

# Introduction to Retrieval-Augmented Generation (RAG)

Kun-Ta Chuang



## Outline

- Introduction
- Related Work
- Methodology
- Experiment
- Conclusion
- References

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

Yunfan Gao<sup>a</sup>, Yun Xiong<sup>b</sup>, Xinyu Gao<sup>b</sup>, Kangxiang Jia<sup>b</sup>, Jinliu Pan<sup>b</sup>, Yuxi Bi<sup>c</sup>, Yi Dai<sup>a</sup>, Jiawei Sun<sup>a</sup>, Meng Wang<sup>c</sup>, and Haofen Wang<sup>a,c</sup>

<sup>a</sup>Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University

<sup>b</sup>Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University

<sup>c</sup>College of Design and Innovation, Tongji University

Main Content from

Gao, Yunfan et al. “Retrieval-Augmented Generation for Large Language Models: A Survey.” (2023)

## Introduction

**Large Language Models (LLMs) have achieved remarkable performance in various natural language processing tasks, but they still face challenges.** These limitations include:

- **Hallucination:** LLMs can generate text that sounds plausible but is factually incorrect.
- **Outdated knowledge:** LLMs are trained on massive datasets, but these datasets might not capture the most recent information.
- **Non-transparent reasoning:** The reasoning process behind LLM outputs can be unclear and difficult to trace.

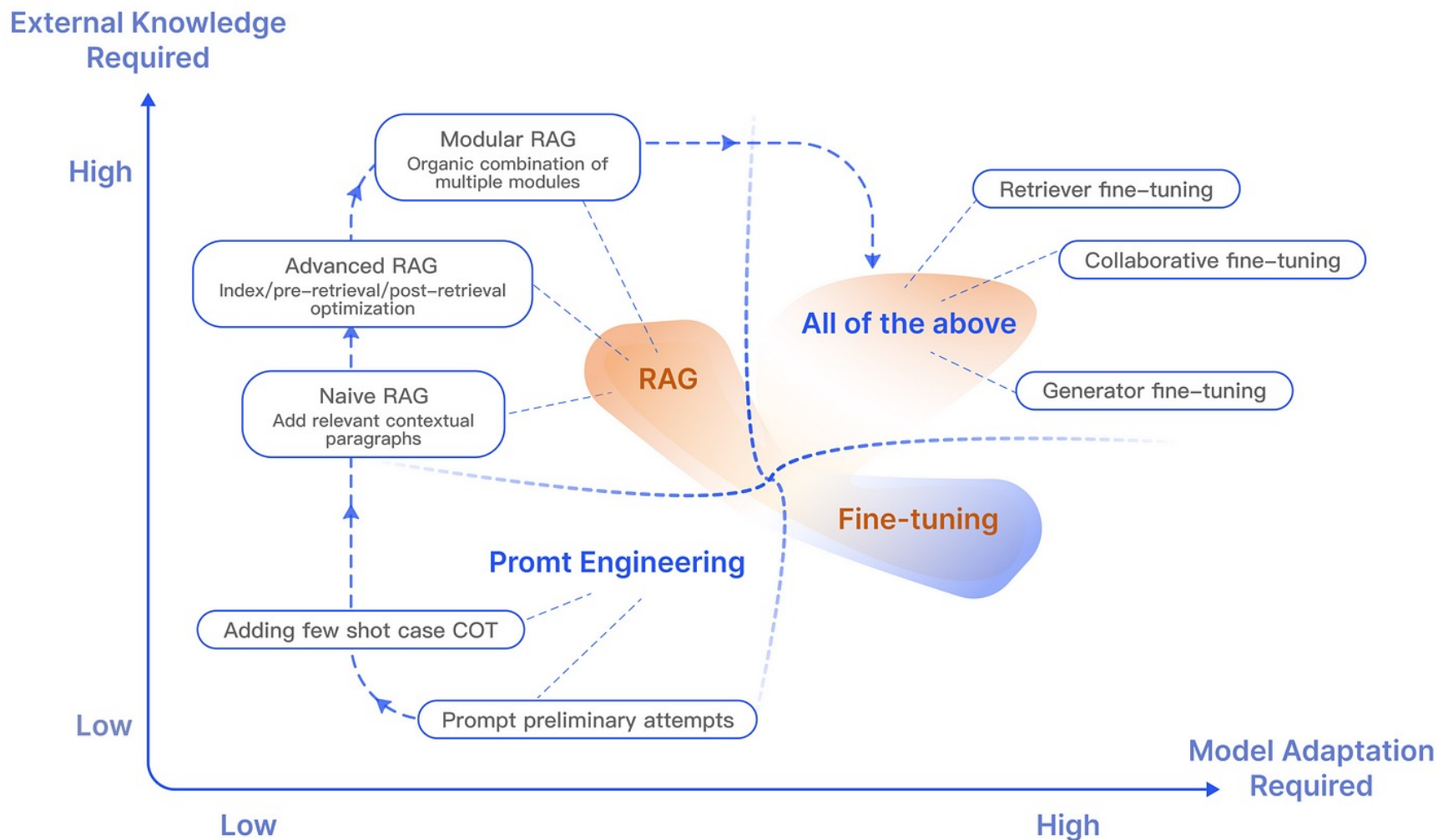
## Introduction

**Challenge:** Large Language Models (LLMs) are powerful but struggle with factual accuracy, outdated information, and unclear reasoning.

**Solution:** Retrieval-Augmented Generation (RAG) tackles these issues by integrating information from external databases with LLMs. This improves:

- **Accuracy and credibility:** LLMs become more reliable, especially for tasks requiring factual knowledge.
- **Knowledge updates:** Information in the external databases can be easily updated, keeping the LLM current.
- **Domain-specific information:** Specific knowledge relevant to different domains can be incorporated.

# Process Integration: Fine-Tuning -> Prompt Engineering -> RAG



## Introduction

**Retrieval-Augmented Generation (RAG) emerges as a promising approach to address these limitations.** RAG systems combine LLMs with **external knowledge sources** accessed through retrieval techniques. Advantages include:

1. RAG improves the accuracy of answers by utilizing external knowledge, effectively reducing misinformation and making the generated responses more accurate and reliable.
2. The use of retrieval techniques enables the identification of the latest information (provided by users), ensuring that the LLM's responses remain timely.
3. RAG references information sources that users can verify, resulting in high transparency and increasing trust in the model's output.
4. By retrieving information from domain-specific data, RAG can provide expert knowledge support for various fields, offering high customization capabilities.
5. In terms of security and privacy management, RAG stores knowledge in a database, providing better control over data usage. In contrast, models fine-tuned for specific tasks often lack clear management of data access permissions, making them prone to data leaks, which is a significant concern for enterprises.
6. Since RAG does not require updating model parameters, it offers greater economic efficiency when handling large datasets.

# ► What is RAG?

## 1. Retrieving relevant information: (Retrieval)

- When a user asks a question, RAG uses search algorithms to find relevant information from an external knowledge base.
- This information can be in various forms, ranging from simple tokens to structured data like knowledge graphs.
- Research is ongoing to determine the **optimal granularity (level of detail)** and structure of retrieved information for different tasks.

## 2. Enriching the LLM's prompt: (Augmentation)

- The retrieved information is then combined with the original user query to create a more informative prompt for the LLM.
- This enriched prompt provides the LLM with additional context to understand the user's intent and generate a more accurate and informative response.

## ► What is RAG?

### 3. Generating a response: (Generation)

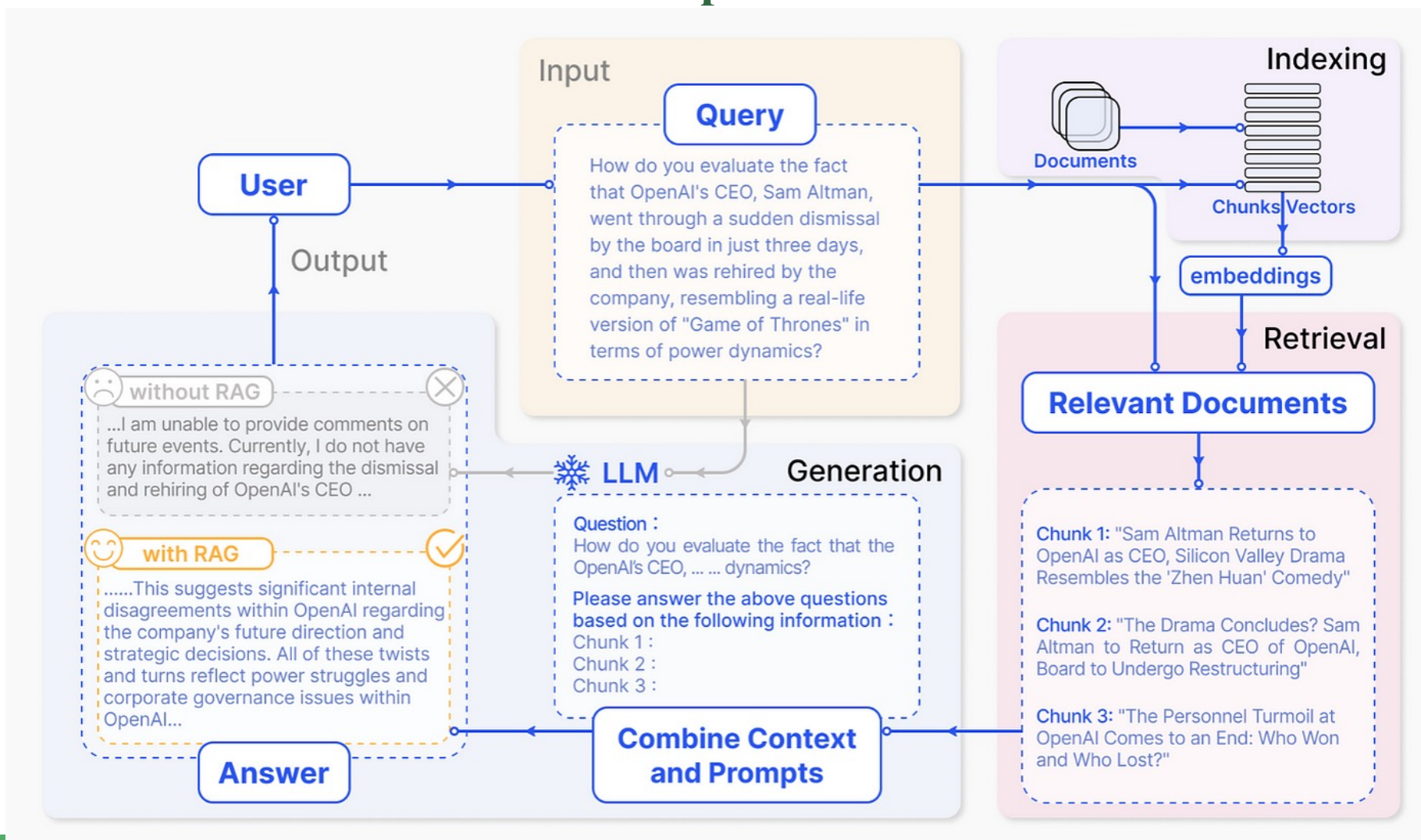
- The LLM, using the enriched prompt, generates a response to the user's query.

#### Advantages of RAG:

- **No need for retraining:** Unlike traditional methods, RAG doesn't require retraining the LLM for every specific task. Instead, it simply adds an external knowledge base, improving the model's output without retraining.
- **High practicality and low barrier to entry:** This advantage makes RAG popular in building conversational products, as it requires minimal modifications to existing LLM systems.



## What is RAG? – Illustrative Example



# Technology Tree of RAG Research

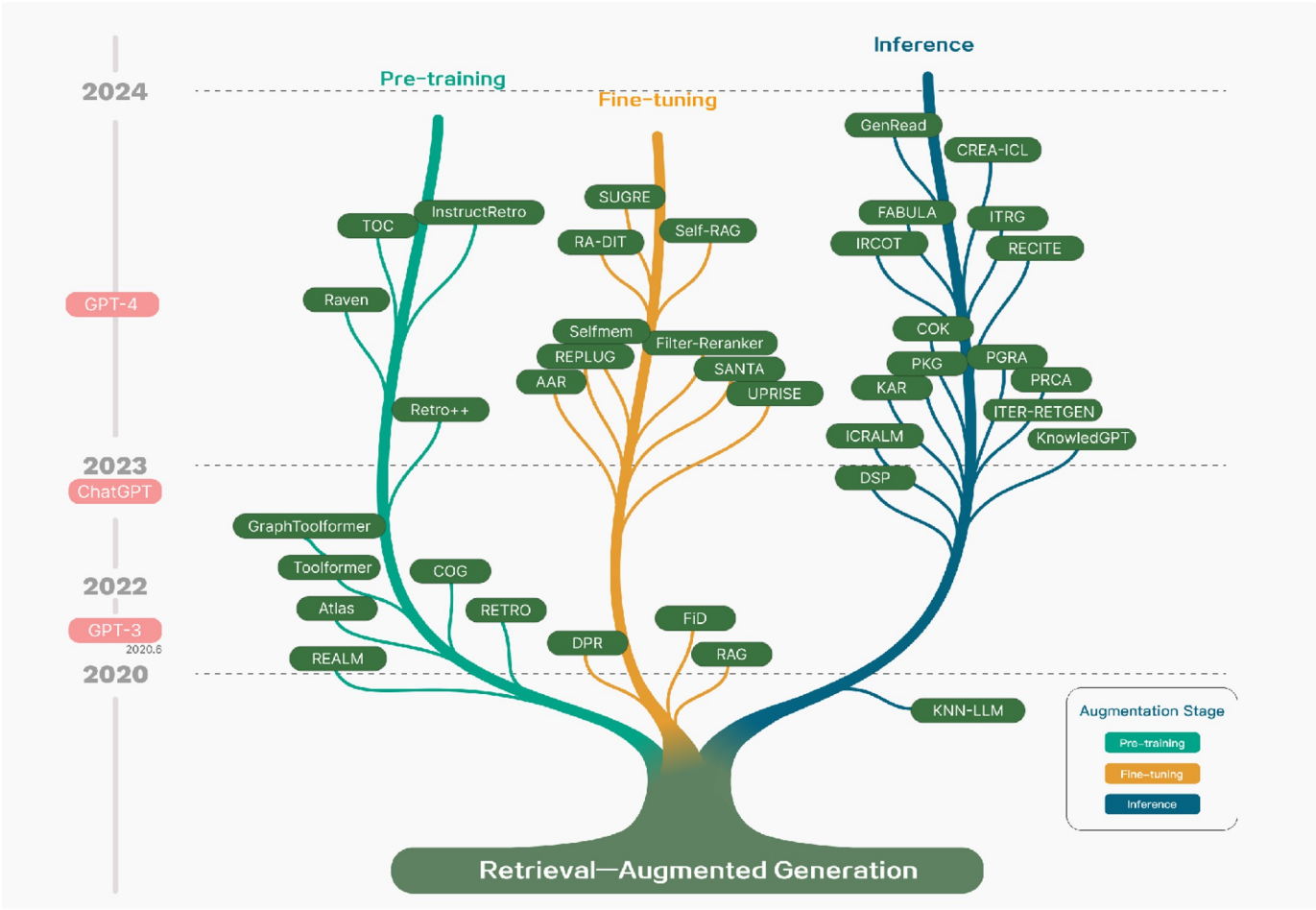
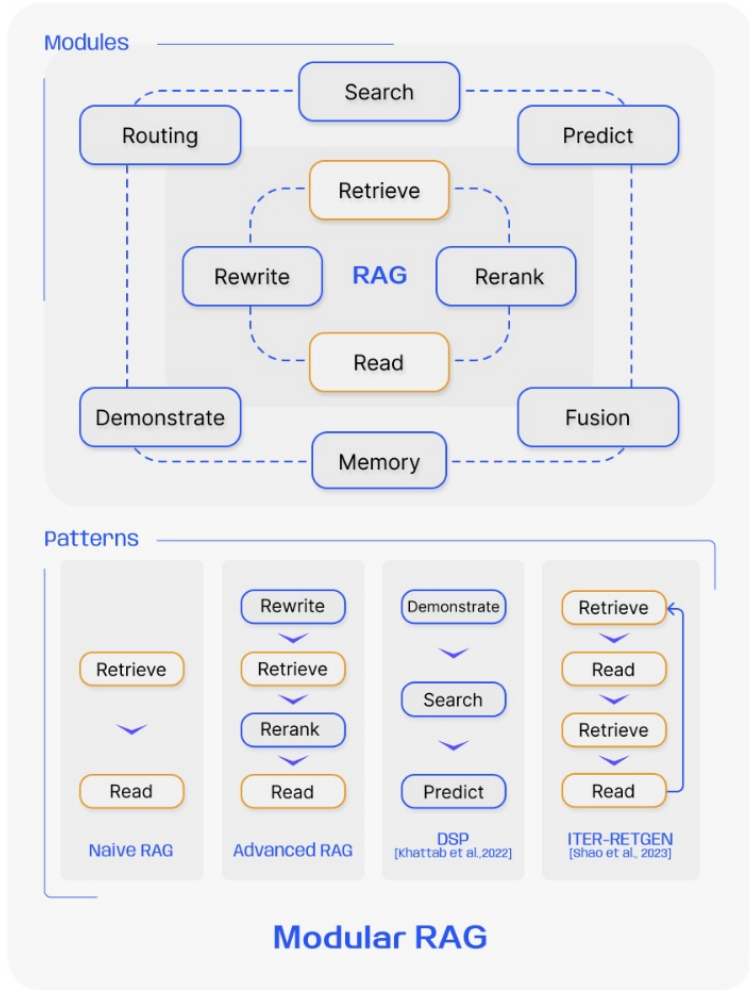
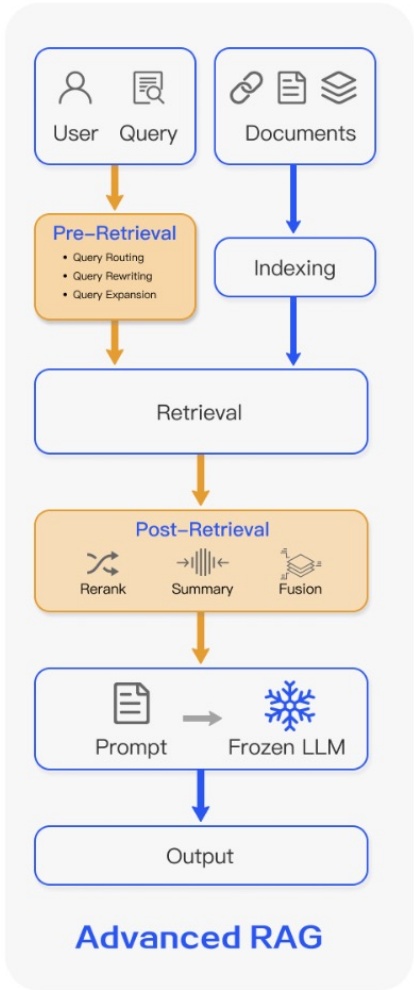
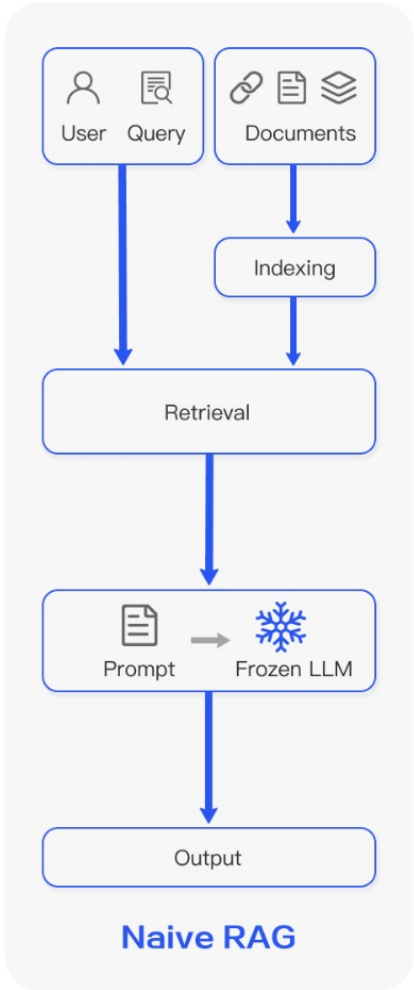


Figure 1: Technology tree of RAG research development featuring representative works

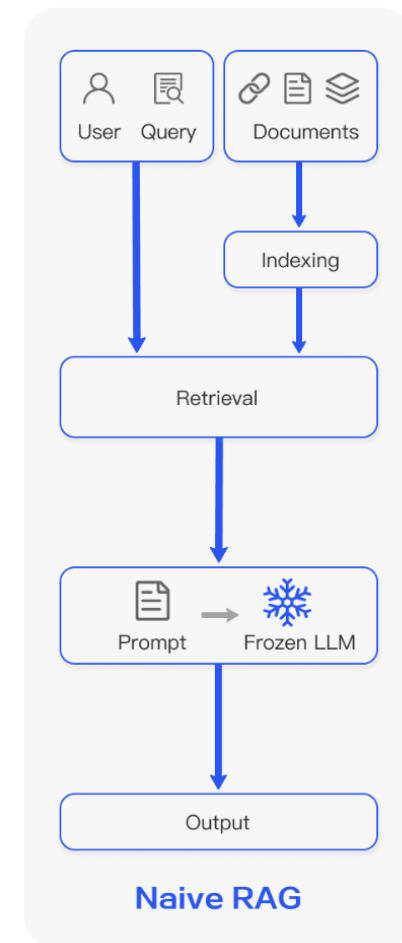
# Paradigms of RAG



# Paradigms of RAG

## 1. Naive RAG:

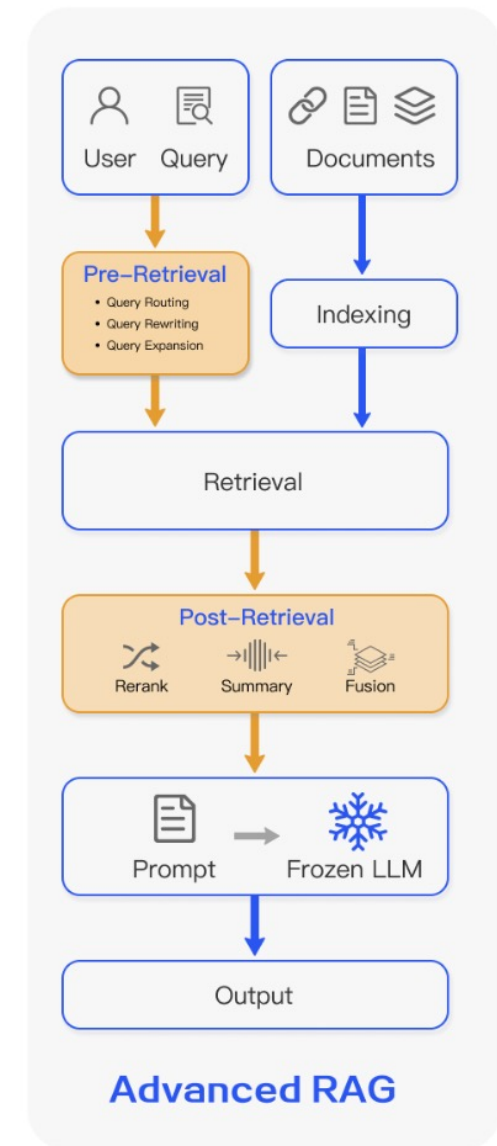
- This is the **initial and basic form** of RAG.
- It involves a **fixed and simple workflow** with pre-defined modules for retrieving information and integrating it with the LLM.
- Doesn't delve deep into details of Naive RAG. Involve basic retrieval techniques and limited options for customizing the process.



# Paradigms of RAG

## 2. Advanced RAG:

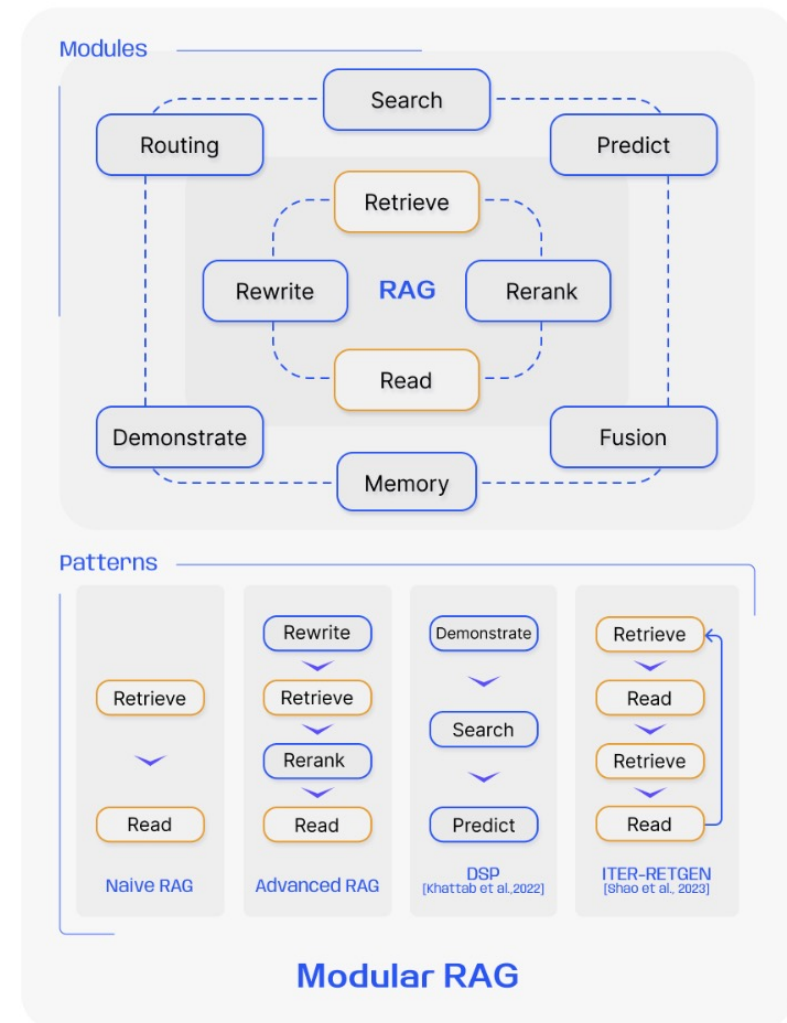
- This stage builds upon Naive RAG by addressing its limitations and **enhancing the retrieval process**.
- It focuses on improving the **quality and efficiency** of retrieved information.
- This includes:
  - **Optimizing pre-retrieval processes** like data indexing and selection.
  - **Improving the retrieval models** themselves for better relevance selection.



# Paradigms of RAG

## 3. Modular RAG:

- This is the most **flexible and advanced** form of RAG.
- It introduces **independent modules** that can be **added, removed, or modified** to customize the information retrieval and integration process.
- These modules handle specific tasks like searching, memory management, and information fusion.
- Both **Naive RAG** and **Advanced RAG** are **considered special cases of Modular RAG** as they have fixed modules and limited customizability.



# Naive RAG

## 1. Indexing:

- Prepare data by:
  - Cleaning and extracting information.
  - Converting documents to plain text.
  - Chunking text into smaller segments.
  - Creating vector representations of chunks for efficient search.

## 2. Retrieval:

- Upon receiving a query, it:
  - Converts the query into a vector representation.
  - Finds the **top K** chunks most similar to the query's vector.
  - These retrieved chunks become the context for response generation.

## 3. Generation:

- Combines the query and retrieved chunks into a prompt for the LLM.
- The LLM generates a response based on the prompt, potentially using its own knowledge or focusing on the retrieved information.




## Naive RAG

### Drawbacks of Naive RAG:

- **Retrieval:**
  - **Low precision:** Retrieved chunks might not accurately match the query, leading to irrelevant information or "hallucinations" in the response.
  - **Low recall:** Not all relevant information might be retrieved, hindering comprehensive responses.
  - **Outdated information:** Retrieved information might not be up-to-date, leading to inaccurate responses.
- **Generation:**
  - **Hallucination:** Generation of answers not grounded in the provided context.
  - **Irrelevant context:** Inclusion of irrelevant information in the response.
  - **Toxicity or bias:** Potential for biased or harmful content in the output.
- **Augmentation:**
  - **Disjointed or incoherent output:** Difficulty in integrating retrieved information seamlessly, leading to a lack of flow in the response.
  - **Redundancy and repetition:** Repetitive content due to similar information in multiple retrieved chunks.
  - **Difficulties in balancing and reconciling retrieved passages:** Challenges in determining the relevance of each passage and ensuring consistent style in the output.
  - **Over-reliance on retrieved information:** LLM might simply repeat the retrieved information without providing new insights.



## Naive RAG

1. Limited Integration: Naive RAG typically uses simple concatenation of retrieved documents and the query, leading to shallow integration of information.
  2. Quality of Retrieval: The retrieval process may bring in irrelevant or low-quality documents, which can degrade the generation quality.
  3. Efficiency: Combining retrieval and generation without optimization can be computationally expensive and slow.
  4. Robustness: It may be less robust to noisy or incomplete retrieved information, resulting in less reliable outputs.
- 

## Advanced RAG

**Advanced RAG** tackles the limitations of Naive RAG, particularly in **retrieval quality**. It employs strategies before and after the retrieval process:

### **Pre-Retrieval:**

- **Optimizes data indexing:**
  - Enhances data quality by improving standardization, consistency, accuracy, and context richness.
  - Optimizes index structures for efficient retrieval by adjusting chunk size, utilizing metadata, and leveraging graph structures.
- **Addresses alignment issues:**
  - Introduces "hypothetical questions" to rectify inconsistencies between documents.

## Advanced RAG

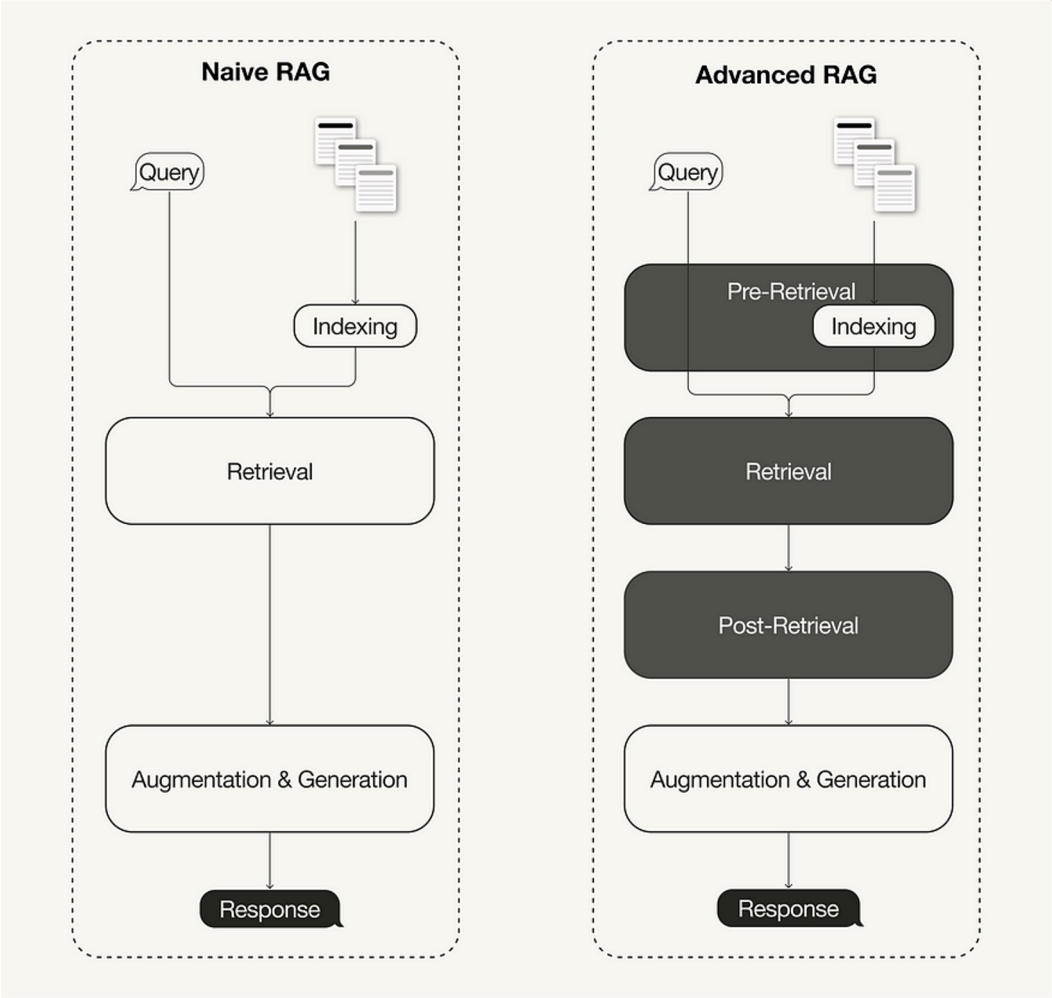
### Retrieval:

- **Fine-tunes embedding models:**
  - Customizes models for specific domains, enhancing retrieval relevance (e.g., BGE-large-EN model).
- **Employs dynamic embedding:**
  - Considers word context for improved relevance (e.g., OpenAI's embeddings-ada-02 model).

### Post-Retrieval:

- **Re-ranks retrieved information:**
  - Prioritizes the most relevant content within the LLM's context window (e.g., Diversity Ranker, LostInTheMiddleRanker).
- **Compresses the prompt:**
  - Reduces noise and highlights key information (e.g., Selective Context, LLMLingua).

# ➤ Comparisons



## Modular RAG

**Modular RAG** surpasses Naive and Advanced RAG by offering a **flexible and adaptable framework**. It allows:

- **Integration of various modules:**
  - Search module for specific scenario searches.
  - Memory module to utilize LLM memory for retrieval.
  - Fusion module to enhance search by expanding queries.
  - Routing module to direct queries to appropriate data sources.
  - Predict module to utilize LLM generation for context.
  - Task adapter module for adapting RAG to specific tasks.
- **Modification of existing modules:**
  - Swapping or rearranging modules for specific needs.
  - Adding or removing modules to enhance functionalities.
- **Adjusting flow between modules:**
  - Enhancing interaction between LLMs and retrieval models.
  - Using one module's output to improve another's functionality.

## Modular RAG

### Optimizing the RAG Pipeline:

- **Hybrid search:** Combines different search techniques for diverse queries.
- **Recursive retrieval:** Retrieves smaller chunks initially followed by larger ones for context.
- **StepBack-prompt approach:** Encourages LLMs to reason around broader concepts.
- **Sub-queries:** Utilizes various query strategies based on the scenario.
- **Hypothetical Document Embeddings:** Uses LLM-generated answers to retrieve similar real documents.

## Enhancing Semantic Representations

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

## ► Optimizing Chunks for Effective RAG Systems

Effective RAG systems rely on well-chunked documents for efficient and accurate retrieval. Choosing the optimal chunk size is crucial to avoid issues like:

- **Overly large chunks:** Can miss granular details and hinder embedding representation.
- **Excessively small chunks:** May lack context and lead to irrelevant retrievals.

### Factors affecting chunk size selection:

- **Content nature:** Long documents may require different chunking compared to short ones.
- **Embedding model:** Different models perform better with specific block sizes (e.g., sentence-transformer vs text-embedding-ada-002).
- **Query complexity:** Complex queries might benefit from smaller chunks for focused retrieval.
- **Application use case:** Semantic search might require different chunking strategies than question answering.
- **LLM limitations:** Token limits of LLMs might necessitate adjustments.



## Optimizing Chunks for Effective RAG Systems

### Chunk optimization techniques:

- **Sliding window:** Enables layered retrieval by merging information from multiple retrievals.
- **Small2big:** Uses small chunks for initial search and provides larger relevant chunks later.
- **Abstract embedding:** Focuses on document summaries for comprehensive context understanding.
- **Metadata filtering:** Leverages document metadata for enhanced filtering.
- **Graph indexing:** Transforms entities and relationships into nodes and connections, improving relevance for multi-hop problems.

## ► Fine-tuning Embedding Models

Effective RAG systems rely on accurate **embedding models** to represent the corpus and enable efficient retrieval based on semantic similarity. This section explores crucial aspects of embedding in RAG:

### Popular Embedding Models:

- **Pre-trained models like Angle, Voyage, BGE:** Offer strong performance but may lack domain-specificity.

### Fine-tuning Embedding Models:

- **Importance:** Enhances the model's ability to understand user queries and retrieve relevant information.
- **Two primary paradigms:**
  - **Domain Knowledge Fine-tuning:**
    - Uses domain-specific datasets for fine-tuning.
    - Requires queries, a corpus, and relevant documents.
    - Tools like LlamaIndex can simplify the process.
  - **Fine-tuning for Downstream Tasks:**
    - Leverages LLMs to improve model performance.
    - Examples:
      - **PROMPTAGATOR:** Utilizes LLMs for creating task-specific retrievers.
      - **LLM-Embedder:** Enhances fine-tuning with LLM-generated reward signals.

## Fine-tuning Embedding Models

### Challenges and Considerations:

- **Compatibility with LLMs:** Fine-tuned retrievers may not always be compatible with specific LLMs.
- **Direct LLM Supervision:** Exploring direct LLM feedback to improve alignment and performance.

# ► Aligning Queries and Documents

## 1. Query Rewriting:

- Aims to bridge the gap between the query's semantics and the documents' semantics.
- Common methods include:
  - **Query2Doc and ITER-RETGEN:** Use LLMs to create a pseudo-document combining the query with additional guidance.
  - **HyDE:** Constructs query vectors using textual cues to generate a "hypothetical" document.
  - **RRR:** Reverses the typical workflow, focusing on query rewriting before retrieval.
  - **STEP-BACKPROMPTING:** Enables LLMs to perform abstract reasoning based on broader concepts.
  - **Multi-query retrieval:** Generates and executes multiple search queries simultaneously for complex problems.

## 2. Embedding Transformation:

- Focuses on fine-tuning query embeddings to better align with the document space.
- Example: **LlamaIndex adapter module:** Integrates after the query encoder for task-specific fine-tuning.

## Addressing Structured Data:

- **SANTA:** Enhances the retriever's understanding of structured information through pre-training strategies:
  - **Structured-aware pre-training:** Leverages the inherent alignment between structured and unstructured data.
  - **Masked Entity Prediction:** Uses entity masking to encourage LLMs to predict and understand entities within structured data.

# Aligning Retriever and LLM

## 1. Fine-tuning Retrievers with LLM Feedback:

- **AAR:** Uses LLM scores to identify preferred documents and fine-tune the retriever for unseen target LLMs.
- **REPLUG:** Utilizes an LLM as the supervisory signal for training the retrieval model, improving performance through KL divergence calculation.
- **UPRISE:** Employs frozen LLMs to fine-tune the prompt retriever, treating the LLM as a "data labeler."
- **Atlas:** Offers four supervised fine-tuning methods:
  - **Attention Distillation:** Transfers knowledge from LLM attention scores.
  - **EMDR2:** Trains using retrieved documents as latent variables.
  - **PerplexityDistillation:** Trains using generated token perplexity.
  - **LOOP:** Utilizes a novel loss function based on LLM prediction impact.

## 2. Adapters for Enhanced Alignment:

- **PRCA:** Optimizes the retriever's output using an adapter trained through context extraction and reward-driven phases.
- **Token filtering:** Employs LLM cross-attention scores to efficiently select high-scoring input tokens.
- **RECOMP:** Introduces extractive and generative compressors for summarizing relevant information from retrieved documents.
- **PKG:** Integrates knowledge into white-box models by directly substituting the retriever module with an LLM-guided document generation approach.

## Generation

### Key Features of RAG's Generator:

- **Incorporates Retrieved Data:** Unlike traditional models, RAG's generator uses both contextual information and relevant text segments retrieved by the retriever.
- **Improves Accuracy and Relevance:** This comprehensive input allows the generator to grasp the query context more comprehensively, leading to more informative and relevant responses.
- **Ensures Coherence:** The generator is guided by retrieved text, ensuring coherence between the generated content and the underlying information.

## ► Post-retrieval with Frozen LLM

### Goals of Post-retrieval Processing:

- **Improve information quality:** Align retrieved information more closely with user needs and subsequent tasks.
- **Address LLM limitations:** Mitigate issues like limited context length and sensitivity to redundancy.

### Common Techniques:

- **Information compression:**
  - **Condensing retrieved information:** Reduce noise, address context limitations, and improve generation effectiveness.
    - **Examples:**
      - PRCA: Trains an information extractor to create condensed contexts from retrieved documents.
      - RECOMP: Uses contrastive learning to train an information condenser.
  - **Reducing the number of documents:** Improve answer accuracy by focusing on the most relevant documents.
    - **Example:** Filter-Reranker: Combines SLMs for filtering and LLMs for re-ranking to prioritize relevant documents.
- **Reranking:**
  - **Reorder retrieved documents:** Prioritize the most relevant documents at the top to improve efficiency and effectiveness.
  - **Example:** Re-ranking models act as optimizers and refiners, providing better input for LLMs.
  - **Contextual compression:** Reduce content within documents and filter irrelevant information to present the most relevant information.

## Fine-tuning LLM for RAG

### Key Points:

- **Unique Input Format:** Unlike standard LLMs, RAG's generator takes both the query and retrieved documents as input.
- **Importance of Fine-tuning:** Fine-tuning helps the generator adapt to this specific input format and ensure the generated text effectively utilizes the retrieved information.
- **Post-retrieval Processing:** Typically, retrieved documents undergo processing before being fed to the fine-tuned model.

### General Fine-tuning Process:

- **Training Data:** Consists of input-output pairs, where the input includes the query and retrieved documents, and the output is the desired text response.
- **Fine-tuning Paradigms:**
  - **Joint-Encoder:** Uses a single encoder followed by an attention-based decoder for generating output tokens.
  - **Dual-Encoder:** Employs separate encoders for the query/context and the document, with subsequent cross-attention processing by the decoder.
- **Optimization:** Utilizes Negative Log-Likelihood loss for both paradigms.



## Fine-tuning LLM for RAG

### Addressing Exposure Bias:

- **Traditional training methods** can lead to exposure bias, limiting the model's exposure to diverse outputs.
- **Contrastive Learning:** Techniques like SURGE employ graph-text contrastive learning to encourage the model to generate a wider range of plausible and coherent responses, reducing overfitting and improving generalization.

### Fine-tuning for Structured Data Retrieval:

- **SANTA framework:** Implements a tripartite training process:
  - **Phase 1:** Contrastive learning refines query and document embeddings for the retriever.
  - **Phase 2:** Contrastive learning aligns structured data with unstructured document descriptions for the generator.
  - **Phase 3:** Masked Entity Prediction enhances the generator's understanding of entity semantics
    - Entities are identified and masked in the input data.
    - The model learns to predict and reconstruct the masked entities, improving its grasp of structural information.

## Fine-tuning LLM for RAG

### Addressing Exposure Bias:

- **Traditional training methods** can lead to exposure bias, limiting the model's exposure to diverse outputs.
- **Contrastive Learning:** Techniques like SURGE employ graph-text contrastive learning to encourage the model to generate a wider range of plausible and coherent responses, reducing overfitting and improving generalization.

### Fine-tuning for Structured Data Retrieval:

- **SANTA framework:** Implements a tripartite training process:
  - **Phase 1:** Contrastive learning refines query and document embeddings for the retriever.
  - **Phase 2:** Contrastive learning aligns structured data with unstructured document descriptions for the generator.
  - **Phase 3:** Masked Entity Prediction enhances the generator's understanding of entity semantics
    - Entities are identified and masked in the input data.
    - The model learns to predict and reconstruct the masked entities, improving its grasp of structural information.

## Augmentation

- **The Augmentation Stage:** When and how augmentation is applied in the overall RAG training pipeline.
- **Sources of Augmentation Data:** Where the additional data for augmentation comes from.
- **The Augmentation Process:** Techniques used to modify the original training data.

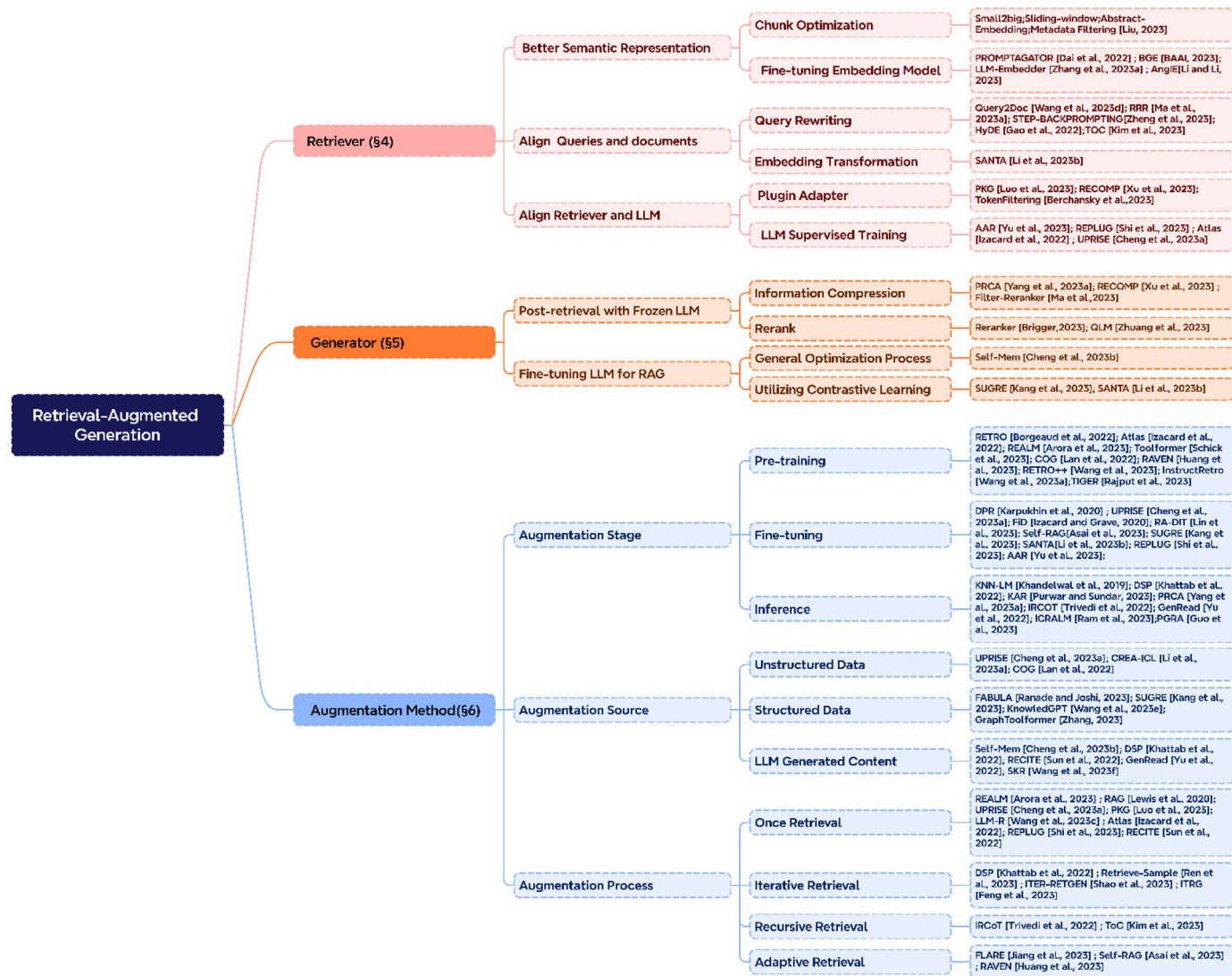


Figure 4: Taxonomy of RAG’s core components

## ➡ RAG Augmentation in Different Stages: Pre-training

### Pre-training:

- **Goal:** Bolster pre-trained language models (PTMs) for open-domain question answering (QA) through retrieval-based strategies.
- **Benefits:**
  - Improved **perplexity** (measure of model uncertainty).
  - Enhanced **text generation quality**.
  - Reduced **toxicity**.
  - Better performance in **downstream tasks**, especially knowledge-intensive ones like open-domain QA.
  - Development of **domain-specific models** through specialized corpora training.
  - **Robust foundational model** exceeding standard GPT models.
- **Challenges:**
  - Requires **extensive pre-training datasets and resources**.
  - **Decreased update frequency** with larger models.

## ► RAG Augmentation in Different Stages: Pre-training

- **Examples:**

- **REALM:** Structured, interpretable method for knowledge embedding.
- **RETRO:** Leverages retrieval augmentation for large-scale pre-training.
- **Atlas:** Incorporates retrieval into T5 architecture for pre-training and fine-tuning.
- **COG:** Novel text generation method that emulates copying text fragments.
- **RETRO++:** Scales up RETRO model size while maintaining or improving performance.

## RAG Augmentation in Different Stages: Fine-Tuning

- **Combining RAG and fine-tuning** allows the system to address specific needs like stylistic adjustments and aligning retrieved documents with the query.
- **Fine-tuning the retriever:**
  - **Improves semantic representations:** Enhances the quality of retrieved documents by fine-tuning the embedding model.
  - **Aligns retriever with LLMs:** Improves model synergy by adapting the retriever's outputs to LLM preferences.
  - **Increases adaptability:** Benefits downstream tasks by fine-tuning for specific contexts.
  - **Offers task-agnostic fine-tuning:** Enhances versatility for multi-task scenarios.
- **Fine-tuning the generator:**
  - **Customization:** Enables adjustments for specific input data formats, like knowledge graphs or text pairs.
  - **Stylistic control:** Generates content in specific formats through directed datasets.
  - **Improved content determination:** Fine-tunes LLMs to generate content for adaptive retrieval scenarios.

### Synergistic Fine-tuning:

- **Joint fine-tuning:** Improves generalization capabilities and avoids overfitting, but requires more resources.
- **RA-DIT:** Provides a lightweight alternative for adding retrieval functionalities to LLMs.

## ■ RAG Augmentation in Different Stages: Fine-Tuning

- **Limitations:** Requires specialized datasets and significant resources.
- **Benefits:**
  - Customizes models to specific needs and data formats.
  - Potentially reduces resource usage compared to pre-training.
  - Fine-tunes output style.



## RAG Augmentation in Different Stages: Inference Stage

The **inference stage** is where RAG models interact with LLMs to generate text. This section explores how advanced techniques enrich the context during this crucial stage:

### Traditional vs. Advanced Approaches:

- **Naive RAG:** Incorporates retrieved content directly for generation, but offers limited context.
- **Advanced Techniques:** Introduce richer contextual information during inference:
  - **DSP:** Enables natural language exchange between LLMs and retrieval models, enhancing context.
  - **PKG:** Equips LLMs with a knowledge-guided module for retrieving relevant information without parameter modification.
  - **CREA-ICL:** Employs synchronous retrieval of cross-lingual knowledge for richer context.
  - **RE-CITE:** Generates context by directly sampling paragraphs from LLMs.

## ➤ RAG Augmentation in Different Stages: Inference Stage

### Multi-step Reasoning Tasks:

- **ITRG:** Iteratively retrieves information to identify correct reasoning paths, improving task adaptability.
- **ITER-RETGEN:** Employs an iterative cycle of "retrieval-enhanced generation" and "generation-enhanced retrieval".

### Non-Knowledge Intensive Tasks:

- **PGRA:** Uses a two-stage approach:
  - A task-agnostic retriever identifies relevant documents.
  - A prompt-guided reranker prioritizes the evidence.
- **IRCOT:** Combines RAG with Chain of Thought (CoT) methodologies, enhancing GPT-3's performance in question-answering.

## RAG Augmentation in Different Stages: Inference Stage

### Benefits and Challenges:

- **Benefits:**

- Leverages pre-trained models without further training.
- Maintains static LLM parameters while providing task-specific context.
- Cost-effective and lightweight.

- **Challenges:**

- Requires meticulous data processing and optimization.
- Limited by the underlying LLM's capabilities.
- Often requires pairing with procedural optimization techniques (step-wise reasoning, iterative retrieval) for diverse tasks.

## Augmentation Source

### Data Categories:

- **Unstructured data:** Textual data like corpora, prompt data, and cross-lingual data.
- **Structured data:** Knowledge graphs (KGs) providing high-quality context.
- **LLM-generated content:** Content created by the LLM itself.

## Augmentation Source

### Unstructured Data Augmentation:

- **Examples:**
  - Prompt data for fine-tuning large models.
  - Cross-lingual data.
- **Retrieval Granularity:**
  - Tokens (e.g., kNN-LM).
  - Phrases (e.g., NPM, COG).
  - Document paragraphs (finer granularity offers precision but increases retrieval complexity).
- **Techniques:**
  - **FLARE:** Active retrieval triggered by low-probability words.
  - **RETRO:** Retrieves nearest neighbors at the chunk level based on previous context.

## Augmentation Source

### Structured Data Augmentation:

- **Benefits:**
  - Provides high-quality context.
  - Mitigates model hallucinations.
- **Examples:**
  - **RET-LLMs:** Constructs a knowledge graph memory from past dialogues.
  - **SUGRE:** Uses GNNs to encode relevant KG subgraphs, ensuring consistency between retrieved information and generated text.
  - **KnowledGPT:** Generates KB search queries and stores knowledge in a personalized base.

## Augmentation Source

### LLM-Generated Content Augmentation:

- **Rationale:** Addresses limitations of external information.
- **Examples:**
  - **SKR:** Classifies questions and applies retrieval enhancement selectively.
  - **GenRead:** Uses LLM-generated contexts, often containing more accurate answers due to alignment with pre-training objectives.
  - **Selfmem:** Creates an unbounded memory pool with a retrieval-enhanced generator, using a memory selector to choose outputs that enhance the generative model.

# ► Augmentation Process in RAG Systems: Addressing Inefficiencies

## Challenges of Single Retrieval:

- **"Lost in the middle" phenomenon:** Redundant information from a single retrieval can dilute or contradict essential details, degrading generation quality.
- **Limited scope for complex problems:** Single retrieval provides insufficient information for multi-step reasoning tasks.

## Advanced Retrieval Methods:

- **Iterative Retrieval:**
  - Allows for multiple retrieval cycles.
  - Improves information depth and relevance.
  - Example: FLARE (introduced in Section 6.2) iteratively retrieves based on LLM-generated content.
- **Recursive Retrieval:**
  - Uses the results of one retrieval as input for the next.
  - Dives deeper into relevant information for complex queries.
  - Useful for tasks requiring a gradual approach to reach a final answer (e.g., research, legal analysis, data mining).
- **Adaptive Retrieval:**
  - Dynamically adjusts the retrieval process based on task and context.
  - Tailors information retrieval to specific needs.



# ► Iterative Retrieval in RAG Systems: Deepening the Knowledge Base

## What is Iterative Retrieval?

- It involves **repeatedly collecting documents** based on the initial query and the generated text so far.
- This progressively builds a **more comprehensive knowledge base** for the LLM.

## Benefits:

- **Enhanced robustness:** Provides additional contextual references for subsequent answer generation.

## Challenges:

- **Semantic discontinuity:** Sequence-based separation between generated text and retrieved content might disrupt meaning flow.
- **Irrelevant information accumulation:** Retrieving based on generated text can potentially introduce irrelevant content.

## Iterative Retrieval in RAG Systems: Deepening the Knowledge Base

### Addressing Challenges:

- **Recursive retrieval:** Utilizes a structured index for hierarchical processing.
  - It can involve summarizing sections before retrieval, refining the search within the document.
  - This is suited for complex data like lengthy PDFs.
- **Multi-hop retrieval:** Designed for graph-structured data sources, extracting interconnected information.

### Synergistic Approaches:

- **ITER-RETGEN:** Combines "retrieval-enhanced generation" and "generation-enhanced retrieval".
  - The model uses the task's required content to retrieve relevant knowledge.
  - This retrieved information then aids in generating improved responses in subsequent iterations.

## Recursive Retrieval: Refining Search Queries for Deeper Understanding

### What is Recursive Retrieval?

- It is a process of **iteratively refining search queries** based on the information retrieved in previous rounds.
- This creates a **feedback loop** that gradually converges on the most relevant information.

### Benefits:

- **Improved depth and relevance:** Enables diving deeper into the search space.
- **Enhanced search experience:** Leads to more pertinent results, especially for complex or nuanced queries.

### Example Techniques:

- **IRCoT:** Uses chain-of-thought to guide retrieval and refine it with retrieved results.
- **ToC:** Creates a clarification tree to systematically optimize ambiguity in the query.

## ► Recursive Retrieval: Refining Search Queries for Deeper Understanding

### Applications:

- **Complex search scenarios:** When the user's needs are unclear initially or the information sought is highly specialized.
- **Continuous learning and refinement:** Allows for adapting the search based on retrieved information, potentially improving user satisfaction.

### Comparison with Iterative Retrieval:

- **Similarities:**
  - Both involve multiple retrieval steps.
  - Both aim to improve information depth and relevance.
- **Differences:**
  - **Iterative Retrieval:** Uses generated text from previous steps to guide subsequent retrievals.
  - **Recursive Retrieval:** Uses retrieved information from previous steps to refine the original query.

## Adaptive Retrieval in RAG Systems: Empowering LLMs for Selective Information Retrieval

### What is Adaptive Retrieval?

- It allows LLMs to judge **optimally when and what content** to retrieve during the generation process.
- This enhances the **efficiency and relevance** of retrieved information.

### Examples:

- **Flare:** Triggers retrieval based on LLM-generated content's confidence level (probability).
- **Self-RAG:** Uses "reflection tokens" ("retrieve" and "critic") for the LLM to:
  - Decide **when** to activate retrieval.
  - Conduct **fragment-level beam search** for the most coherent sequence during retrieval.
  - Update retrieval based on **critic scores** (adjustable during inference).

## ► Adaptive Retrieval in RAG Systems: Empowering LLMs for Selective Information Retrieval

### Adaptive Retrieval aligns with a broader trend:

- LLMs employing **active judgment** in operations, similar to model agents (AutoGPT, Toolformer, Graph-Toolformer).
- **Examples:**
  - **Graph-Toolformer:** Decides when to search, use retrievers, and employ prompts.
  - **WebGPT:** Utilizes reinforcement learning to train GPT-3 to autonomously search during generation.

### Benefits:

- **Optimized retrieval cycles:** Retrieves information only when necessary.
- **Improved model autonomy:** LLMs become better at judging when external information is needed.

## RAG Evaluation

- **Objectives of RAG Evaluation:**
  - Comprehend strengths and weaknesses.
  - Identify areas for improvement.
  - Compare different models and approaches.
- **Aspects Assessed:**
  - Retrieval performance: Measuring retrieved information's relevance and factuality.
  - Generation performance: Evaluating the quality and coherence of generated text using the retrieved information.
  - Overall system performance: Combining retrieval and generation metrics.
- **Benchmarks and Tools:**
  - Limited dedicated benchmarks and tools for RAG evaluation.
  - RALLE is an example tool for automatic evaluation based on task-specific metrics.

## Evaluation Aspects

- **RAG Evaluation Targets:**
  - Retrieval Quality: How well the model finds relevant information.
  - Generation Quality: How well the model uses retrieved information to create accurate responses.
- **Quality Scores:**
  - Context Relevance: The retrieved information is precise and targeted to the question.
  - Answer Faithfulness: The generated answer accurately reflects the retrieved information.
  - Answer Relevance: The generated answer addresses the core of the question.
- **Required Abilities:**
  - Noise Robustness: Can the model handle irrelevant but potentially related information?
  - Negative Rejection: Can the model recognize when it should not generate a response?
  - Information Integration: Can the model successfully combine information from multiple sources?
  - Counterfactual Robustness: Can the model identify and disregard known inaccuracies?



## ► Future Challenges

- **Context Length:** Striking the right balance between too much and too little context is key as LLM context windows expand.
- **Robustness:** RAG needs to become more resistant to noise and misinformation during retrieval.
- **RAG + Fine-tuning:** Finding the best ways to combine these techniques (sequentially, alternating, or joint training) is an active research area.
- **Expanding LLM Roles:** LLMs can do more than just generate answers within RAG frameworks - how can we further leverage their potential?
- **Scaling Laws:** Do established LLM scaling laws apply to RAG? The potential of an inverse scaling law is particularly interesting.
- **Production-Readiness:** Enhancing retrieval efficiency, improving data security, and real-world alignment are crucial for wider RAG adoption.

## ► Modality Expansion

- **Image:** Models like RA-CM3, BLIP-2, and "Visualize Before You Write" integrate images and text in novel ways.
- **Audio & Video:** RAG techniques are used in speech translation, automatic speech recognition (ASR), and video understanding tasks.
- **Code:** Models like RBPS assist developers by retrieving relevant code examples.
- **Structured Knowledge:** Approaches like CoK integrate knowledge graphs to enhance question-answering capabilities.

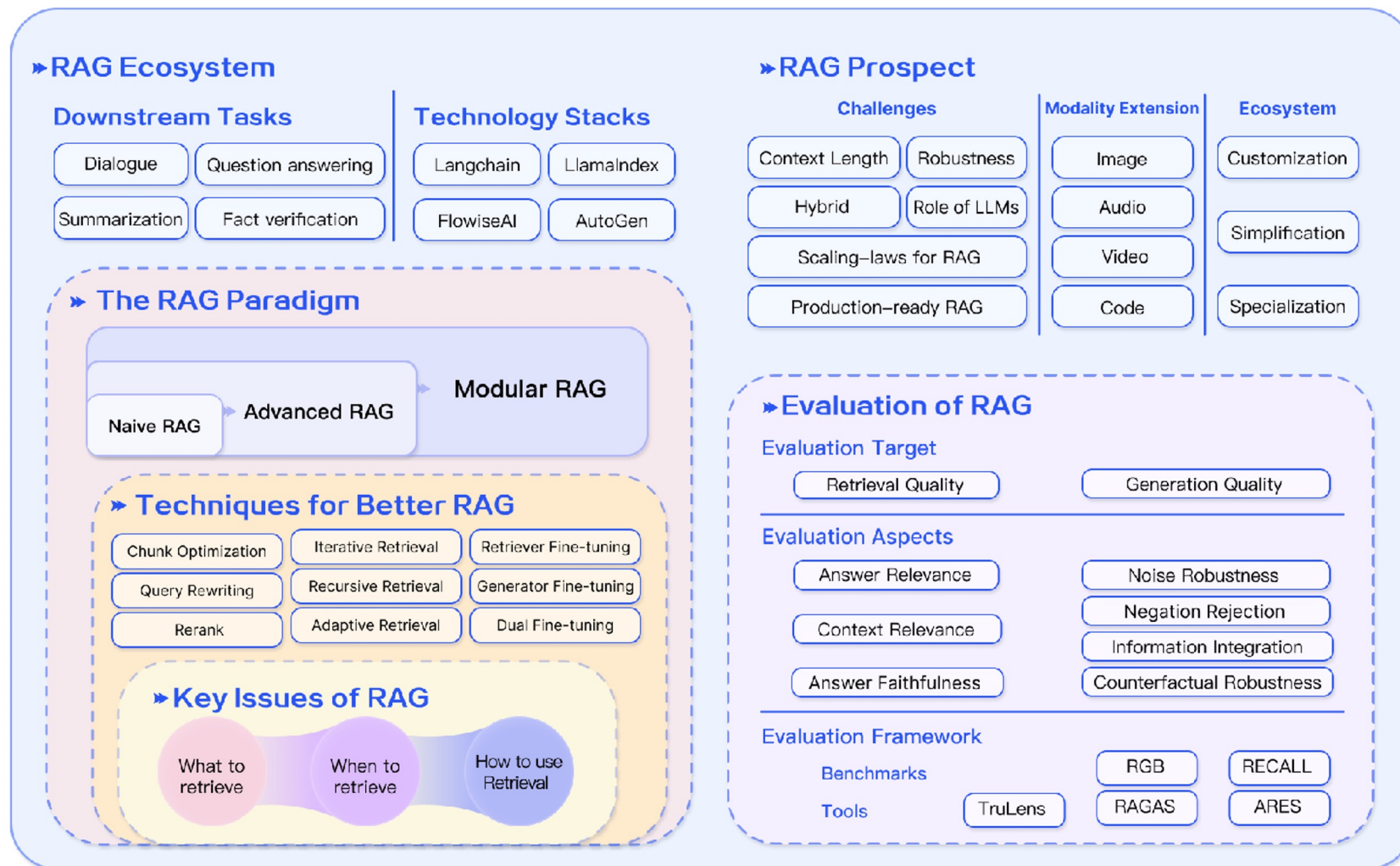


Figure 7: Summary of RAG ecosystem

## Ecosystem of RAG

### RAG's Impact

- **Downstream Tasks:** RAG excels in open-ended question answering, fact verification, and could potentially benefit specialized fields like medicine, law, and education.
- **Evaluation Needs:** We need refined metrics to assess the unique aspects of RAG models, such as contextual relevance and creativity in responses.
- **Interpretability:** Explaining how RAG models arrive at their answers is important for fostering trust and transparency.

### The Technical Stack

- **Key Tools:** LangChain and LLaIndex are popular for their extensive RAG-related APIs.
- **Specialization:** Emerging tools like Flowise AI (low-code), Haystack, Meltano, and Cohere Coral offer distinct advantages.
- **Traditional Tech:** Companies like Weaviate (Verba) and Amazon (Kendra) are expanding their offerings to include RAG services.
- **Trends:** The RAG technical stack is evolving towards customization, simplification, and production-focused specialization.

## Conclusion

### RAG Advancements

- **Combining Strengths:** RAG successfully integrates the power of LLMs (parameterized knowledge) with external knowledge bases (non-parameterized knowledge) for improved results.
- **Evolution:** RAG has progressed through Naive, Advanced, and Modular stages. Advanced RAG, with techniques like query rewriting and chunk reranking, offers better performance and interpretability.
- **Hybrid Approaches:** Combining RAG with fine-tuning and reinforcement learning further enhances its capabilities.

### Future Directions

- **Remaining Challenges:** Improving robustness and handling longer contexts are key areas of ongoing RAG research.
- **Expanding Domains:** RAG is being adapted for multimodal tasks, handling images, videos, and code. This has significant practical implications for real-world AI deployments.
- **The RAG Ecosystem:** A growing ecosystem of tools and applications supports RAG development.
- **Evaluation Needs:** As RAG evolves, we need refined evaluation methods to accurately measure its performance and progress.