highlighting key paragraphs, and shortening the overall context length. For example, methods such as Selective Context [Litman *et al.* , 2020] and LLMLingua [Anderson *et al.* , 2022] use small language models to calculate the mutual information or perplexity between prompts to estimate the importance of each element.

  However, in the context of RAG or long-length contexts, these methods may miss key information.

  Recomp [Xu *et al.* , 2023a] addresses this problem by training compressors of varying degrees of sophistication.

  When dealing with long contexts [Xu *et al.* , 2023b], this approach handles large amounts of context by decomposing and compressing it, while “wandering through

the memory maze" [Chen *et al.* , 2023a] designs a hierarchical summary tree to enhance the large language model's (LLM) perception of key information.

- **RAG pipeline optimization:** The optimization of the retrieval process aims to improve the efficiency and information quality of the RAG system. Current research focuses on intelligently combining different search techniques, optimizing retrieval steps, introducing cognitive backtracking concepts, flexibly applying diverse query strategies, and leveraging embedded similarity. These efforts jointly pursue the balance between efficiency and contextual information richness in RAG retrieval.
- **混合搜索的探索：** RAG 系统巧妙结合了基于关键词、语义以及向量的多种搜索技术。这种综合方法让 RAG 系统能够应对不同的查询类型和信息需求，有效地获取最相关且内容丰富的信息。混合搜索作为一种强大的补充手段，显著提升了 RAG 流程的整体表现。
- **Recursive retrieval and query engine:** In the RAG system, the use of recursive retrieval and advanced query engines is another effective means to improve retrieval results. The first step of recursive retrieval is to obtain small document blocks in the initial stage to capture key semantics. Subsequently, the process provides larger document blocks, providing richer contextual information for the large language model (LM). This dual retrieval strategy ensures efficiency while providing in-depth contextual responses.
- **StepBack-prompt method: The StepBack-prompt method [Zheng** *et al.* , 2023] integrated into the RAG process enables the large language model (LLM) to take a step back when dealing with specific cases and think about the general concepts or principles behind them. The study found that this method combined with backward prompts performed well in dealing with a variety of complex and reasoning-intensive tasks, fully demonstrating its good compatibility with RAG. This method can be used for both backward prompt answer generation and the final question-answering phase.
- **Subquery:** Depending on the scenario, we can adopt a variety of query strategies, such as using the query engine provided by frameworks such as LlamaIndex, tree query, vector query, or basic block sequence query.
- **HyDE method:** This method is based on the assumption that the answers generated by a large language model (LLM) may be closer in embedding space than the direct query. HyDE first generates a hypothetical document (answer) in response to the query, then embeds it, and uses this embedding to retrieve real documents similar to the hypothetical document. This method emphasizes the embedding similarity between answers rather than simply relying on the embedding similarity of queries. However, in some cases, especially when the language model is not familiar enough with the topic, it may lead to an increase in false instances.

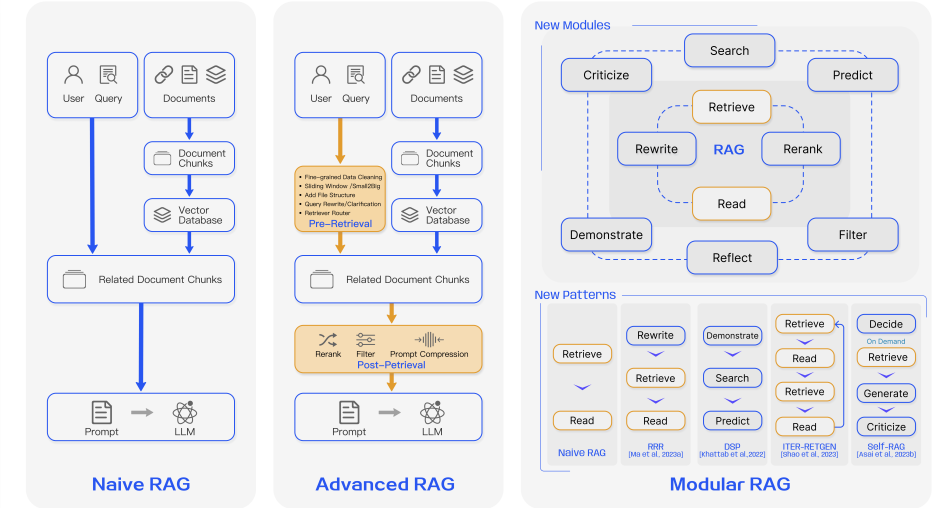

Figure 3: Comparison of three RAG paradigms

Modular RAG

The modular RAG structure breaks the traditional "raw RAG" framework, which originally involved indexing, retrieval, and generation, and now provides wider diversity and higher flexibility. It not only integrates various methods to enrich functional modules, such as adding a search module to similarity retrieval, but also adopts a fine-tuning strategy in the retriever [Lin *et al.* , 2023]. Special problems have also spawned reconstructed RAG modules [Yu *et al.* , 2022], as well as iterative methods like [Shao *et al.* , 2023]. This modular RAG paradigm is gradually becoming a trend in the RAG field, supporting everything from serialized processes to end-to-end training methods across multiple modules. The comparison of the three RAG paradigms is shown in detail in Figure 3.

New Modules

- **Search module:** Different from the query and general similarity retrieval between simple/advanced RAG, the search module in this specific scenario combines methods for searching directly in the (attached) corpus. These methods include code generated by large language models (LLMs), query languages such as SQL, Cypher, or other customized tools. The search data sources are diverse, covering search engines, text data, tabular data, or knowledge graphs [Wang *et al.* , 2023c].
- **Memory module:** This module makes full use of the memory function of the large language model itself to guide information retrieval. Its core principle is to find the memory that best matches the current input. For example, Self-mem [Cheng *et al.* , 2023b] creates an infinite memory pool that combines "original questions" and "dual questions" by iteratively using a generative model of enhanced retrieval. This generative model of enhanced retrieval can use its own output to improve itself, making the text closer to the data distribution during reasoning, rather than relying solely on training data [Wang *et al.* , 2022a].
- **Additional generation module:** In the face of redundancy and noise in the retrieval content, this module generates the necessary context through a large language model instead of directly retrieving it from the data source [Yu *et al.* , 2022]. In this way, the content generated by the large language model is more likely to contain information relevant to the retrieval task.
- **Task Adaptation Module:** This module is dedicated to adapting RAG to various downstream tasks. For example, UPRISE[Cheng *et al.* , 2023a] can automatically retrieve appropriate prompts for a given zero-shot task input from a pre-built data pool, thereby improving the generality between tasks and models.

PROMPTAGATOR[Dai *et al.* , 2022] uses a large language model as a few-shot query generator to create a task-specific retriever based on the generated data. Leveraging the generalization power of a large language model, PROMPTAGATOR makes it possible to create an end-to-end retriever specialized for a specific task with just a few examples.

- **Alignment module:** In the application of RAG, the alignment between query and text has always been a key factor affecting the effect. In the development of modular RAG, researchers found that adding a trainable Adapter module to the retriever can effectively solve the alignment problem. For example, PRCA [Yang *et al.* , 2023b] trained a context adapter driven by a large language model reward through reinforcement learning, which is located between the retriever and the generator. Through the reinforcement learning stage in the labeled autoregressive strategy, it is able to optimize the retrieved information and maximize the reward during the reinforcement learning process.

  AAR [Yu *et al.* , 2023b] proposes a general plugin that learns language model preferences from large language models (LLMs) from known sources and uses this knowledge to assist large language models that are unknown or not yet jointly fine-tuned.

  RRR[Ma *et al.* , 2023a] designs a reinforcement learning-based module that rewrites queries so that they better match the documents in the corpus.

- **Verification module:** In the real world, we cannot always guarantee the reliability of the retrieved information. Retrieving irrelevant data may cause large language models to produce wrong information. Therefore, an additional verification module can be added after retrieving documents to evaluate the relevance between the retrieved documents and the query, which can improve the robustness of RAG[Yu *et al. , 2023a]*.

New Mode Modular

RAG's organizational approach is highly flexible, and the modules in the RAG process can be replaced or reconfigured according to the context of a specific problem. In the basic Naive RAG, two core modules, retrieval and generation (some literature calls them reading or synthesis modules), are included, making this framework highly adaptable and diverse. Current research focuses on two organizational modes: one is to add or replace modules, and the other is to adjust the workflow between modules.

- **Add or replace modules** In the strategy of adding or replacing modules, we retain the original retrieval-reading structure while adding new modules to enhance specific functions. RRR [Ma *et al.* , 2023a] proposed a rewrite-retrieve-reading process, in which the performance of the large language model (LLM) is used as a reward mechanism for the rewrite module in reinforcement learning. In this way, the rewrite module can adjust the retrieval query to improve the reader's performance in subsequent tasks. Similarly, we can also selectively replace modules in other methods, such as in generate-read [Yu *et al.* , 2022], the generation module of the large language model replaces the retrieval module. Recite-read [Sun *et al.* , 2022] transforms traditional external retrieval into retrieval from model weights, where the large language model first memorizes task-related information and then generates the output required to handle knowledge-intensive natural language processing tasks.

- **Adjusting the workflow between modules** In the area of adjusting the workflow between modules, the focus is on strengthening the interaction between the language model and the retrieval model. DSP[Khattab *et al.* , 2022] introduced the

display-search-prediction framework, treating the contextual learning system as an explicit program rather than a simple terminal task prompt to cope with knowledge-intensive tasks. ITER-RETGEN[Shao *et* al. , 2023] uses generated content to guide retrieval, and forms a retrieval-reading-retrieval-reading workflow by iteratively performing "retrieval-enhanced generation" and "generation-enhanced retrieval". Self-RAG[Asai *et al.* , 2023b] adopts the decision-retrieval-reflection-reading process and introduces a module for active judgment. This adaptive and diverse approach allows various modules to be dynamically organized in the Modular RAG framework.

4 Retriever

In the RAG (Retrieval-Augmented Generation) technique, “R” stands for retrieval, which is to retrieve the top k most relevant documents from a large knowledge base. However, building a high-quality retriever is challenging. In this chapter, we will explore three key questions: 1) How to obtain accurate semantic representation? 2) How to match the semantic space of query and document? 3) How to make the output of the retriever consistent with the preferences of the large language model (LLM)?

4.1 How to obtain accurate semantic representation?

In RAG, semantic space refers to the multidimensional space where queries and documents are mapped.

When performing retrieval, we evaluate in this semantic space. If the semantic expression is inaccurate, the impact on RAG will be disastrous. This section will introduce two methods to construct an accurate semantic space.

Block Optimization

The first step in processing external documents is to chunk them to get more detailed features. Then, these document chunks are embedded.

Embedding text chunks that are too large or too small may not achieve optimal results. Therefore, it is critical to find the optimal chunk size for your corpus documents to ensure accuracy and relevance of search results.

When choosing a chunking strategy, factors to consider include: the characteristics of the indexed content, the embedding model used and its optimal chunk size, the expected length and complexity of user queries, and how the search results are used in a specific application. For example, different chunking models should be used for content of different lengths. Different embedding models, such as Sentence-Transformer and text-embedding-ada-002, work differently when processing text chunks of different sizes; for example, Sentence-Transformer is more suitable for single sentence processing, while text-embedding-ada-002 is more suitable for processing text chunks containing 256 or 512 tokens. The length and complexity of the user's question text, as well as the specific needs of the application (such as semantic search or question answering), also affect the choice of chunking strategy. This may be directly related to the token limit of the selected large language model, so the chunk size may need to be adjusted. In fact, accurate query results are achieved by flexibly applying multiple chunking strategies. There is no best strategy, only the most suitable strategy.

Current RAG research has adopted a variety of block optimization methods to improve the efficiency and accuracy of retrieval. Among them, technologies such as sliding window technology aggregate global relevant information through multiple retrievals to achieve hierarchical retrieval.

Small2big 技术在搜索过程中使用小文本块，并为语言模型提供更大的相关文本块进行处理。摘要嵌入（Abstract embedding）技术对文档摘要执行 Top K 检索，以提供完整的文档上下文。元数据过滤（Metadata Filtering）技术通过文档的元数据进行过滤。图索引（Graph Indexing）技术把实体和关系转化为节点和连接，这在处理多跳问题时显著提升了相关性。这些方法的结合显著提升了 RAG 的检索效果和性能。

Fine-tuning the embedding model

After determining the appropriate size of the Chunk, we need to embed the Chunk and the query into the semantic space through an embedding model. Therefore, it becomes extremely important whether the embedding model can effectively represent the entire corpus. Today, some excellent embedding models have been launched, such as UAE[AnglE, 2023], Voyage[VoyageAI, 2023], BGE[BAAI, 2023], etc., which are pre-trained on large-scale corpora. However, when applied in specific domains, these models may not accurately reflect domain-specific corpus information. In addition, in order to ensure that the model can understand the relevance of user queries to content, it is crucial to perform task-specific fine-tuning of the embedding model, otherwise the un-fine-tuned model may not meet the needs of specific tasks. Therefore, fine-tuning the embedding model is essential for its downstream applications.

Domain knowledge fine-tuning

The two basic paradigms of embedding model fine-tuning include domain knowledge fine-tuning. In order to make the embedding model accurately understand the domain-specific information, we need to build a dedicated domain dataset to fine-tune the embedding model.

However, fine-tuning of embedding models is different from fine-tuning of regular language models. The main difference lies in the dataset used. The current mainstream method of fine-tuning embedding models uses datasets including queries, corpus, and relevant documents. The embedding model retrieves relevant documents in the corpus based on the query, and then uses the hit of the relevant documents of the query as the criterion for measuring the model.

In the process of building a dataset, fine-tuning a model, and evaluating, each part may encounter various challenges. LlamaIndex [Liu, 2023] introduces a series of key categories and functions specifically for the fine-tuning process of embedding models, which greatly simplifies this process. By preparing a corpus of domain knowledge and using the methods it provides, we can easily obtain specialized embedding models suitable for specific domain requirements.

Fine-tuning on downstream tasks

It is equally important to fine-tune the embedding model according to the downstream task. When using RAG for specific tasks, there have been studies that fine-tune the embedding model through the power of the large language model (LLM). For example, PROMPTAGATOR[Dai et al . , 2022] uses a large language model as a few-shot query generator and creates a task-specific retriever based on the generated data. This can solve the problem that some fields are difficult to perform conventional supervised fine-tuning due to insufficient data. LLM-Embedder[Zhang *et al.* , 2023a] uses a large language model to output reward values for data in multiple specific tasks, and double-fine-tunes the retriever using a hard labeled dataset and soft rewards from the LLM.

This approach improves semantic expression to a certain extent by introducing domain knowledge and fine-tuning for specific tasks. However, the retriever obtained by this

training method is not always directly beneficial to the large language model, so some studies have directly fine-tuned the embedding model by obtaining feedback signals from the LLM. (More details will be introduced in Section 4.4)

4.2 How to coordinate the semantic space of query and document

In RAG applications, some retrievers use the same embedding model to process queries and documents, while others use two different models. In addition, the user's original query may be unclear or lack necessary semantic information. Therefore, it is particularly important to coordinate the user's query with the semantic space of the document. This section will introduce two key techniques to help achieve this goal.

Query Rewrite

A straightforward approach is to rewrite the query.

As pointed out in Query2Doc[Wang *et al.* , 2023b] and ITER-RETGEN[Shao *et al.* , 2023], the power of large language models can be leveraged to generate a guiding pseudo-document, and then the original query is combined with this pseudo-document to form a new query.

In HyDE [Gao *et al.* , 2022], query vectors are constructed using text identifiers, which are used to generate a related but possibly non-existent "hypothetical" document, with the goal of capturing relevant patterns.

The RRR framework proposed by Ma's team in 2023 pioneered a new approach that reverses the order of retrieval and reading, focusing on how to rewrite queries. In this method, a large language model is first used to generate search queries, then relevant information is found through a web search engine, and finally a small language model is used to help the large model perform the so-called "training rewrite" to improve its effectiveness. The STEP-BACKPROMPTING method proposed by Zheng's team in 2023 enables large language models to conduct deeper abstract thinking, extract key concepts and principles, and perform information retrieval based on these.

In addition, multi-query retrieval methods allow large language models to generate multiple search queries simultaneously. These queries can be run simultaneously and their results processed together, which is particularly suitable for complex problems that require multiple small questions to be solved together.

Embedding Transformation

For embedding transformation, in addition to macro methods such as query rewriting, there are also some more micro techniques. In LlamaIndex proposed by Liu in 2023, researchers optimized the embedding representation of the query by adding a special adapter after the query encoder and fine-tuning it to make it more suitable for specific tasks.

When dealing with queries and documents with different structures, such as unstructured queries and structured documents, it is crucial to align the two. The SANTA method proposed by Li's team in 2023 is to enable the retrieval system to understand and process structured information. They proposed two pre-training methods: one is to use the natural correspondence between structured and unstructured data for comparative learning; the other is to adopt a masking strategy designed around entities, allowing the language model to predict and fill in the masked entity information.

4.3 Adjusting Retriever Results to Fit the Needs of Large Language Models

In the RAG (Retrieval-Augmented Generation) process, even if we use various techniques to improve the retrieval effect, the overall performance of RAG may not be significantly improved in the end. The reason is that the retrieved documents may not meet the requirements of the large language model (LLM). This section will introduce two methods to make the retriever output better meet the preferences of the LLM.

LLM supervised training Many studies have tuned embedding models by obtaining feedback signals from large language models. AAR[20] provides supervision signals to a pre-trained retriever through a language model (LM) based on an encoder-decoder architecture. The retriever is fine-tuned by analyzing the documents preferred by the LM (based on the cross-attention scores of FiD), using "hard negative sampling" and the traditional cross-entropy loss method. After such training, the retriever can be directly used to improve new target LLMs and achieve better results in related tasks. The training loss formula of the retriever is as follows:

$$g = \sum_{q} \sum_{d^{+} \in D^{a^{+}}} \sum_{d^{-} \in D^{-}} l\left(f\left(q, d^{+}\right), f\left(q, d^{-}\right)\right)$$

in,$D^{a^{+}}$ is the document set preferred by LLM,$D^{a^{-}}$ is a set of documents that are not preferred.$l$represents the traditional cross entropy loss function. The study finally pointed out that LLM may tend to focus on documents that are easy to read rather than informative.

REPLUG[14] uses a supervised training method by combining the document probability distribution calculated by the retriever and LLM. During the training process, the retrieval model is adjusted by calculating the KL divergence to improve its performance. This method is simple and effective, using LM as a supervisory signal without relying on a specific cross-attention mechanism. The training loss formula of the retriever is as follows:

$$g = \frac{1}{|D|} \sum_{x \in D} K.\ L\left(P_R\left(d|x\right) || Q_{LM}\left(d|x, and\ \right)\right)$$

Here, D represents the input context set, PR is the retrieval probability of the document, and QLM is the probability of each document based on LM.

UPRISE [Cheng *et al.* , 2023a] also uses a frozen large language model to fine-tune the Prompt Retriever.

In these studies, both language models and retrievers take prompt-input pairs as input. These models use the scores provided by the large language model to guide the training of the retriever, which is equivalent to annotating the dataset with the large language model.

Atlas [Izacard *et al.* , 2022] proposed four methods for fine-tuning supervised embedding models. One of them, Attention Distillation, learns through the cross-attention scores generated by the language model when generating output. EMDR2 uses the expectation-maximization algorithm to train the model using retrieved documents as hidden variables. Perplexity Distillation directly uses the perplexity of the tokens generated by the model as a training metric. LOOP introduces a new loss function based on the impact of document deletion on the prediction of large language models, which provides an effective training method for models to better adapt to specific tasks.

Insert the adapter

However, fine-tuning the embedding model may encounter some challenges, such as using APIs to implement embedding functions or insufficient local computing resources. Therefore, some studies choose external adapters to perform model alignment. PRCA [Yang *et al.* , 2023b] trains adapters in the context extraction stage and the reward-driven stage, and optimizes the output of the retriever through a token-based autoregressive strategy.

The TokenFiltering[Berchansky *et al.* , 2023] method effectively performs token filtering by calculating cross-attention scores and selecting the input token with the highest score. RECOMP[Xu *et al.* , 2023a] proposed the concept of extractive and generative compressors, which generate summaries by selecting relevant sentences or synthesizing document information to achieve multi-document query focused summarization. In addition, the novel method PKG[Luo *et al.* , 2023] injects knowledge into a white-box model through prescriptive fine-tuning and directly replaces the retriever module to directly output relevant documents based on the query.

5 Generate Components

In the RAG system, the generation component is one of the core parts, and its responsibility is to convert the retrieved information into natural and fluent text. This design is inspired by traditional language models, but unlike general generative models, the generation component of RAG improves the accuracy and relevance of the text by utilizing the retrieved information. In RAG, the input of the generation component includes not only traditional contextual information, but also relevant text fragments obtained through the retriever. This enables the generation component to have a deeper understanding of the context behind the question and produce more informative answers. In addition, the generation component will guide the generation of content based on the retrieved text to ensure that the generated content is consistent with the retrieved information. Precisely because of the diversity of input data, we have carried out a series of targeted work on the generation stage to better adapt to the input data from queries and documents.

5.1 How to improve search results through post-search processing?

For large language models that have not been fine-tuned, most studies rely on well-known large language models such as GPT-4[OpenAI, 2023] to fully retrieve document information with the help of their powerful internal knowledge base. However, these large models still have some inherent problems, such as context length limitations and sensitivity to redundant information. To address these problems, some studies have begun to focus on post-retrieval processing. Post-retrieval processing refers to further processing, filtering, or optimizing relevant information after it is retrieved from a large document database through a retriever. Its main purpose is to improve the quality of retrieval results, better meet user needs, or prepare for subsequent tasks. It can be understood as secondary processing of documents obtained in the retrieval stage. Post-retrieval processing usually includes information compression and re-ranking of results.

Information Compression

In terms of information compression, even if the retriever can extract relevant information from a huge knowledge base, we still face the challenge of processing a large amount of retrieved document information. Some studies have tried to solve this problem by expanding the context length of large language models, but current large models are still limited by context. In this case, information condensation becomes necessary. In general, the importance of information condensation is mainly reflected in

reducing information noise, solving context length limitations, and improving generation effects.

PRCA [Yang *et al.* , 2023b] solves this problem by training an information extractor. In the context extraction stage, this extractor can extract the context based on the given input text.S_{input}, generating an output sequence$C_{extracted}$, this sequence represents the condensed context in the input document. The training goal is to make$C_{extracted}$As close to the actual context as possibleC_{truth}The loss function they used is defined as follows:

$$min\ L\ (\ \theta\ ) = -\frac{1}{N}\sum_{i=1}^{N} C_{truth}^{(i)} log(f_{.}(S_{input}^{(i)}; i\ ))$$

here,$f_{.}$represents the function of the information extractor, and$i$is its parameter. Another project RECOMP[11] uses contrastive learning to train an information concentrator. In each training sample, there is one positive sample and five negative samples. In this process, the project uses the contrastive loss method[13] to train the encoder. The specific optimization objective is expressed as follows:

$$-log\frac{It\ is^{sim(x_i,p_i)}}{sim(x_i,p_i) + \sum_{n_j \in N_i} It\ is^{sim(x_i,p_i)}}$$

in$x_i$represents the training data,$p_i$is a positive sample,n_j 是负样本，sim(x,y) 用于计算 x 和 y 之间的相似度。还有一项研究则是致力于进一步减少文档的数量，以此提高模型回答问题的准确度。 [Ma *et al.*, 2023b] 提出了一种新的“Filter-Ranker”模式，它结合了大语言模型 (LLMs) 和小语言模型 (SLMs) 的优点。在这种模式下，SLMs 充当过滤器，LLMs 则作为排序器。通过激励 LLMs 对 SLMs 筛选出的难点样本进行重新排序，研究表明，这在各类信息提取 (IE) 任务中都取得了显著的提升。

Document reflow

In the document re-ranking process, the main function of the re-ranking model is to optimize the document set retrieved by the retriever.

当大语言模型 (LLM) 面临额外上下文的添加时，其性能往往会下降。为了应对这一挑战，重排序被提出作为一种行之有效的策略。其核心在于对文档记录进行重新组织，优先安排最相关的内容位于前列，同时将文档总量控制在一定数量之内。这种做法不仅有效缓解了检索时可能出现的上下文窗口扩大问题，也显著提升了检索的效率和响应速度[Zhuang *et al.*, 2023]。

The context compression function introduced in the re-ranking process aims to directly filter out relevant information based on the specific query context. The uniqueness of this strategy is that it can more centrally display the key information in the retrieval results by reducing the content of each document and filtering out irrelevant documents. Therefore, the re-ranking model plays a role in optimizing and refining the entire information retrieval process, providing more effective and accurate input for the subsequent large language model processing.

5.2 How to optimize the generator to deal with input data?

In the RAG model, optimizing the generator is crucial. The generator is responsible for converting the retrieved information into relevant text, which forms the final output of the model. The purpose of its optimization is to ensure that the generated text is both fluent and can effectively utilize the retrieved documents to better respond to user queries.

在一般的大语言模型 (LLM) 生成任务中，输入通常是个查询。而 RAG 的不同之处在于，输入不仅包括查询，还涵盖了检索器找到的多种文档（无论是结构化还是非结构化）。额外信息的加入对模型理解尤其是小型模型造成显著影响，因此，针对查询和检索文档的输入进行模型微调变得尤为重要。一般在将输入提供给微调过的模型之前，需要对检索器找到的文档进行后续处理。值得注意的是，RAG 中对生成器的微调方式与大语言模型的普通微调方法大体相同。本文将简要介绍包括格式化和非格式化数据及其优化函数的一些代表性研究。

General Optimization Process

The general optimization process involves input-output pairs in the training data, with the goal of having the model learn to produce output y based on input x.

In the study of Self-mem [Cheng *et al.* , 2023b], a traditional training method was used. Given an input x, relevant documents z are retrieved (the most relevant one is selected from the document), and then combined with (x, z), the model generates output y.

The paper explores two mainstream fine-tuning methods, namely the joint encoder (Joint-Encoder) [Arora *et al.* , 2023, Wang *et al.* , 2022b, Lewis *et al.* , 2020] and the dual encoder (Dual-Encoder) [Xia *et al.* , 2019, Cai *et al.* , 2021, Cheng *et al.* , 2022].

In the joint encoder mode, a standard encoder-decoder model is used. The encoder first processes the input, and then the decoder combines the encoding results through the attention mechanism to generate tokens autoregressively.

$$H = Encoder(x[SEP]m)$$

$$h^i = Decoder(CrossAttn(H), and < i)$$

$$P_{G_X}(.|x, and < i) = Softmax(h^i)$$

In the dual encoder system, two independent encoders are constructed, each responsible for encoding the input (query, context) and the document. These outputs are then processed by the decoder in turn, with bidirectional cross-attention processing. The authors used the Transformer [Vaswani et al., 2017] as the basis for both architectures and optimized the negative log-likelihood (NLL) loss of Gξ.

$$H_x = SourceEncoder(x) H_m = MemoryEncoder(x)$$

$$h^i = Decoder(CrossAttn(H_x, H_m), and < i)$$

$$\mathfrak{L}_{nll} = -\sum_{t=1}^{|y|} log P_{G_X}(\ and_t | x, m, and < t)$$

Using contrastive learning

During the training data preparation phase, interaction pairs between input and output are usually generated for contrastive learning.

In this scenario, the model is only exposed to one actual output, which may lead to the "exposure bias" problem [Ranzato et al., 2015]: that is, during the training phase, the model is only exposed to a single correct feedback and cannot learn about other possible generated tokens.

This may affect the performance of the model in real applications, as it may overfit to specific feedback in the training data instead of generalizing effectively to other situations. Therefore, SURGE [Kang et al., 2023] proposed a graph-based contrastive

learning method. For any pair of interactions between input and output, the objective of this contrastive learning method can be defined as follows:

$$\mathfrak{L}_{cont} = \frac{1}{2} log \frac{It\ is^{s\ im\ (\ \zeta\ (\ z\ )\ ,\ \xi\ (\ h\ ))\ /\ i}}{\sum_{h'} It\ is^{s\ im\ (\ \zeta\ (\ z\ )\ ,\ \xi\ (\ h'))\ /\ i}} + \frac{1}{2} log \frac{It\ is^{s\ im\ (\ \zeta\ (\ z\ )\ ,\ \xi\ (\ h\ ))\ /\ i}}{\sum_{With'} It\ is^{s\ im\ (\ \zeta\ (\ with'\ )\ ,\ \xi\ (\ h\ ))\ /\ i}}$$

in$g, X$is a learnable linear projection layer. z represents the average representation of the graph in the encoder, and h is the average representation in the decoder.$With', h'$ They represent the corresponding negative samples respectively.

In this text, the symbols 'h' and 'z' represent negative examples. By using contrastive learning, the model can more effectively learn to generate a variety of reasonable responses, not limited to the examples in the training data. This approach helps reduce the risk of overfitting, thereby improving the generalization ability of the model in real-world scenarios.

When dealing with retrieval tasks involving structured data, the SANTA [Li *et al.* , 2023d] study adopted a three-stage training process aimed at deeply understanding the structure and semantic information of the data.

Specifically, during the training phase of the retriever, contrastive learning is used to optimize the embedding representations of queries and documents, with the following optimization objectives:

$$\mathfrak{L}_{DR} = -log \frac{It\ is^{sim(q,d^+)}}{It\ is^{f(q,d^+)} + \sum_{d^- \in D^-} It\ is^{sim(q,d^-)}}$$

Here, q and d are the query and document processed by the encoder, respectively.d^- andd^+Represent negative samples and positive samples respectively. In the initial training stage of the generator, we use contrastive learning to align the relevant document descriptions of structured data and unstructured data. Its optimization goal is the same as above.

In the later training stage of the generator, we are inspired by references [16, 17] and realize the importance of entity semantics for learning text data representation in retrieval tasks. Therefore, we first perform entity recognition on structured data and then apply masks to these entities in the input part of the generator training data so that the generator can predict these masked parts. The optimization goal of this stage is:

$$\mathfrak{L}_{MEP} = \sum_{j=1}^{k} - l\ o\ g\ P\ (\ Y_d(t_j) | X_d^{mask}, AND_d(t_1, ..., j-1))$$

In sequenceAND_dmiddle,AND_d(and_jrepresents the jth Token. Here,AND_d = $< mask >_1$, ent_1, ..., $< mask >_n$, ent_nRepresents a sequence that contains some of the entity information that is masked. During training, we reveal these masked entities by analyzing the information in the context, understand the structural semantics of the text, and match them with the relevant entities in the structured data. Our goal is to enable the language model to effectively fill in this missing information and gain a deeper understanding of the meaning of the entity[21].

6 Enhancements to RAG Technology

This chapter mainly introduces the progress of RAG technology from three aspects: enhancement stage, data source and enhancement process.

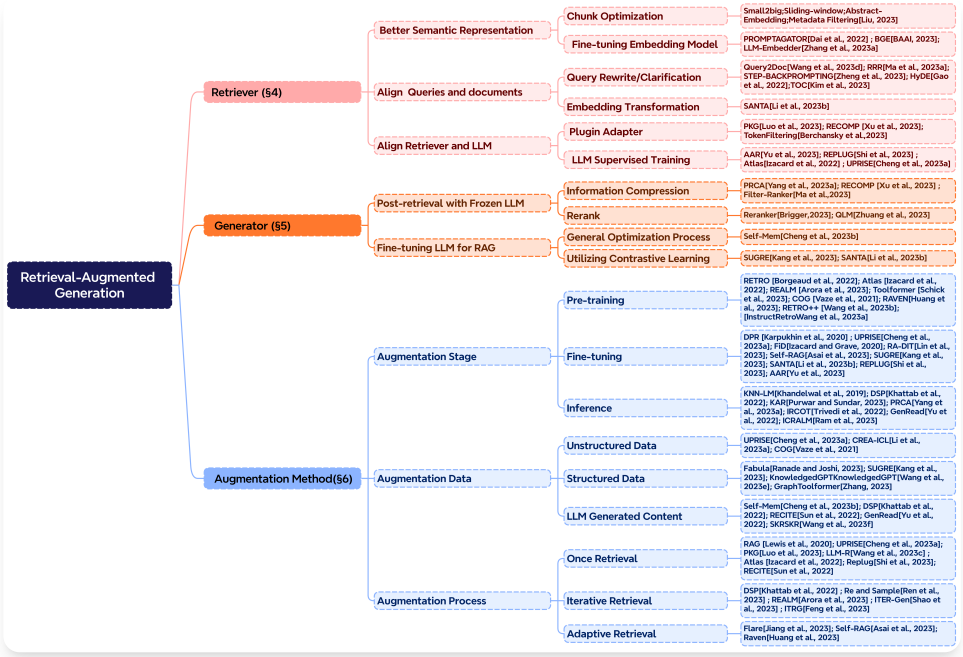

Figure 4: Classification of RAG core technologies.

6.1 Application of RAG in various enhancement stages

As a knowledge-intensive task, RAG uses a variety of technical means in the pre-training, fine-tuning, and inference stages of language model training.

Pre-training phase

In the pre-training stage, researchers strive to improve the performance of pre-trained language models in open-domain question answering through retrieval methods. The identification and expansion of implicit knowledge in pre-trained models is a challenge. In 2023, Arora *et al.* proposed REALM, a more modular and easy-to-understand knowledge embedding method. REALM adopts the masked language model (MLM) approach, treating pre-training and fine-tuning as a process of retrieval and then prediction, that is, the language model predicts the masked token y based on the masked sentence x, modeling P(x|y).

2022 年，Borgeaud *et al.* 提出的 RETRO 则是利用检索增强来预训练自回归语言模型，它通过从大量标记数据集中检索信息，实现了从零开始的大规模预训练，并显著减少了模型的参数量。

RETRO not only shares the main structure with the GPT model, but also adds a RETRO encoder to encode the features of related entities retrieved from an external knowledge base.

Furthermore, RETRO adds a block-wise cross-attention layer to the Transformer structure of its decoder, which effectively incorporates the retrieval information from the RETRO encoder. This enables RETRO to show lower perplexity than the standard GPT model when dealing with complex problems. In addition, RETRO is more flexible in updating the knowledge stored in the language model, which can be achieved by updating the retrieval database without retraining the entire model [Petroni *et al.* , 2019].

Atla[Izacard *et al.* , 2022] adopts a similar approach to the T5 architecture[Raffel *et al.* , 2020], incorporating retrieval mechanisms in both pre-training and fine-tuning stages. Before starting pre-training, Atla initializes its large language model base for the encoder-decoder with the pre-trained T5, and initializes the dense retriever with the pre-trained Contriever.

During the pre-training process, this method improves efficiency by reducing the use of parameters compared to traditional pre-trained models. It is particularly good at handling tasks that require a lot of knowledge and can build specialized models by training on corpora in specific fields. However, this method also has its shortcomings, such as the need for a large amount of pre-training data, more training resources, and slow updates. In particular, the cost of retrieval-based training will increase relatively as the model size increases. Despite these limitations, this method excels in enhancing the robustness of the model. Once training is completed, the retrieval-enhanced model based on pure pre-training no longer requires dependence on external libraries, thereby improving generation speed and operational efficiency.

Fine-tuning phase

In the downstream fine-tuning stage, researchers have adopted a variety of methods to improve the information retrieval capabilities of retrievers and generators in open-domain question-answering tasks. For example, REPlUG [Shi *et al.* , 2023] treats the language model (LM) as a black box and optimizes it through an adjustable retrieval model. REPLUG obtains feedback from the black-box language model through supervision signals to improve the initial retrieval model. And UPRISE [Cheng *et al.* , 2023a] creates a lightweight and flexible retriever by fine-tuning on a diverse set of tasks.

This retriever can automatically generate retrieval hints for zero-shot tasks, demonstrating its versatility and superior performance across different tasks and models.

At the same time, researchers have also made efforts to fine-tune the generator. For example, Self-Mem[Cheng *et al.* , 2023b] fine-tunes the generator by leveraging the example pool, while Self-RAG[Asai *et al.* , 2023b] generates reflection tokens to meet the needs of active retrieval.

The RA-DIT[Lin *et al.* , 2023] method fine-tunes both the generator and the retriever by increasing the probability of the correct answer under retrieval enhancement instructions. It effectively utilizes relevant background knowledge by minimizing the semantic similarity between the document and the query.

In addition, SUGRE [Kang *et al.* , 2023] introduced the concept of contrastive learning and achieved end-to-end fine-tuning of the retriever and generator, thereby ensuring the high accuracy of text generation and the detailedness of the retrieved subgraph.

SURGE uses a context-aware subgraph retriever based on Graph Neural Networks to extract knowledge relevant to the ongoing conversation from the knowledge graph, ensuring that the generated responses faithfully reflect the retrieved knowledge. To achieve this goal, SURGE uses an efficient and invariant graph encoder and a graph-text comparison learning objective.

总的来说，微调阶段的增强方法有几个显著特征。

First, fine-tuning the large language model (LLM) and the retriever can better adapt to specific tasks, which provides the flexibility to fine-tune either one simultaneously or separately. For example, the RePlug[Shi *et al.* , 2023] and RA-DIT[Lin *et al.* , 2023] methods demonstrate this. Second, fine-tuning helps the model adapt to diverse downstream tasks, as shown in UPRISE[Cheng *et al.* , 2023a], making the model more versatile. In addition, fine-tuning also enables the model to better handle a variety of corpora with different data structures, especially in processing graph-structured corpora, which has obvious advantages, as exemplified by the SUGRE method.

However, the fine-tuning phase also has limitations, such as the need for specially prepared datasets for RAG fine-tuning and the need for more computing resources compared to the inference phase. Overall, during the fine-tuning phase, researchers can customize the model according to specific needs and data formats, while reducing resource consumption while still being able to adjust the output style of the model.

Reasoning stage

In the reasoning stage, the combination of RAG method and large language model has become a research hotspot. For example, Naive RAG is a research model that integrates retrieval content into the reasoning stage.

To make up for the shortcomings of Naive RAG, researchers introduced more context into RAG in the reasoning stage. The DSP[Khattab *et al.* , 2022] framework passes natural language text between a frozen language model (LM) and a retrieval model (RM) through a complex process, providing the model with richer context and thus improving the generation quality. PKG equips the large language model with a knowledge-guided module, allowing the model to access relevant knowledge without changing parameters, enabling it to perform more complex tasks. Meanwhile, CREA-ICL[Li *et al.* , 2023b] uses synchronous retrieval of cross-lingual knowledge to obtain additional information, while RECITE builds context by extracting one or more paragraphs from a large language model.

During the inference phase, optimization of the RAG process helps the model adapt to more complex tasks.

For example, ITRG [Feng *et al.* , 2023a] improves the model's adaptability to multi-step reasoning tasks by iteratively retrieving and finding the correct reasoning path.

ITER-RETGEN[Shao *et al.* , 2023] adopts an innovative iterative approach to closely combine information retrieval and content generation. This method alternates the process of "using retrieval to help generation" and the process of "using generation to feed back retrieval", thereby effectively improving the accuracy and relevance of information. IRCOT[Trivedi *et al.* , 2022] is a method that combines the concepts of RAG and CoT[Wei *et al.* , 2022]. It effectively improves the performance of GPT-3 in various question-answering tasks by alternating between using CoT-guided retrieval and using retrieval results to strengthen CoT, which highlights the great potential of integrating retrieval and generation techniques.

In summary, the enhancement technology in the reasoning stage is highly praised for its light weight, high efficiency, no need for additional training, and the ability to effectively utilize existing powerful pre-trained models. Its biggest feature is to keep the parameters of the large language model (LLM) unchanged when fine-tuning the model. The focus is on providing more appropriate contextual information according to different needs, while having the advantages of fast and low cost. However, this method also has some limitations, such as the need for additional data processing and process optimization, and being limited by the performance of the basic model. In order to better adapt to different task requirements, this method is often combined with optimization techniques such as step-by-step reasoning, iterative reasoning, and adaptive retrieval.

6.2 Data Augmentation Sources

The source of data is crucial to the effectiveness of RAG (Retrieval-Augmented Generation). Different data sources provide knowledge of different granularity and dimensions, so different processing methods are required. They are mainly divided into

three categories: unstructured data, structured data, and content generated by large language models.

Unstructured Data Enhancement

In terms of unstructured data, this type of data is mainly textual and usually comes from a corpus of plain text. In addition, there are other text data that can be used for retrieval, such as Prompt data for fine-tuning large models [Cheng *et al.* , 2023a] and cross-lingual data [Li *et al.* , 2023b].

In terms of the granularity of text processing, in addition to common sentence blocks, the retrieval units can also be tokens (such as kNN-LM [Khandelwal *et al.* , 2019]), phrases (such as NPM [Lee *et al.* , 2020], COG [Vaze *et al.* , 2021]) and document paragraphs. More detailed retrieval units can better cope with rare patterns and out-of-domain scenarios, but the retrieval cost will also increase accordingly.

At the vocabulary level, FLARE implements an active retrieval strategy that only initiates retrieval when the large language model generates a low-probability word. This approach involves first generating a temporary next sentence to retrieve relevant documents, and then generating the next sentence again based on the retrieved documents to predict the next sentence.

At the level of text blocks, RETRO uses the previous block to retrieve the closest block, and integrates this information into the context of the previous block to guide the generation of the next block. Specifically, RETRO extracts the nearest neighbor block N(Ci−1) of the previous block from the retrieval database, and combines the context information of the previous block (C1*,...,C*i−1) with the retrieval information of N(Ci−1), and guides the generation of the next block Ci through a cross-attention mechanism. In order to maintain the coherence of causal logic, when generating the i-th block Ci, only the nearest neighbor N(Ci−1) of the previous block can be referenced, and N(Ci) cannot be used.

Structured Data Augmentation

In terms of structured data enhancement, data sources such as Knowledge Graph (KG) are gradually being integrated into the RAG framework. Verified knowledge graphs can provide higher-quality contextual information, thereby reducing the possibility of model illusions.

For example, RET-LLM [Modarressi *et al.* , 2023] builds a personalized knowledge graph memory by extracting relation triplets from past conversations for future conversation processing.

SUGRE[Kang *et al.* , 2023] uses a graph neural network (GNN) to embed relevant subgraphs retrieved from a knowledge graph, which prevents the model from generating responses that are not relevant to the topic.

SUGRE [Kang *et al.* , 2023] adopts a graph encoding method that incorporates graph structure into the representation space of pre-trained models (PTMs) and uses a multimodal contrastive learning objective between graph and text modalities to ensure the consistency of retrieved facts and generated text.

KnowledgeGPT[Wang *et al.* , 2023c] generates code-formatted search queries that are applicable to knowledge bases (KBs) and include predefined KB operation functions. In addition to retrieval capabilities, KnowledgeGPT is also able to store knowledge in personalized knowledge bases to meet the personalized needs of users. These

structured data sources provide RAG with richer knowledge and context, thereby improving model performance.

LLM Generated Content RAG

Given that the auxiliary information recalled by RAG is sometimes ineffective or even counterproductive, some studies have expanded the application paradigm of RAG and explored the internal knowledge of the large language model (LLM). This method uses the content generated by the LLM itself for retrieval, with the aim of improving the performance of downstream tasks. The following are some important studies in this field: SKR [Wang *et al.* , 2023d] used a labeled training set to classify questions that the model can answer directly as "known" and questions that require additional retrieval enhancement as "unknown". The model is trained to distinguish whether a question is "known" or not, and only applies retrieval enhancement to "unknown" questions, while directly giving answers to other questions.

GenRead[Yu *et al.* , 2022] replaces the retriever with an LLM generator. Experimental results show that context documents generated by LLM contain correct answers more often than those retrieved by traditional RAG, and the generated answers are of higher quality. The authors believe that this is because the task of generating document-level context matches the pre-training objective of causal language modeling, allowing the model to more effectively utilize the world knowledge stored in the parameters.

Selfmem[Cheng *et al.* , 2023b] builds an infinite memory pool by iteratively using a retrieval-augmented generator. The system includes a memory selector that selects a generated output as the memory for subsequent generation. This output corresponds to the other side of the original question. By combining the original question and its opposite, the retrieval-augmented generative model can use its own output to improve itself.

These different approaches demonstrate innovative strategies in the field of RAG retrieval enhancement, aiming to improve the performance and effectiveness of the models.

6.3 Enhancement Process

In most RAG (retrieval and generation) studies, only a single retrieval and generation operation is usually performed. However, a single retrieval may carry duplicate information, causing the generated content to be "out of focus" [Liu *et al.* , 2023]. Such duplicate information may obscure key information or contain content that contradicts the correct answer, thereby negatively affecting the quality of generation [Yoran *et al.* , 2023]. In addition, the information obtained by a single retrieval is limited in problems that require multi-step reasoning.

The main methods for optimizing the retrieval process currently include iterative retrieval and adaptive retrieval. These methods enable the model to perform multiple iterations in the retrieval process or adaptively adjust the retrieval method according to different tasks and scenarios.

Iterative search

By periodically collecting documents based on the original query and generated text, more references can be provided to the large language model (LLM) [Borgeaud *et al.* , 2022, Arora *et al.* , 2023]. The added references in multiple iterations of retrieval have improved the robustness of subsequent answer generation. However, this approach may

have semantic gaps and sometimes collect noisy information because it mainly relies on a series of tokens to distinguish generated and retrieved documents.

Recursive and multi-hop retrieval are applied to specific data scenarios. Recursive retrieval first processes the data through a structured index and then searches layer by layer. When searching for documents with rich hierarchies, you can create summaries for each part, whether it is a whole document or a long PDF. After searching based on the summary, once the document is identified, a secondary search is performed on its internal parts to achieve recursive retrieval. Multi-hop retrieval is often used to deeply mine information in graph-structured data sources [Li *et al.* , 2023c].

Some approaches combine iterations of the retrieval and generation steps.

ITER-RETGEN [Shao *et al.* , 2023] combines "retrieval-augmented generation" and "generation-augmented retrieval" for tasks that require recurrent information. That is, the model responds to the input task with the content needed to complete the task, which then becomes the information background for retrieving more relevant knowledge. This helps generate better responses in the next iteration.

IRCoT[Trivedi *et al.* , 2022] explores a method to retrieve documents at each step of the thought chain, performing a retrieval for each sentence generated. It uses CoT (Continuous Task) to guide retrieval and uses the retrieval results to optimize CoT, thereby ensuring semantic integrity.

Adaptive Search

In the field of adaptive retrieval, methods such as Flare[Jiang *et al.* , 2023b] and Self-RAG[Asai *et al.* , 2023b] have improved on conventional RAG methods. Traditional RAG methods take a passive approach when retrieving information, while these new methods allow large language models (LLMs) to actively decide when and what to retrieve, thereby improving the efficiency and accuracy of information retrieval.

In fact, the practice of actively using tools and making judgments with large language models (LLMs) did not originate from RAG, but has been widely used in many large-model AI agents [Yang *et al.* , 2023c, Schick *et al.* , 2023, Zhang, 2023].

Taking Graph-Toolformer[Zhang, 2023] as an example, its retrieval step is divided into several stages: LLM actively uses the retriever to stimulate search queries through few-shot prompts. When LLM deems it necessary, it actively searches for related questions to collect necessary information, similar to the process of AI agents calling tools.

WebGPT[Nakano *et al.* , 2021] uses reinforcement learning to train the GPT-3 model, enabling it to query, browse, and cite on search engines through special tokens, thereby effectively utilizing search engines in text generation.

Flare[Jiang *et al.* , 2023b] effectively reduces the cost of document retrieval by automatically determining the best time for information retrieval. This method monitors the probability changes during the text generation process. Once the probability of generating a term drops below a certain threshold, it triggers the information retrieval system to supplement the required knowledge.

Self-RAG[Asai *et al.* , 2023b] introduces a novel "reflection token", which is divided into two types: "retrieval" and "criticism". This enables the model to autonomously decide when to retrieve information based on the set criteria, thereby effectively obtaining the required paragraphs.

When information retrieval is needed, the generator processes multiple paragraphs at the same time and uses a technique called "fragment-level beam search" to determine the optimal combination of content. During this process, the importance of each part is evaluated and updated through a method called "critic scores", and these scores can be adjusted as needed during the answer generation process to customize the model's response. An innovation of the Self-RAG framework is that it allows the large language model (LLM) to decide whether it needs to review past information, thus avoiding the need to train additional classifiers or rely on natural language inference (NLI) models. This greatly improves the model's ability to independently judge information and generate accurate answers.

7 RAG Assessment

In the process of exploring and optimizing RAG (Retrieval Augmentation Generator), how to effectively evaluate its performance has become a key issue. This chapter mainly discusses the evaluation methods, the key indicators that RAG should have, its core capabilities, and some commonly used evaluation frameworks.

7.1 Evaluation Methodology

There are two main methods to evaluate the effectiveness of RAG: independent evaluation and end-to-end evaluation [Liu, 2023].

Independent evaluation

The independent evaluation involves the assessment of the retrieval module and the generation module (i.e., reading and synthesizing information).

1. **Retrieval module** : The performance of the RAG retrieval module is usually evaluated using a series of metrics that measure the effectiveness of a system (such as a search engine, recommendation system, or information retrieval system) in ranking items based on a query or task. These metrics include hit rate, mean reciprocal ranking (MRR), normalized discounted cumulative gain (NDCG), precision, etc.
2. **Generation module** : Generation module refers to combining the retrieved documents with the query to form an enhanced or synthesized input. This is different from the generation of the final answer or response, which is usually evaluated in an end-to-end manner. The evaluation of the generation module focuses on contextual relevance, that is, the relevance of the retrieved documents to the query question.

End-to-end evaluation

End-to-end evaluation is the evaluation of the final response generated by the RAG model for a specific input, involving the relevance and consistency of the answer generated by the model with the input query.

From the perspective of the goal of content generation, evaluation can be divided into unlabeled and labeled content evaluation. The evaluation indicators of unlabeled content include accuracy, relevance, and harmlessness of the answer, while the evaluation indicators of labeled content include accuracy and exact match (EM). In addition, depending on the evaluation method, end-to-end evaluation can be divided into manual evaluation and automatic evaluation using a large language model (LLM). In general, these are conventional methods for end-to-end evaluation of RAG. RAG applications in specific fields also adopt specific evaluation indicators, such as exact match (EM) for question-answering tasks [Borgeaud *et al.* , 2022, Izacard *et al.* , 2022],

UniEval and E-F1 for summary tasks [Jiang *et al.* , 2023b], and BLEU for machine translation [Zhong *et al.* , 2022].

These indicators help to understand the performance of RAG in various specific application scenarios.

7.2 Key indicators and capabilities

Existing studies often lack rigorous evaluation of the generation effect of retrieval-enhanced large language models (LLMs). Typically, evaluating the application of RAG in different downstream tasks and different retrievers may yield different results. However, some academic and engineering practices have begun to focus on the general evaluation metrics of RAG and the capabilities required for effective use. This section mainly introduces the key metrics for evaluating the effectiveness of RAG and the basic capabilities required to evaluate its performance.

Key indicators

A recent OpenAI report [Jarvis and Allard, 2023] discusses various techniques for optimizing large language models (LLMs), including RAG and its evaluation criteria.

In addition, the latest evaluation frameworks such as RAGAS [Es *et al.* , 2023] and ARES [Saad-Falcon *et al.* , 2023] also apply the RAG evaluation criteria. After combing through these studies, we mainly focus on three key indicators: answer accuracy, answer relevance, and context relevance.

1. Answer accuracy: This metric focuses on ensuring that the answers generated by the model are consistent with the truth of the given context, ensuring that the answers do not conflict or deviate from the contextual information. This evaluation metric is crucial to avoid misleading in large models.
2. Answer relevance: This metric emphasizes that the generated answer needs to be closely related to the question itself.
3. Contextual relevance: This metric requires that the extracted contextual information must be as precise and targeted as possible to avoid irrelevant content. After all, processing long texts is costly for large language models, and too much irrelevant information will reduce the efficiency of the model's use of context. OpenAI's report also specifically mentions "context extraction" as a supplementary metric to measure the model's ability to retrieve relevant information needed to answer questions. This metric reflects the degree of search optimization of the RAG retrieval module. Low recall may suggest the need to optimize the search function, such as introducing a re-ranking mechanism or adjusting the embedding to ensure that more relevant content is retrieved.

Key Capabilities

The study by RGB [Chen *et al.* , 2023b] analyzes the performance of different large language models in four basic capabilities required for processing RAGs, including noise resistance, rejection of invalid answers, information synthesis, and counterfactual robustness, thus setting standards for retrieval-enhanced generation. RGB focuses on the following four capabilities:

1. **Noise immunity:** This capability evaluates how effectively the model can handle noisy documents that contain invalid information but are relevant to the question.

2. **Invalid answer rejection:** The model should correctly reject answers when the document it retrieved lacks the information needed to solve the problem. When testing invalid answer rejection, the external documents contain only invalid information. Ideally, a large language model should emit a "not enough information" or similar rejection signal.
3. **Information synthesis ability:** This ability evaluates whether the model can integrate information from multiple documents to answer more complex questions.
4. **Counterfactual robustness testing:** This test aims to evaluate whether the model can identify and correct incorrect information in documents when it is told that the retrieved information may be risky. Counterfactual robustness testing includes questions that the large language model can answer directly, but the relevant external documents contain incorrect facts.

7.3 Evaluation Framework

Recently, the large language model community has begun to explore automatic evaluation methods using large language models as evaluators, and many studies have used advanced models such as GPT-4 to evaluate the effects of their large language model applications. Databricks has used GPT-3.5 and GPT-4 as evaluators to review their chatbot applications, and the results show that this automatic evaluation method is quite effective [Leng *et al.* , 2023]. They also believe that this method is both efficient and cost-effective for evaluating retrieval-generation (RAG)-based applications. In the field of RAG evaluation frameworks, RAGAS and ARES are newer methods. These evaluations focus on three core indicators: accuracy, relevance, and context relevance of answers. In addition, the open source library TruLens proposed by the industry also adopts a similar evaluation method. All of these frameworks use large language models as evaluators. Since TruLens is similar to RAGAS, this section will focus on RAGAS and ARES.

RAGAS

This framework focuses on the ability of retrieval systems to pick out key contextual passages, the ability of large language models to accurately utilize these passages, and the overall quality of generated content. RAGAS is an evaluation framework based on simple handwritten prompts that fully automatically measures the accuracy, relevance, and contextual relevance of answers. In the implementation and experiments of this framework, all prompts are evaluated using the gpt-3.5-turbo-16k model from the OpenAI API [Es *et al.* , 2023].

Algorithm Principle

1. **Answer fidelity evaluation** : Use the Large Language Model (LLM) to decompose the answer into multiple statements and check the consistency of each statement with the context. Finally, a "fidelity score" is calculated based on the ratio of the number of supported statements to the total number of statements.
2. **Answer relevance evaluation** : Use a large language model (LLM) to create possible questions and analyze their similarity to the original question. The answer relevance score is calculated by averaging the similarity of all generated questions to the original question.
3. **Context relevance evaluation** : Use the Large Language Model (LLM) to filter out sentences directly related to the question, and determine the context relevance score based on the proportion of these sentences to the total number of context sentences.

ARES

ARES 的目标是自动化评价 RAG 系统在上下文相关性、答案忠实度和答案相关性三个方面的性能。这些评价指标与 RAGAS 中的相似。但是，RAGAS 作为一个基于简单手写提示的较新评估框架，在适应新 RAG 评估场景方面有一定局限性，这正是 ARES 项目的显著意义。此外，ARES 在评估中的表现明显不如 RAGAS。ARES 减少了评估成本，通过使用少量的手动标注数据和合成数据，并应用预测驱动推理 (PDR) 提供统计置信区间，提高了评估的准确性[Saad-Falcon 等人，2023]。

Algorithm Principle

1. **Generating Synthetic Datasets** : ARES first uses a language model to generate synthetic questions and answers from documents in the target corpus, creating both positive and negative samples.
2. **Training Large Language Models (LLMs) Judges** : ARES then fine-tunes lightweight language models, training them on synthetic datasets to evaluate context relevance, answer fidelity, and answer relevance.
3. **Ranking RAG systems based on confidence intervals** : Finally, ARES uses these referee models to score RAG systems and, combined with a manually annotated validation set, uses the PPI method to generate confidence intervals to reliably evaluate the performance of RAG systems.

8 Future Outlook

In this chapter, we discussed the three major future development directions of RAG: vertical optimization, horizontal expansion, and the construction of the RAG ecosystem.

8.1 Vertical Optimization of Rag

尽管 RAG 技术在过去一年里取得了显著进展，但其垂直领域仍有几个重点问题有待深入探究。

The first is the problem of handling long context in RAG. As pointed out in the literature [Xu *et al.* , 2023c], RAG is limited by the context window size of the large language model (LLM) when generating content. A window that is too small may lead to insufficient relevant information, while a window that is too large may cause information loss. At present, continuously expanding the scale of the LLM context window and even achieving infinite context has become a key direction for the research and development of large language models. However, once the limitation of the context window is removed, how RAG adapts to this change has become a problem worthy of attention.

其次，RAG 的鲁棒性研究也是一个重要焦点。在检索过程中，如果出现无关噪声或与事实矛盾的内容，会显著影响 RAG 的效果。这种情况就像“打开一本书偶遇毒蘑菇”，因此，如何增强 RAG 的鲁棒性，已成为研究者们关注的热点，相关研究有 [Yu *et al.*, 2023a, Glass *et al.*, 2021, Baek *et al.*, 2023] 等。

Third, the synergy between RAG and fine-tuning is also one of the research focuses. For example, studies such as RA-DIT [Lin *et al.* , 2023] show that hybrid methods have become the mainstream trend of RAG. How to effectively coordinate the relationship between parametric and non-parametric methods while maintaining the advantages of the two is an unresolved problem.

Finally, the engineering application of RAG has also attracted much attention. The rise of RAG is partly due to its ease of implementation and its compliance with enterprise

engineering needs.

However, in engineering practice, key issues that need to be addressed urgently include how to improve retrieval efficiency and document recall in large-scale knowledge base scenarios, and how to ensure enterprise data security—for example, preventing LLM from being induced to leak the source, metadata, or other sensitive information of documents [Alon *et al.* , 2022].

Horizontal Scaling of RAG

In the horizontal field, RAG research is also expanding rapidly. Starting from the initial field of text question answering, the application of RAG has gradually expanded to more modal data, including images, codes, structured knowledge, audio and video, etc. In these fields, many related research results have emerged.

In the field of images, the BLIP-2 [Li *et al.* , 2023a] proposal uses a frozen image encoder and a large language model to pre-train visual language, which greatly reduces the cost of model training. More importantly, the model can achieve zero-shot conversion from image to text.

在文本生成领域，VBR[Zhu *et al.*, 2022] 方法通过生成图像来引导语言模型进行文本创作，这在开放式文本生成任务中显示出显著的效果。

In the field of coding, RBPS [Nashid *et al.* , 2023] is applied to small-scale learning processes related to coding. This method can automatically find code examples similar to the developer's current task through coding or frequency analysis, which has been proven to be very effective in generating test assertions and program repair.

In the structured knowledge domain, methods such as CoK [Li *et al.* , 2023c] first extract facts related to the question from the knowledge graph and then incorporate these facts into the input in the form of prompts. This approach performs well in knowledge graph question answering tasks.

For audio and video, the GSS[Zhao *et al.* , 2022] method can quickly convert machine translation (MT) data into speech translation (ST) data by retrieving and concatenating audio clips from a spoken word corpus. UEOP[Chan *et al.* , 2023] brings new breakthroughs in end-to-end automatic speech recognition, introducing an external offline strategy for converting sound to text.

By leveraging the audio embeddings and semantic text embeddings generated by the text-to-speech method, automatic speech recognition (ASR) can be optimized through a KNN-based attention fusion strategy, effectively shortening the time for domain adaptation.

The Vid2Seq[Yang *et al.* , 2023a] architecture enhances the language model by introducing special time markers, enabling it to seamlessly predict event boundaries and text descriptions in the same output sequence.

8.2 RAG Ecosystem

Downstream tasks and evaluation

By integrating relevant information from a wide range of knowledge bases, RAG demonstrates great potential in processing complex queries and generating information-rich responses. Numerous studies have shown that RAG performs well in a variety of downstream tasks such as open-ended question answering and fact

verification. The RAG model not only improves the accuracy and relevance of information in downstream applications, but also increases the diversity and depth of responses.

The success of RAG has paved the way for exploring its applicability and universality in multiple fields, and future work will focus on this. In particular, in knowledge question answering in professional fields such as medicine, law, and education, the application of RAG may provide lower training costs and better performance than fine-tuning.

At the same time, improving the evaluation system of RAG to better evaluate and optimize its application in different downstream tasks is crucial to improving the efficiency and effectiveness of the model in specific tasks. This involves developing more accurate evaluation metrics and frameworks for various downstream tasks, such as contextual relevance, content innovation, and harmlessness.

In addition, enhancing the interpretability of RAG models so that users can more clearly understand how and why the model makes a specific response is also an important task.

Technology Stack

In the technology ecosystem of RAG, the development of related technology stacks plays a driving role. For example, with the popularity of ChatGPT, LangChain and LLamaIndex quickly became well-known technologies, which provide rich RAG-related APIs and become one of the key technologies in the era of large models. At the same time, new technology stacks are constantly emerging. Although these new technologies are not as versatile as LangChain and LLamaIndex, they pay more attention to their own unique characteristics. For example, Flowise AI6 focuses on low-code operations, enabling users to implement various AI applications represented by RAG through simple drag and drop. Other emerging technologies such as HayStack, Meltno, and Cohere Coral are also developing.

In addition to AI native frameworks, traditional software or cloud service providers are also expanding their service scope. For example, Verba7 launched by vector database company Weaviate focuses on the field of personal assistants, while Amazon provides services to users based on RAG ideas through its intelligent enterprise search service Kendra, allowing them to search in different content repositories.

The development of the technology stack and the progress of RAG promote each other. New technologies have put forward higher requirements for the existing technology stack, and the optimization of technology stack functions has further promoted the development of RAG technology. Overall, the technology stack of the RAG tool chain has been initially established, and many enterprise-level applications have gradually emerged. However, a comprehensive integrated platform is still being perfected.

9 Conclusion

This paper explores the Retrieval-Augmented Generation (RAG) technique in depth. This technique uses external knowledge bases to enrich the context of large language models (LLMs) and generate answers. The characteristic of RAG is that it combines the parameterized knowledge in the large language model with the external non-parametric knowledge, effectively reduces the errors and false content of the generated information, and uses retrieval technology to obtain timely information, thereby improving the accuracy of the answer. In addition, RAG improves the transparency of the model output and the user's trust in the results by citing the source. RAG can also be customized according to the needs of specific fields by integrating relevant text data.

The development of RAG can be summarized into three modes: basic RAG, advanced RAG and modular RAG. Each of these three modes has different models, methods and limitations. Basic RAG mainly performs simple "search-read" operations. Advanced RAG is more sophisticated in data processing, optimizes the knowledge base index, and adopts multiple or iterative search methods.

As the technology is explored in depth, RAG begins to incorporate other technologies such as fine-tuning, forming a modular RAG. This new model makes the RAG process richer and more flexible by introducing new modules.

In the following part of the paper, we will analyze the three core components of RAG in depth. The fourth part introduces the retrieval mechanism of RAG, including how to process text data to obtain more accurate semantic representation, narrow the semantic gap between query and document, and how to adjust the retriever to better cooperate with the generator. The fifth part explains how the generator can improve the generation quality by post-processing the retrieved documents to avoid missing information in the processing process, and explores how to adjust the generator to better adapt to the retriever. The sixth part reviews the various methods currently used to improve retrieval results, and discusses them from aspects such as the retrieval stage, data source, and retrieval process.

Section 7 describes how to evaluate existing RAG methods, including evaluation criteria, key indicators, and the current evaluation framework. Finally, we look forward to the future research direction of RAG. As a technology that combines retrieval and generation, RAG has broad room for development in future research. With the continuous improvement of technology and the expansion of its application scope, the performance and practicality of RAG will be further improved.

‹ See all posts