Modular RAG and RAG Flow: Part I

Yunfan Gao · Follow

18 min read · Jan 24, 2024

> *A compressive and high-level summarization of RAG .*
>
> *In Part I, we will focus the concept and components of Modular RAG, containing 6 module types, 14 modules and 40+ operators.*

Intro

Over the past year, the concept of **Retrieval-Augmented Generation (RAG)** as a method for implementing LLM applications has garnered considerable attention. We have authored a comprehensive survey on RAG , delving into the shift from Naive RAG to Advanced RAG and Modular RAG. However, the survey primarily scrutinized RAG technology through the lens of Augmentation (e.g. Augmentation Source/Stage/Process).

This piece will specifically center on the Modular RAG paradigm. We further defined a **three-tier Modular RAG** paradigm, comprising **Module Type**, **Module**, and **Operator.** Under this paradigm, we expound upon the core technologies within the current RAG system, encompassing 6 major Module Types, 14 Modules, and 40+Operators, aiming to provide a comprehensive understanding of RAG.

By orchestrating different operators, we can derive various **RAG Flows**, a concept we aim to elucidate in this article. Drawing from extensive research, we have distilled and summarized typical patterns, several specific implementation cases and best industry cases. (Due to space constraints, this part will be addressed in Part II.)

The **objective** of this article is to offer a more sophisticated comprehensi of the present state of RAG development and to pave the way for future advancements. Modular RAG presents plenty opportunities, facilitating the definition of new operators, modules, and the configuration of new Flows.

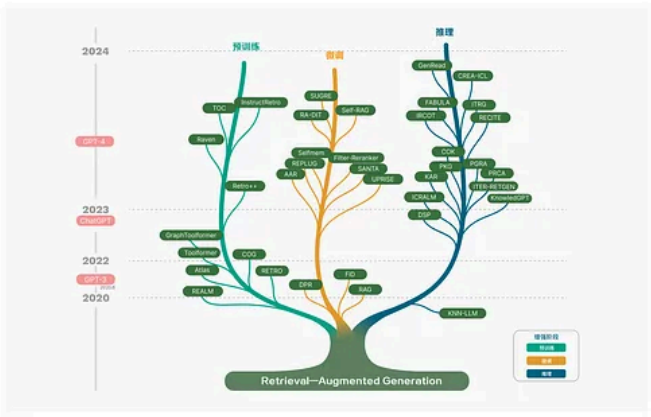


Figure 1: Technology tree of RAG research development featuring representative works

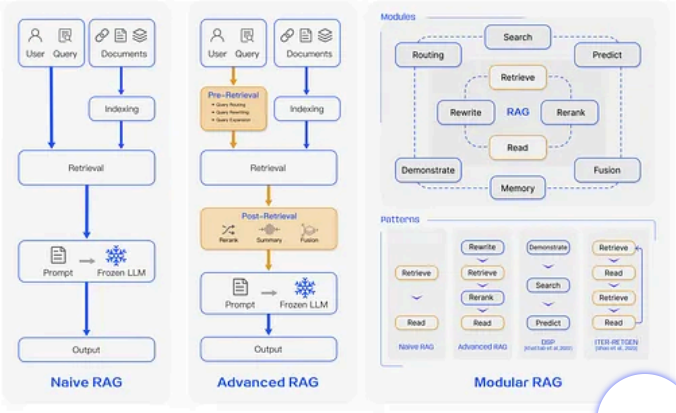


Figure 2: Comparison between the three paradigms of RAG

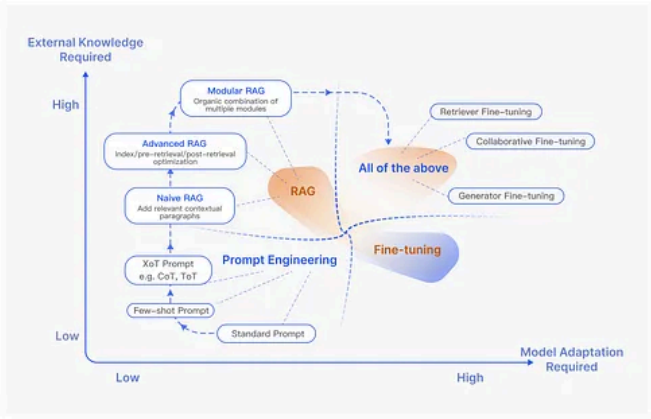


Figure 3: RAG compared with other model optimization methods

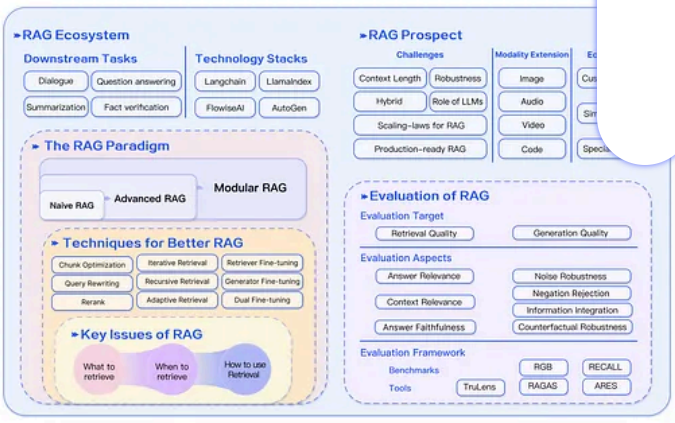


Figure 4: Summary of RAG ecosystem

The Figures in our RAG Survey

What is Modular RAG?

The progress of RAG has brought about a more diverse and flexible process, as evidenced by the following crucial aspects:

1. **Enhanced Data Acquisition:** RAG has expanded beyond traditional unstructured data and now includes semi-structured and structured data, with a focus on preprocessing structured data to improve retrieval and reduce the model's dependence on external knowledge sources.
2. **Incorporated Techniques**: RAG is integrating with other techniques, including the use of fine-tuning, adapter modules, and reinforcement learning to strengthen retrieval capabilities.
3. **Adaptable Retrieval Process**: The retrieval process has evolved to support multi-round retrieval enhancement, using retrieved content to guide generation and vice versa. Additionally, autonomous judgment and the use of LLM have increased the efficiency of answering questions by determining the need for retrieval.

18:01

Definition of Modular RAG

Above, we can see that the rapid development of RAG has surpassed the **Chain-style Advanced RAG** paradigm, showcasing a modular characteristic. To address the current lack of organization and abstraction, we propose a Modular RAG approach that seamlessly integrates the development paradigms of Naive RAG and Advanced RAG.

Modular RAG presents a highly **scalable** paradigm, dividing the RAG system into a **three-layer** structure of Module Type, Modules, and Operators. Each Module Type represents a core process in the RAG system, containing multiple functional modules. Each functional module, in turn, includes multiple specific operators. The entire RAG system becomes a permutation and combination of multiple modules and corresponding operators, forming what we refer to as RAG Flow. Within the Flow, different functional modules

can be selected in each module type, and within each functional module, one or more operators can be chosen.

The relationship with the previous paradigm

The Modular RAG organizes the RAG system in a multi-tiered modular form. Advanced RAG is a modular form of RAG, and Naive RAG is a special case of Advanced RAG. The relationship between the three paradigms is one of inheritance and development.

Opportunities in Modular RAG

The benefits of Modular RAG are evident, providing a fresh and comprehensive perspective on existing RAG-related work. Through modu organization, relevant technologies and methods are clearly summarized.

- **Research perspective**. Modular RAG is highly scalable, facilitating researchers to **propose new Module Types, Modules, and operators** based on a comprehensive understanding of the current RAG development.
- **Application perspective.** The design and construction of RAG systems become more convenient, allowing users to customize RAG Flow based on their existing data, usage scenarios, downstream tasks, and other requirements. Developers can also reference current Flow construction methods and **define new flow and patterns** based on different application scenarios and domains.

The Framework of Modular RAG

Module Type — Module — Operators

> *In this chapter, we will delve into the three-tier structure and constrcuct a technical roadmap for RAG. Due to space constraints, we will refrain from delving*

> *into technical specifics; however, comprehensive references will be provided for further reading.*

1. Indexing

Indexing, the process of breaking down text into manageable chunks, is a crucial step in organizing the system, facing three main challenges:

- **Incomplete Content Representation.**The semantic information of chunks is influenced by the segmentation method, resulting in the loss or submergence of important information within longer contexts.
- **Inaccurate Chunk Similarity Search.** As data volume increases, noise retrieval grows, leading to frequent matching with erroneous data, making the retrieval system fragile and unreliable.
- **Unclear Reference Trajectory.** The retrieved chunks may originate from any document, devoid of citation trails, potentially resulting in the presence of chunks from multiple different documents that, despite being semantically similar, contain content on entirely different topics.

Chunk Optimization

Larger chunks can capture more context, but they also generate more noise, requiring longer processing time and higher costs. While smaller chunks may not fully convey the necessary context, they do have less noise.

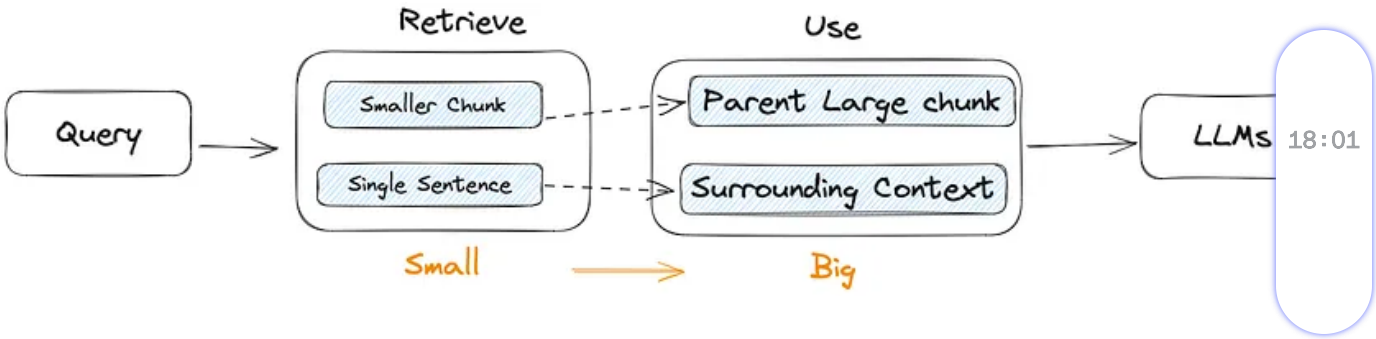
- *Sliding Window*

One simple way to balance these demands is to use overlapping chunks.By employing a sliding window, semantic transitions are enhanced. However,

limitations exist, including imprecise control over context size, the risk of truncating words or sentences, and a lack of semantic considerations.

- *Small-to-Big*

The key idea is to separate the chunks used for retrieval from the chunks used for synthesis. Using smaller chunks can improve the accuracy of retrieval, while larger chunks can provide more context information.



Specifically, one approach could involve retrieving **smaller chunks** and then **referencing parent IDs** to return larger chunks. Alternatively, individual sentences could be retrieved, and the **surrounding text window** of the sentence returned.

Detailed information and LlamaIndex Implementation.

Advanced RAG 01: Small-to-Big Retrieval

Child-Parent RecursiveRetriever and Sentence Window Retrieval with LlamaIndex

towardsdatascience.com

- *Summary*

It is akin to the Small-to-Big concept, where a summary of larger chunks is generated first, and the retrieval is performed on the summary. Subsequently, a secondary retrieval can be conducted on the larger chunks.

- **Metadata Attachment**

Chunks can be enriched with metadata information such as **page number**, **file name, author, timestamp, summary**, or the questions that the chunk can answer. Subsequently, retrieval can be filtered based on this metadata, limiting the scope of the search. See the implementation in LlamaIndex.

Structural Oraginzation

One effective method for enhancing information retrieval is to establish a hierarchical structure for the documents. By constructing chunks structu RAG system can expedite the retrieval and processing of pertinent data.

- *Hierarchical Index*

In the hierarchical structure of documents, nodes are arranged in parent-child relationships, with chunks linked to them. Data summaries are stored at each node, aiding in the swift traversal of data and assisting the RAG system in determining which chunks to extract. This approach can also mitigate the illusion caused by block extraction issues.

The methods for constructing a structured index primarily include：

- **Structural awareness.**paragraph and sentence segmentation in docs
- **Content awareness .**inherent structure in PDF, HTML, Latex

- **Semantic awareness.**Semantic recognition and segmentation of text based on NLP techniques, such as leveraging NLTK.

Check Arcus's hierarchical index at large-scale.

- *KG Organization Docs*

The utilization of Knowledge Graphs (KGs) in constructing the hierarchical structure of documents contributes to maintaining consistency. It delineates the connections between different concepts and entities, markedly reducing the potential for illusions.

Another advantage is the transformation of the information retrieval proc into instructions that LLM can comprehend, thereby enhancing the accuracy of knowledge retrieval and enabling LLM to generate contextually coherent responses, thus improving the overall efficiency of the RAG system.

Check Neo4j implementation and LlmaIndex Neo4j query engine.

For organizing multiple documents using KG, you can refer to this research paper **KGP:Knowledge Graph Prompting for Multi-Document Question Answering**.

Knowledge Graph Prompting: A New Approach for Multi-Document Question Answering

Multi-document question answering (MD-QA) involves answering questions that require synthesizing information across...

medium.com

2. Pre-Retrieval

One of the primary challenges with Naive RAG is its direct reliance on the user's orginal query as the basis for retrieval. Formulating a precise and clear question is difficult, and imprudent queries result in subpar retrieval effectiveness.

The primary challenges in this stage include:

- **Poorly worded queries.** The question itself is complex, and the langua is not well-organized.
- **language complexity & ambiguity.**Language models often struggle wh dealing with specialized vocabulary or ambiguous abbreviations with multiple meanings. For instance, they may not discern whether "LLM" refers to *large language model* or a *Master of Laws* in a legal context.

18:01

Query Expansion

Expanding a single query into multiple queries enriches the content of the query, providing further context to address any lack of specific nuances, thereby ensuring the optimal relevance of the generated answers.

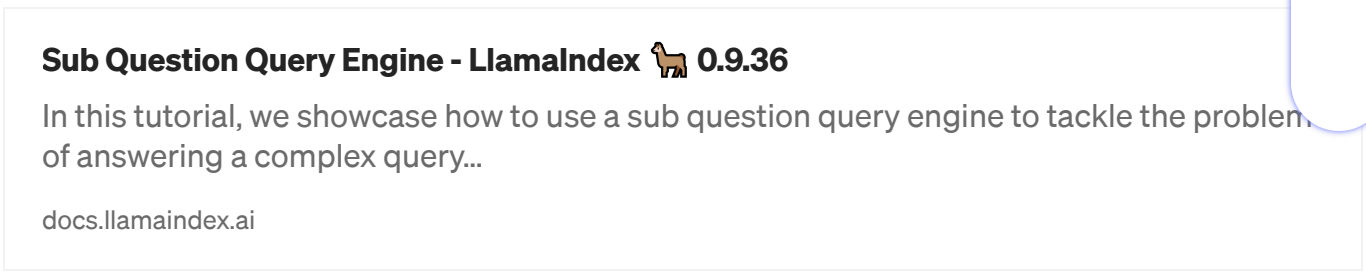
- *Multi-Query*

By employing prompt engineering to expand queries via LLMs, these queries can then be executed in parallel. The expansion of queries is not random, but rather meticulously designed. Two crucial criteria for this design are the **diversity** and **coverage** of the queries.

One of the challenges of using multiple queries is the potential **dilution** of the user's original intent. To mitigate this, we can instruct the model to assign greater weight to the original query in prompt engineering.

- *Sub-Query*

The process of sub-question planning represents the generation of the necessary sub-questions to contextualize and fully answer the original question when combined. This process of adding relevant context is, in principle, similar to query expansion. Specifically, a complex question can be decomposed into a series of simpler sub-questions using the **least-to-most prompting** method.



- *CoVe*

Another approach to query expansion involves the use of the **Chain-of-Verification(CoVe)** proposed by Meta AI. The expanded queries undergo validation by LLM to achieve the effect of reducing hallucinations. Validated expanded queries typically exhibit higher reliability.

Query Transformation

> *Retrieve and generate using a transformed query instead of the user's original query.*

- *Rewrite*

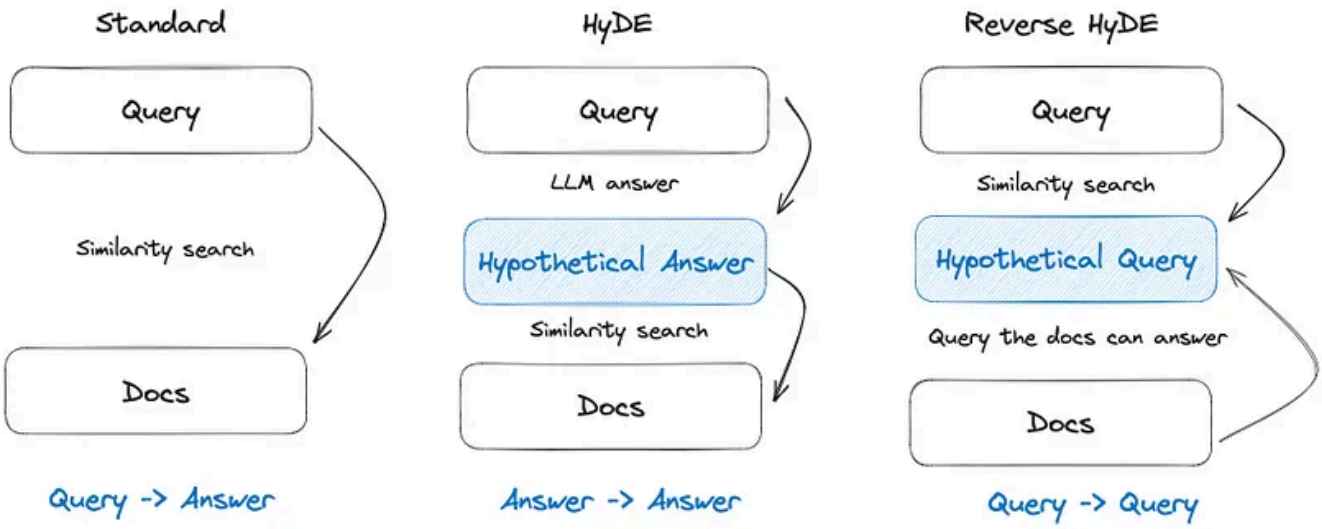
The original queries are not always optimal for LLM retrieval, especially in real-world scenarios. Therefore, we can prompt LLM to rewrite the queries. In addition to using LLM for query rewriting, specialized smaller language models, such as **RRR（Rewrite-retrieve-read)**, can also be utilized.

The implementation of the Query Rewrite method in the Taobao promotion system, known as **BEQUE:Query Rewriting for Retrieval-Augmented Large Language Models**, has notably enhanced recall effectiveness for long-tail queries, resulting in a rise in GMV.

18:01

- *HyDE*

When responding to queries, LLM constructs hypothetical documents (assumed answers) instead of directly searching the query and its computed vectors in the vector database. It focuses on embedding similarity from answer to answer rather than seeking embedding similarity for the problem or query. In addition, it also includes **Reverse HyDE**, which focuses on retrieval from query to query.



The core idea of bothHyDE and Reverse HyDE is to bridge the map between query and answer.

Advanced RAG — Improving retrieval using Hypothetical Document Embeddings(HyDE)

What is HyDE ?

medium.aiplanet.com

- *Step-back Prompting*

Using the Step-back Prompting method proposed by Google DeepMind, th original query is abstracted to generate a high-level concept question (ste back question). In the RAG system, both the step-back question and the original query are used for retrieval, and both the results are utilized as th basis for language model answer generation.

18:01

A New Prompt Engineering Technique Has Been Introduced Called Step-Back Prompting

Step-Back Prompting is a prompting technique enabling LLMs to perform abstractions, derive high-level concepts & first...

cobusgreyling.medium.com

Query Routing

Based on varying queries, routing to distinct RAG pipeline,which is suitable for a versatile RAG system designed to accommodate diverse scenarios.

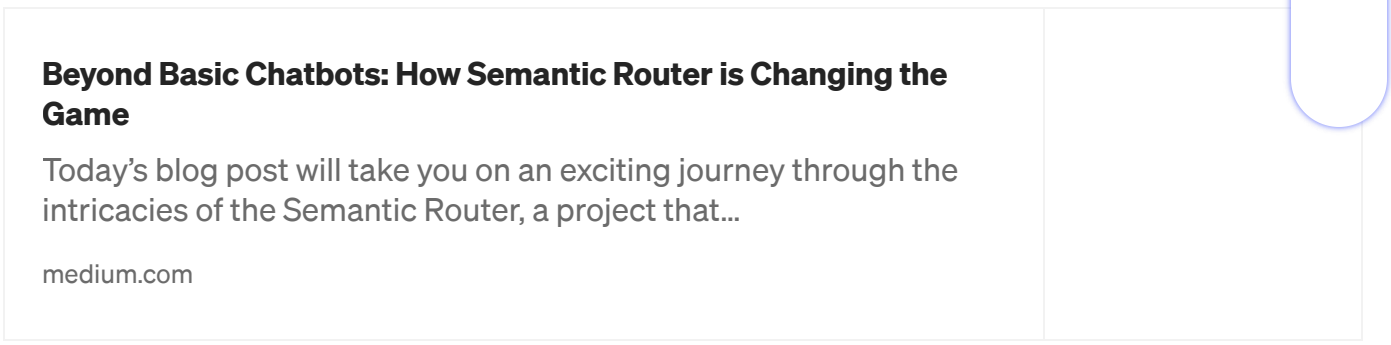
- *Metadata Router/ Filter*

The first step involves extracting keywords (entity) from the query, followed by filtering based on the keywords and metadata within the chunks to narrow down the search scope.

- *Semantic Router*

Another method of routing involves leveraging the semantic information of the query. Specific apporch see Semantic Router.Certainly, a hybrid routing approach can also be employed, combining both semantic and metadata-based methods for enhanced query routing.

Check Semantic router repo.



Query Construction

Converting a user's query into another query language for accessing alternative data sources. Common methods include:

- *Text-to-Cypher*
- *Text-to-SQL*

In many scenarios, structured query languages (e.g., SQL, Cypher) are often used in conjunction with semantic information and metadata to construct

more complex queries. For specific details, please refer to the Langchain blog.

Query Construction

Key Links * Text-to-metadata: Updated self-query docs and template * Text-to-SQL+semantic: Cookbook and template...

blog.langchain.dev

18:01

3 Retrieval

The retrieval process plays a crucial role in RAG. Leveraging powerful PL enables the effective representation of queries and text in latent spaces, facilitating the establishment of semantic similarity between questions and documents to support retrieval.

Three main considerations need to be taken into account :

- **Retrieval Efficiency**
- **Embedding Quality**
- **Alignment of tasks , data and models**

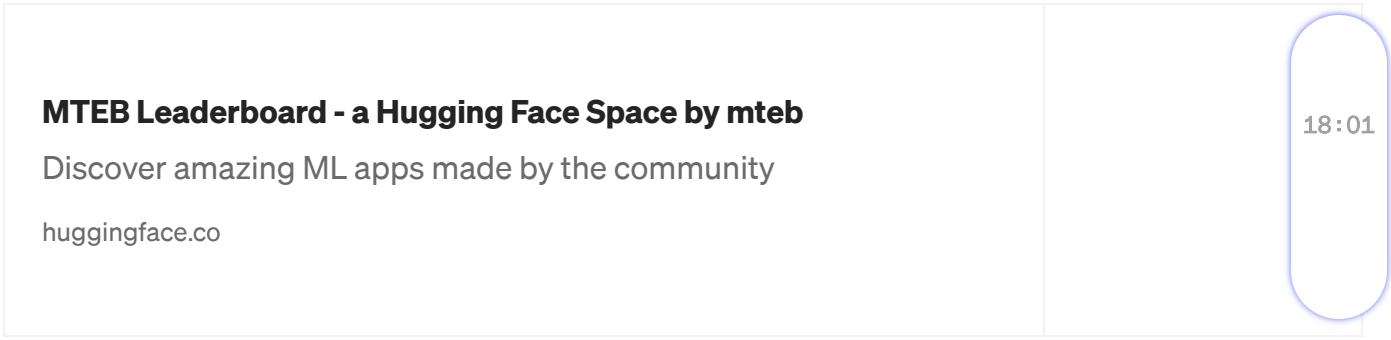
Retriver Selection

Since the release of ChatGPT, there has been a frenzy of development in embedding models.Hugging Face's **MTEB** leaderboard evaluates nearly all available embedding models across 8 tasks — Clustering,Classification,Bitext Ming, Pair Classification, Reranking, Retrieval, Semantic Text Similarity

(STS), and Summarization, covering 58 dataset Additionally, **C-MTEB** focuses on evaluating the capabilities of Chinese embedding models, covering 6 tasks and 35 datasets.

When constructing RAG applications, there is no one-size-fits-all answer to "which embedding model to use." However, you may notice that specific embeddings are better suited for particular use cases.

Check the MTEB/C-MTEB Leaderboard.



- *Sparse Retriever*

While sparse encoding models may be considered a somewhat antiquated technique, often based on statistical methods such as word frequency statistics, they still hold a certain place due to their higher encoding efficiency and stability. Common coefficient encoding models include **BM25** and **TF-IDF.**

- *Dense Retriever*

Neural network-based dense encoding models encompass several types:

- Encoder-Decoder language models built on the BERT architecture, such as ColBERT.

- Comprehensive multi-task fine-tuning models like BGE and Baichuan-Text-Embedding.
- Cloud API-based models such as OpenAI-Ada-002 and Cohere Embedding.
- Next-generation accelerated encoding framework Dragon+, designed for large-scale data applications.
- ***Mix/hybrid Retrieval***

Two embedding approaches capture different relevance features and can benefit from each other by leveraging complementary relevance information. For instance, sparse retrieval models can be used to provide initial search results for training dense retrieval models. Additionally, PL can be utilized to learn term weights to enhance sparse retrieval. Specifically, it also demonstrates that sparse retrieval models can enhance the zero-shot retrieval capability of dense retrieval models and assist dense retrievers in handling queries containing rare entities, thereby improving robustness.

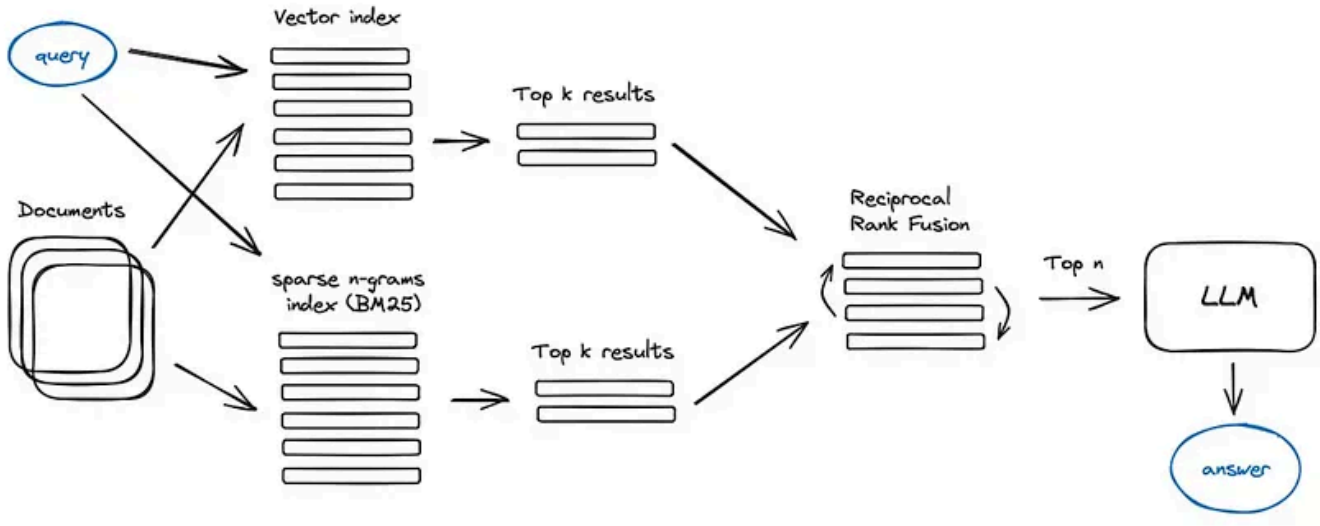


Image from IVAN ILIN:Advanced RAG Techniques: an Illustrated Overview

Retriever Fine-tuning

In cases where the context may diverge from what the pre-trained model deems similar in the embedding space, particularly in highly specialized fields like healthcare, law, and other domains abundant in proprietary terminology, adjusting the embedding model can address this issue. While this adjustment demands additional effort, it can substantially enhance retrieval efficiency and domain alignment.

- ***SFT***

You can construct your own fine-tuning dataset based on domain-specific data, a task that can be swiftly accomplished using LlamaIndex.

- ***LSR (LM-supervised Retriever)***

In contrast to directly constructing a fine-tuning dataset from the dataset, LSR utilizes the LM-generated results as supervisory signals to fine-tune the embedding model during the RAG process.

- **RL(**Reinforcement learning)

Inspired by RLHF(Reinforcement Learning fromHuman Feedback), utilizing LM-based feedback to reinforce the Retriever through reinforcement learning.

- **Adapter**

At times, fine-tuning an entire retriever can be costly, especially when dealing with API-based retrievers that cannot be directly fine-tuned. In such

cases, we can mitigate this by incorporating an adapter module and conducting fine-tuning.Another benefit of adding an adapter is the ability to achieve better alignment with specific downstream tasks.

- Task Specific.PRCA: Fitting Black-Box Large Language Models for Retrieval QuestionAnswering via Pluggable Reward-Driven Contextual Adapter.
- Task Agnostic.The AAR(Augmentation -Adapted Retriver) introduces a universal adapter designed to accommodate multiple downstream tasks.

4 Post-Retrieval

Retrieving entire document chunks and feeding them directly into the LL contextual environment is not an optimal choice. Post-processing the documents can aid LLM in better leveraging the contextual information.

The primary challenges include:

- **Lost in the middle**. Like humans, LLM tends to remember only the beginning and end of long texts, while forgetting the middle portion.
- **Noise/anti-fact chunks.** Retrieved noisy or factually contradictory documents can impact the final retrieval generation.
- **Context Window.** Despite retrieving a substantial amount of relevant content, the limitation on the length of contextual information in large models prevents the inclusion of all this content.

Rerank

Rerank the retrieved document chunks without altering their content or length, to enhance the visibility of the more crucial document chunks for

LLM. In specific terms：

- *Rule-base Rerank*

According to certain rules, metrics are calculated to rerank chunks. Common metrics include:

- Diversity
- Relevance
- MRR (Maximal Marginal Relevance, 1998)

The idea behind MMR is to reduce redundancy and increase result divers and it is used for text summarization. MMR selects phrases in the final ke phrase list based on a combined criterion of query relevance and information novelty.

Check there rerank implementation in HayStack

Enhancing RAG Pipelines in Haystack: Introducing DiversityRanker and LostInTheMiddleRanker

How the latest rankers optimize LLM context window utilization in Retrieval-Augmented Generation (RAG) pipelines

towardsdatascience.com

- *Model-base Rerank*

Utilize a language model to reorder the document chunks, with options including:

- Encoder-Decoder models from the BERT series, such as SpanBERT
- Specialized reranking models, such as Cohere rerank or bge-raranker-large
- General large language models, such as GPT-4

Compression and Selection

A common misconception in the RAG process is the belief that retrieving as many relevant documents as possible and concatenating them to form a lengthy retrieval prompt is beneficial. However, excessive context can introduce more noise, diminishing the LLM's perception of key informati and leading to issues such as " lost in the middle" . A common approach t address this is to compress and select the retrieved content.

18:01

- ***(Long)LLMLingua***

By utilizing aligned and trained small language models, such as GPT-2 Small or LLaMA-7B, the detection and removal of unimportant tokens from the prompt is achieved, transforming it into a form that is challenging for humans to comprehend but well understood by LLMs. This approach presents a direct and practical method for prompt compression, eliminating the need for additional training of LLMs while balancing language integrity and compression ratio.

check the LLMLingua project.

LLMLingua | Explore the special language for LLMs via Prompt Compression

Prompt Compression

Prompt Compressionwyydsb.xin

- *Recomp*

Recomp introduces two types of compressors: an **extractive compressor** that selects pertinent sentences from retrieved documents, and an **abstractive compressor** that produces concise summaries by amalgamating information from multiple documents. Both compressors are trained to enhance the performance of language models on end tasks when the generated summaries are prepended to the language models' input, while ensuring the conciseness of the summary. In cases where the retrieved documents are irrelevant to the input or do not provide additional information to the language model, compressor can return an empty string, thereby implementing selective augmentation.

- *Selective Context*

By **identifying and removing redundant content in the input context**, the input can be streamlined, thus improving the language model's reasoning efficiency. Selective Context is akin to a "stop-word removal" strategy. In practice, selective context assesses the information content of lexical units based on the self-information computed by the base language model. By retaining content with higher self-information, this method offers a more concise and efficient textual representation for language model processing, without compromising their performance across diverse applications. However, it overlooks the interdependence between compressed content and the alignment between the targeted language model and the small language model utilized for prompting compression.

- *Tagging-Filter*

Tagging is a relatively intuitive and straightforward approach. Specifically, the documents are first labeled, and then filtered based on the metadata of the query.

Tagging | 🦜🔗 Langchain

Open In Colab

python.langchain.com

- **LLM-Critique**

Another straightforward and effective approach involves having the LLM evaluate the retrieved content before generating the final answer. This allows the LLM to filter out documents with poor relevance through LLM critique. For instance, in Chatlaw, the LLM is prompted to self-suggestion on the referenced legal provisions to assess their relevance.

5 Generation

Utilize the LLM to generate answers based on the user's query and the retrieved context information.

Generator Selection

Depending on the scenario, the choice of LLM can be categorized into the following two types:

- *Cloud API-base Generator*

Cloud API-based Utilize third-party LLMs by invoking their APIs, such as OpenAI's ChatGPT, GPT-4, and Anthropic Claude, among others. **Benefits:**

- No server pressure
- High concurrency
- Ability to use more powerful models

Drawbacks:

- Data passes through third parties, leading to data privacy concerns
- Inability to adjust the model (in the vast majority of cases)
- *On-Premises*

Locally deployed open-source or self-developed LLMs, such as the Llama series, GLM, and others.The advantages and disadvantages are opposite to those of Cloud API-based models. Locally deployed models offer greater flexibility and better privacy protection but require higher computational resources.

Generator Fine-tuning

In addition to directl LLM usage, targeted fine-tuning based on the scenario and data characteristics can yield better results. This is also one of the greatest advantages of using an on-premise setup. Common fine-tuning methods include the following:

- *SFT*

When LLMs lack data in a specific domain, additional knowledge can be provided to the LLM through fine-tuning. Huggingface's fine-tuning data can also be used as an initial step.

Another benefit of fine-tuning is the ability to adjust the model's input and output. For example, it can enable LLM to adapt to specific data formats and generate responses in a particular style as instructed.

- ***RL***

Aligning LLM outputs with human or retriever preferences through reinforcement learning is a potential approach. For instance, manually annotating the final generated answers and then providing feedback thro reinforcement learning. In addition to aligning with human preferences, also possible to align with the preferences of fine-tuned models and retrievers.

- ***Distillation***

When circumstances prevent access to powerful proprietary models or larger parameter open-source models, a simple and effective method is to distill the more powerful models(e.g. GPT-4).

- ***Dual FT***

Fine-tuning both Generator and Retriever to align their preferences. A typical approach, such as *RA-DIT*, aligns the scoring functions between Retriever and Generator using KL divergence.

6 Orchestration

Orchestration refers to the modules used to control the RAG process. RAG no longer follows a fixed process, and it involves making decisions at key points and dynamically selecting the next step based on the results. This is also one of the key features of modularized RAG compared to Naive RAG.

Scheduling

The Judge module assesses critical point in the RAG process, determining the need to retrieve external document repositories, the satisfaction of the answer, and the necessity of further exploration. It is typically used in recursive, iterative, and adaptive retrieval. Specifically, it mainly includes following two operators:

- *Rule-base*

The next course of action is determined based on predefined rules. Typically, the generated answers are scored, and then the decision to continue or stop is made based on whether the scores meet predefined thresholds. Common thresholds include confidence levels for tokens.

- *Prompt-base*

LLM autonomously determines the next course of action. There are primarily two approaches to achieve this. The first involves prompting LLM to reflect or make judgments based on the conversation history, as seen in the ReACT framework. The benefit here is the elimination of the need for fine-tuning the model. However, the output format of the judgment depends on the LLM's adherence to instructions. A prompt-base case is FLARE.

- *Tuning-base*

The second approach entails LLM generating specific tokens to trigger particular actions, a method that can be traced back to Toolformer and is applied in RAG, such as in Self-RAG.

Fusion

This concept originates from RAG Fusion. As mentioned in the previous section on *Query Expansion*, the current RAG process is no longer a singular pipeline. It often requires the expansion of retrieval scope or diversity through multiple branches. Therefore, following the expansion to multip branches, the Fusion module is relied upon to merge multiple answers.

18:01

- *Possibility Ensemble*

The fusion method is based on the weighted values of different tokens generated from multiple beranches, leading to the comprehensive selection of the final output. Weighted averaging is predominantly employed. See REPLUG.

- **RRF (Reciprocal Rank Fusion)**

RRF, is a technique that combines the rankings of multiple search result lists to generate a single unified ranking. Developed in collaboration with the University of Waterloo (CAN) and Google, RRF produces results that are more effective than reordering chunks under any single branch.

Forget RAG, the Future is RAG-Fusion

The Next Frontier of Search: Retrieval Augmented Generation meets Reciprocal Rank Fusion and Generated Queries

towardsdatascience.com

Advanced RAG Techniques: an Illustrated Overview

A comprehensive study of the advanced retrieval augmented generation techniques and algorithms, systemising various...

pub.towardsai.net

18:01

Conclusion

The upcoming content on RAG Flow will be introduced in PART II, to be published soon.

As this is my first time publishing an article on Medium, I am still getting familiar with many features. Any feedback and criticism are welcome.

Large Language Models | Rag

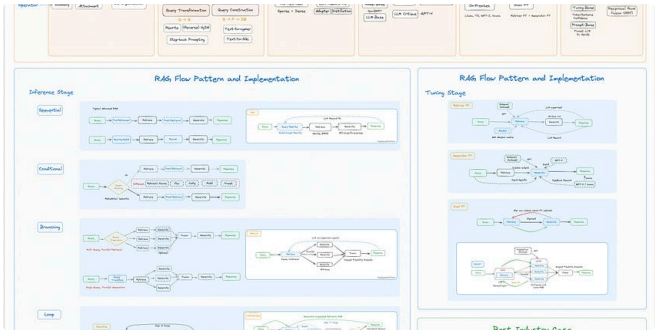
Written by Yunfan Gao

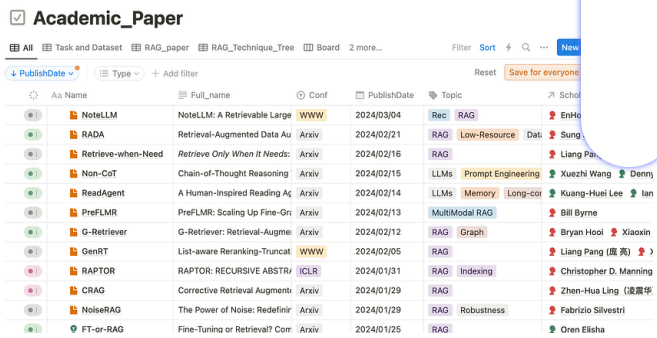
89 Followers

Cs PhD Candidate in Tongji University

Follow

More from Yunfan Gao





Yunfan Gao

Modular RAG and RAG Flow: Part II

Unsure how to design your RAG flow? Take a look at Modular RAG Flow.

12 min read · Jan 29, 2024

112 3

Yunfan Gao

OpenRAG Base: Your individual RAG Knowledge Base

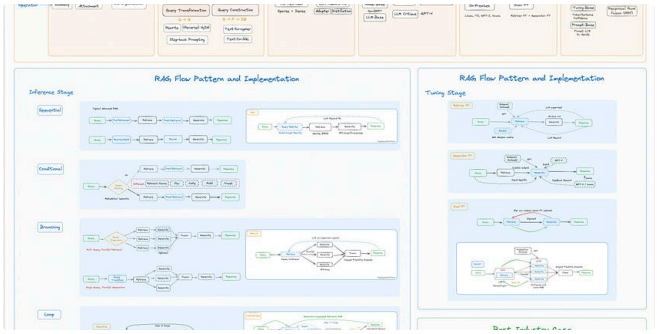
We're officially launching the RAG Knowledge Base: OpenRAG Base 🎉✨

9 min read · Apr 3, 2024

16

See all from Yunfan Gao

Recommended from Medium



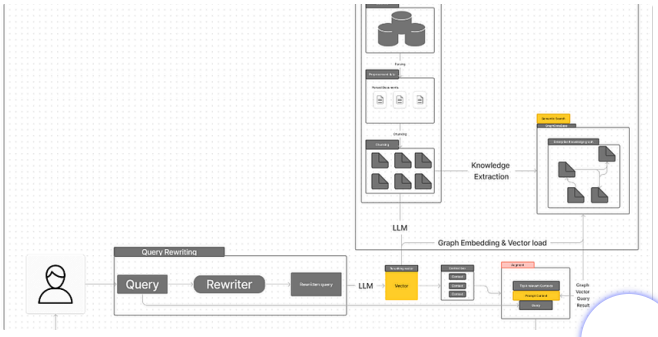
Yunfan Gao

Modular RAG and RAG Flow: Part II

Unsure how to design your RAG flow? Take a look at Modular RAG Flow.

12 min read · Jan 29, 2024

112 3



Jeong ii tae

From RAG to GraphRAG , What i the GraphRAG and why i use it?

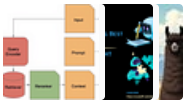
Before discussing RAG and GraphRAG,

13 min read · Mar 12, 2024

126 1

18:01

Lists

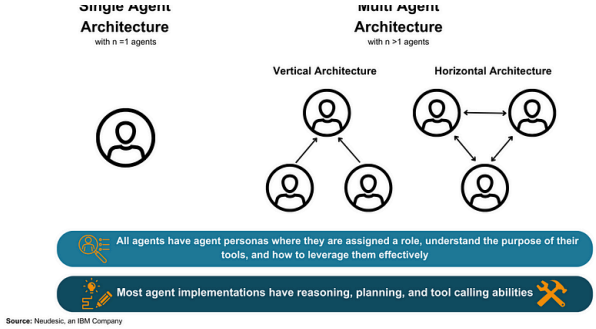


Natural Language Processing
1417 stories · 910 saves

AI Regulation
6 stories · 430 saves

ChatGPT prompts
47 stories · 1496 saves

Generative AI Recommended Reading
52 stories · 984 saves



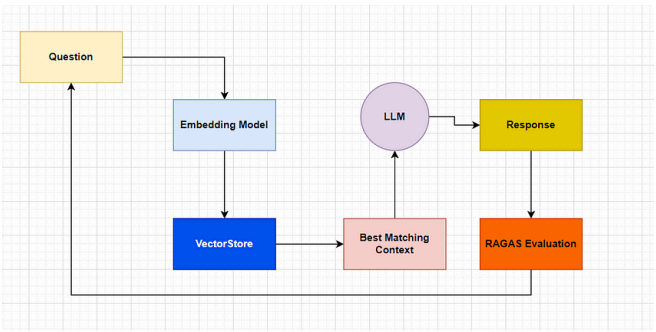
Sandi Besen in Towards Data Science

The Landscape of Emerging AI Agent Architectures for Reasonin...

We set out to uncover the key design elements for AI Agents to effectively execut...

7 min read · Apr 23, 2024

192 2



Plaban Nayak in AI Planet

Evaluating Naive RAG and Advanced RAG pipeline using...

What is RAG(Retrieval Augmented Generation) ?

25 min read · Feb 11, 2024

289 4

18:01

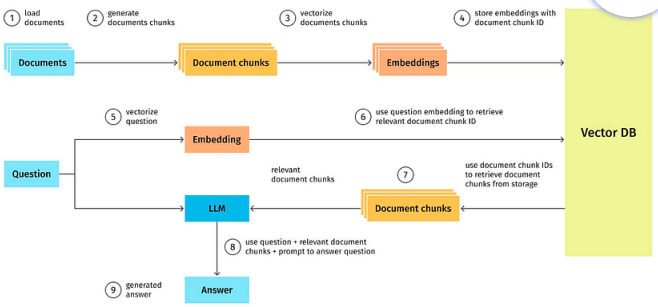
IVAN ILIN in Towards AI

Advanced RAG Techniques: an Illustrated Overview

A comprehensive study of the advanced retrieval augmented generation techniques...

19 min read · Dec 17, 2023

5.5K 36



Chia Jeng Yang in WhyHow.AI

A first intro to Complex RAG (Retrieval Augmented Generation)

Understanding basic technical concepts of RAG, and unsolved opportunities & problem...

11 min read · Dec 14, 2023

909 5

See more recommendations

18:01