FLORIDA ATLANTIC

COT 6930 - Generative Intelligence
and Software Development Lifecycle

Topic 4 -
Retrieval-Augmented Generation

Dr. Fernando Koch
kochf@fau.edu
http://www.fernandokoch.me

Agenda

Fundamentals of RAG

Variations of RAG Structures

Use Cases

Exercises

Our Key Question

How can we combine retrieval mechanisms with generative models to produce grounded, accurate, and contextually relevant outputs?

Evolution of Automated Decision Mechanisms

1950s-

Computation Models

If-Then-Else
Finite State Machines
Hardcode Rules

1970s-

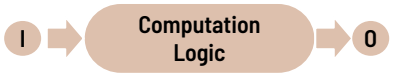
Retrieval Models

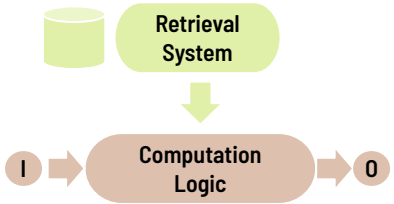
Databases
Rule-Base Systems
Decision Trees

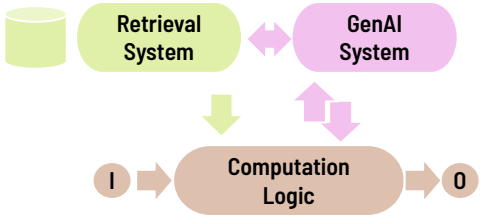
2010-

Generative Models

Large Language Models
Generative Adversarial Networks
Diffusion Models



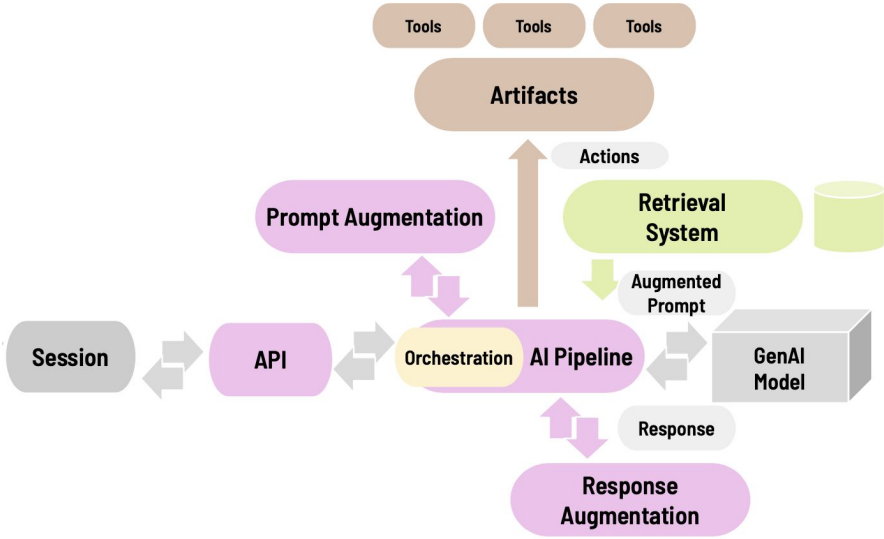




Generative Intelligence System

End-to-end systems with autonomous content generation

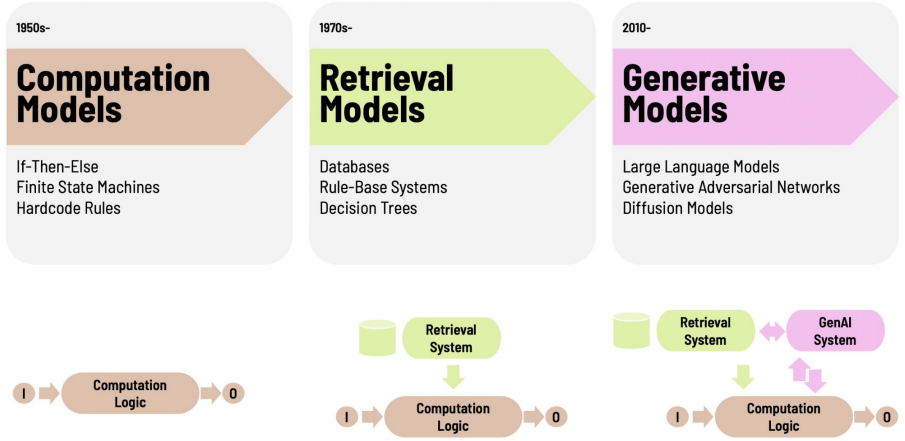
- Session-aware workflows.
- Automated prompt engineering.
- Context-aware content generation
- Adaptive behaviour combining orchestration, memory, retrieval augmentation, others.
- End-to-end automation for LGM consumption.



Fundamentals of RAG

Why do we need Retrieval-Augmented Generation?

- **Brings together precision of retrieval with creativity of generation.**
- RAG provides scalable access to external sources beyond model size.
- LLMs cannot hold entire knowledge bases in memory.
- Ensures outputs are grounded in current, task-specific knowledge.
- Reduces hallucinations and improves domain relevance.



Lewis et al, 2020

https://arxiv.org/abs/2005.11401

FLORIDA ATLANTIC

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis[†‡], **Ethan Perez**[⋆],

Aleksandra Piktus[†], **Fabio Petroni**[†], **Vladimir Karpukhin**[†], **Naman Goyal**[†], **Heinrich Küttler**[†],

Mike Lewis[†], **Wen-tau Yih**[†], **Tim Rocktäschel**[†‡], **Sebastian Riedel**[†‡], **Douwe Kiela**[†]

[†]Facebook AI Research; [‡]University College London; [⋆]New York University;
plewis@fb.com

Abstract

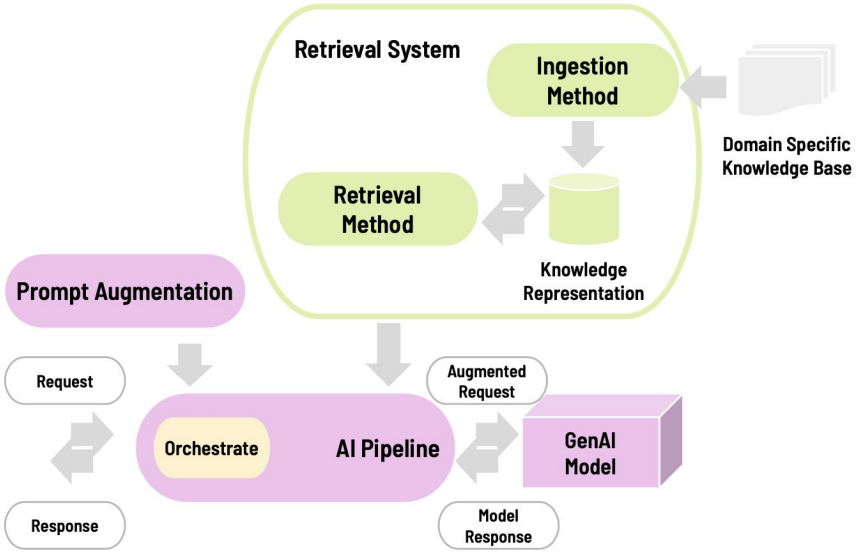
Large pre-trained language models have been shown to store factual knowledge in their parameters, and achieve state-of-the-art results when fine-tuned on downstream NLP tasks. However, their ability to access and precisely manipulate knowledge is still limited, and hence on knowledge-intensive tasks, their performance lags behind task-specific architectures. Additionally, providing provenance for their decisions and updating their world knowledge remain open research problems. Pre-trained models with a differentiable access mechanism to explicit non-parametric memory have so far been only investigated for extractive downstream tasks. We explore a general-purpose fine-tuning recipe for retrieval-augmented generation (RAG) — models which combine pre-trained parametric and non-parametric memory for language generation. We introduce RAG models where the parametric memory is a pre-trained seq2seq model and the non-parametric memory is a dense vector index of Wikipedia, accessed with a pre-trained neural retriever. We compare two RAG formulations, one which conditions on the same retrieved passages across the whole generated sequence, and another which can use different passages per token. We fine-tune and evaluate our models on a wide range of knowledge-

What is Retrieval-Augmented Generation (RAG)?

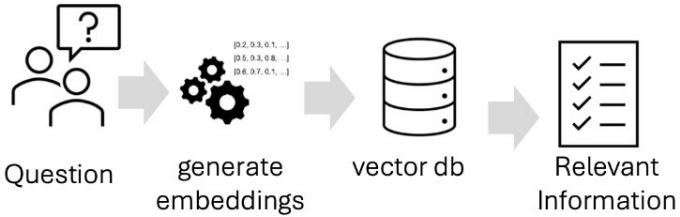
Add Retrieval Systems as part of AI Pipeline to augment prompts with grounded facts from trusted knowledge sources.

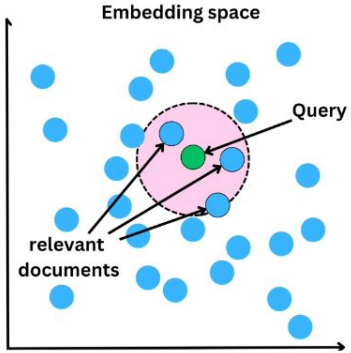
How do we overcome the limits of context windows and training cutoffs, while ensuring answers remain accurate and relevant?

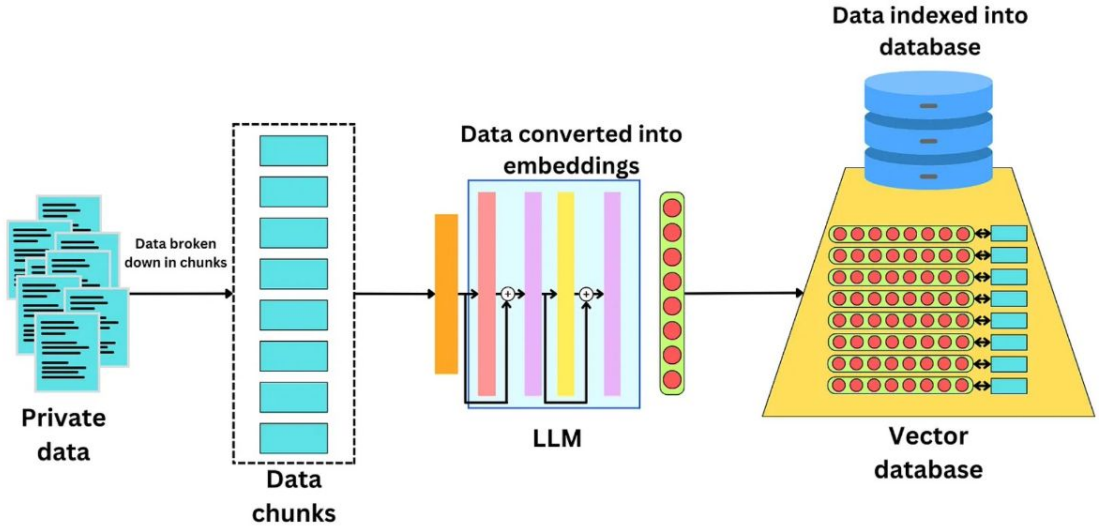
- Combines retrieval systems with generative models in a unified pipeline.
- Dynamically pulls relevant documents from external knowledge bases.
- Grounds generative outputs in context-specific, up-to-date information.



How does RAG operate?

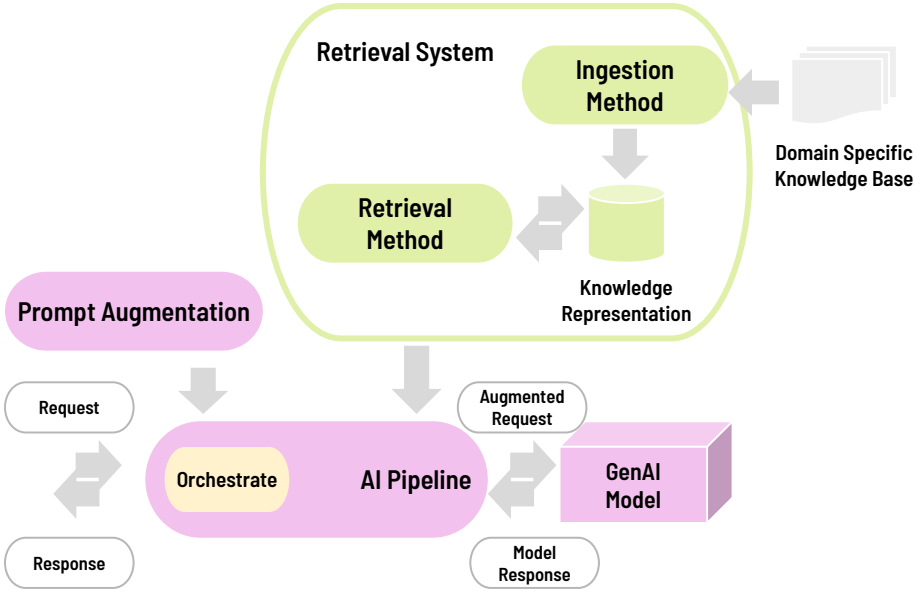






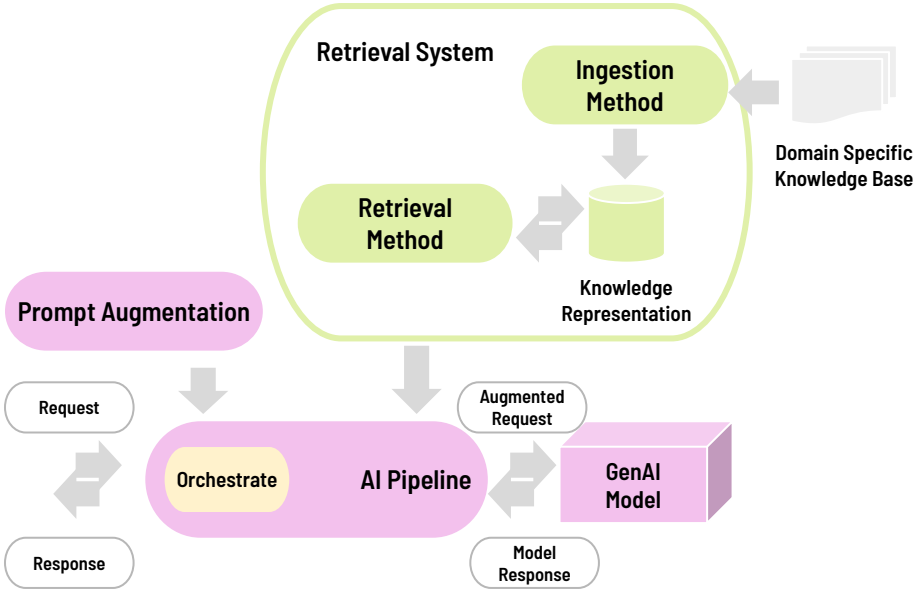
How to implement RAG?

1. **Preprocessing**: chunking, metadata tagging, embeddings.
2. **Retrieval**: query embedding → search → Re-ranking for relevance.
3. **Augmentation**: build a prompt/context window (instructions + top passages + metadata/citations).
4. **Generation**: LLM produces answer; post-process (citation injection, safety filters).



Implementation Decisions

1. **Chunking:** semantic or fixed; window overlap for coherence.
2. **Embeddings:** domain-tuned vs general; dimensionality vs latency trade-off.
3. **Retrieval:** vector only vs hybrid (BM25 + vector); k, filters, and time-decay.
4. **Re-ranking:** cross-encoders (high accuracy) vs LLM re-rank (flexible).



Variations of RAG Structures

Variations of RAG Architectures

Different contexts require different RAG architectures — no single design fits all needs.

How can we balance accuracy, efficiency, adaptability, and cost across diverse applications?

- **One pipeline cannot optimize** for all possibilities simultaneously.
- **Diverse constraints**, e,g latency, context windows, compute, drive alternative designs.
- **Innovation pushes exploration** of ranking, modularity, graph-based, and self-learning methods.

Foundational Approaches

- *Original RAG* – baseline retrieval + generation.
- *LongRAG* – extended context windows.
- *Self-RAG* – self-supervised improvement.

Accuracy-Oriented

- *Corrective RAG* – error correction loop.
- *Golden-Retriever RAG* – precision retrieval.
- *RankRAG* – advanced ranking for relevance.

Efficiency-Focused

- *EfficientRAG* – optimized for latency and cost.
- *Speculative RAG* – proactive retrieval.
- *Multi-Head RAG* – parallelized retrieval pipelines.

Adaptive & Modular

- *Adaptive RAG* – domain/task tuning.
- *Modular RAG* – pluggable components.
- *Graph RAG* – structured knowledge integration.

Category A – Foundational Models

You need to build a system for general knowledge tasks that can scale to long contexts and improve without heavy supervision, so how do you design a versatile baseline?

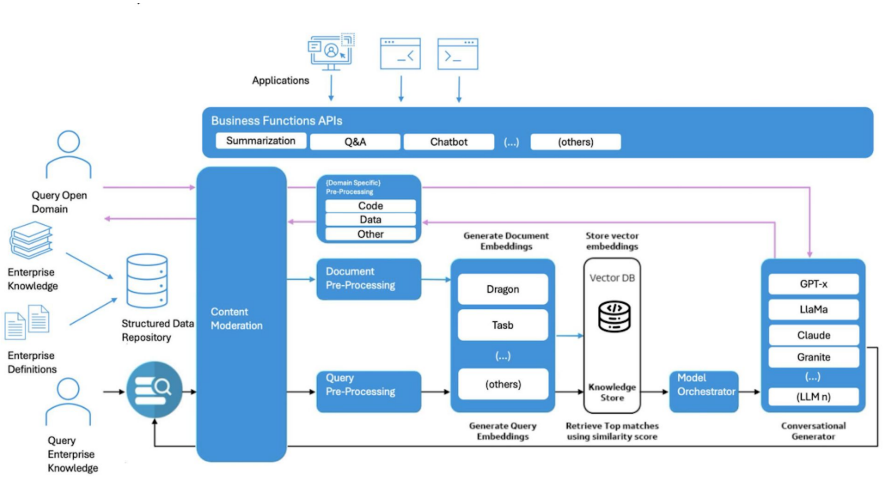
Use Case: Baseline Enterprise RAG

Building a company-wide knowledge assistant that answers FAQs, summarizes reports, and learns from ongoing interactions.

- Provides general-purpose Q&A and summarization.
- Handles extended documents such as reports or policies.
- Learns over time without labeled supervision.

Why this type of RAG is needed?

- Enterprises require a baseline system before specializing.
- Self-learning reduces reliance on manual training updates.

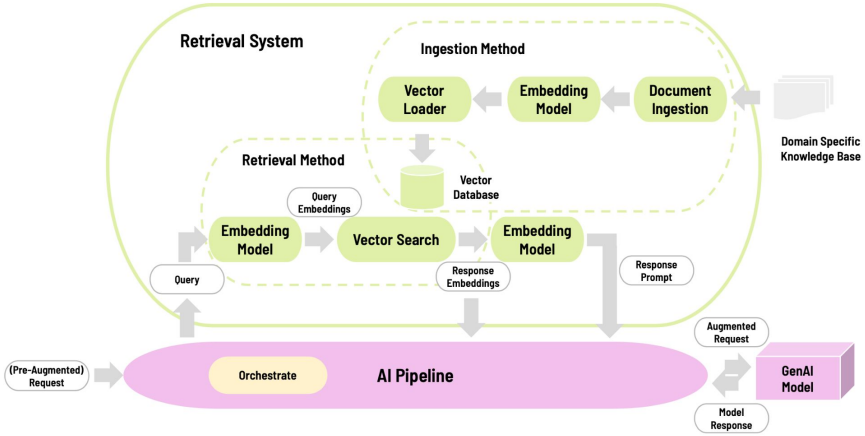


Original RAG

The baseline model that combines retrieval with generation.

You need to build a system for general Q&A and summarization, so how do you combine retrieval and generation in a simple baseline?

- Provides a simple, versatile foundation.
- Works across many Q&A and summarization tasks.
- Serves as the template for specialized extensions.



Lewis et al, 2020

https://arxiv.org/abs/2005.11401

FLORIDA ATLANTIC

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis[†‡], **Ethan Perez**[⋆],

Aleksandra Piktus[†], **Fabio Petroni**[†], **Vladimir Karpukhin**[†], **Naman Goyal**[†], **Heinrich Küttler**[†],

Mike Lewis[†], **Wen-tau Yih**[†], **Tim Rocktäschel**[†‡], **Sebastian Riedel**[†‡], **Douwe Kiela**[†]

[†]Facebook AI Research; [‡]University College London; [⋆]New York University;
plewis@fb.com

Abstract

Large pre-trained language models have been shown to store factual knowledge in their parameters, and achieve state-of-the-art results when fine-tuned on downstream NLP tasks. However, their ability to access and precisely manipulate knowledge is still limited, and hence on knowledge-intensive tasks, their performance lags behind task-specific architectures. Additionally, providing provenance for their decisions and updating their world knowledge remain open research problems. Pre-trained models with a differentiable access mechanism to explicit non-parametric memory have so far been only investigated for extractive downstream tasks. We explore a general-purpose fine-tuning recipe for retrieval-augmented generation (RAG) — models which combine pre-trained parametric and non-parametric memory for language generation. We introduce RAG models where the parametric memory is a pre-trained seq2seq model and the non-parametric memory is a dense vector index of Wikipedia, accessed with a pre-trained neural retriever. We compare two RAG formulations, one which conditions on the same retrieved passages across the whole generated sequence, and another which can use different passages per token. We fine-tune and evaluate our models on a wide range of knowledge-

Original RAG

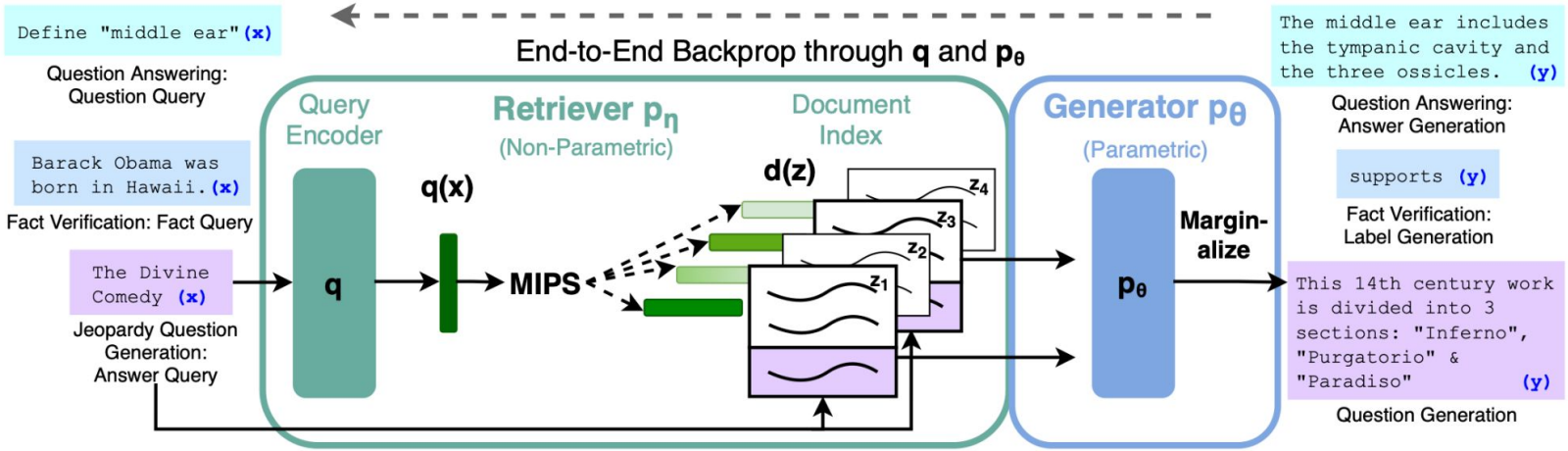


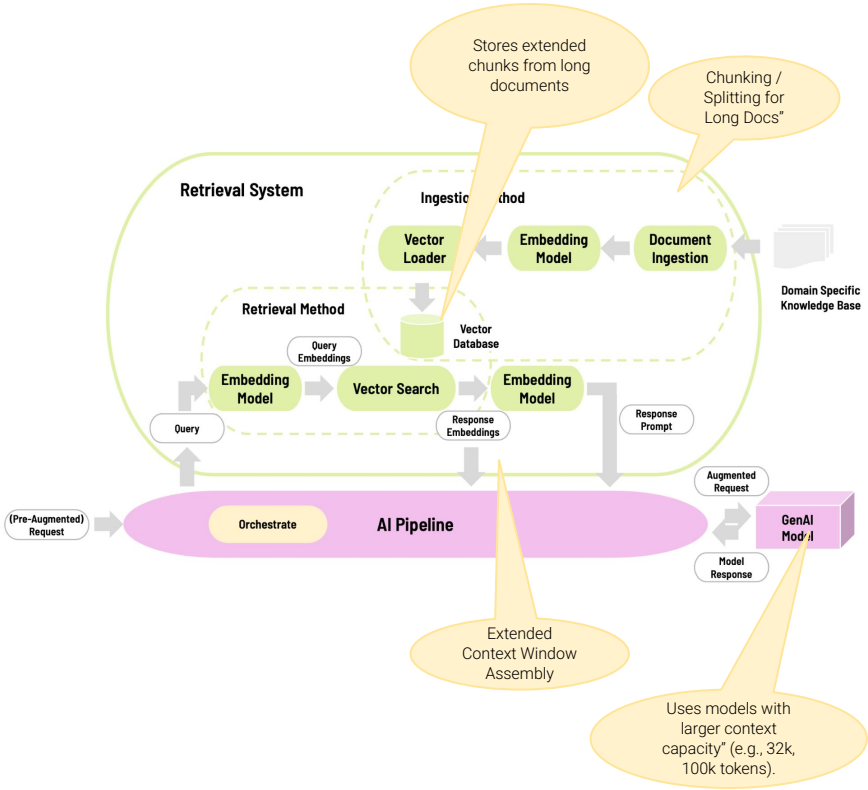
Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

Long RAG

Optimized to handle long documents and extended contexts.

You need to build a system for lengthy legal or scientific documents, so how do you scale retrieval and generation to extended contexts?

- Splits documents into manageable chunks.
- Handles queries spanning thousands of tokens.
- Enables summarization of complex long-form content.

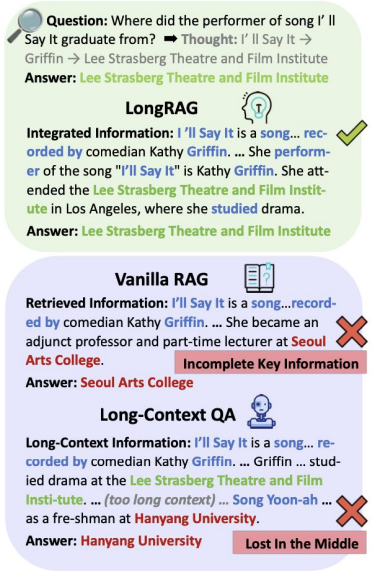


LongRAG: A Dual-Perspective Retrieval-Augmented Generation Paradigm for Long-Context Question Answering

Qingfei Zhao[1,2,†], **Ruobing Wang**[1,2], **Yukuo Cen**[4],
Daren Zha[1], **Shicheng Tan**[3], **Yuxiao Dong**[3], **Jie Tang**[3,*]
[1]Institute of Information Engineering, Chinese Academy of Sciences;
[2]School of Cyber Security, University of Chinese Academy of Sciences;
[3]Tsinghua University; [4]Zhipu AI
{zhaoqingfei, wangruobing, zhadaren}@iie.ac.cn, yukuo.cen@zhipuai.cn
tsctan@foxmail.com, {yuxiaod, jietang}@tsinghua.edu.cn

Abstract

Long-Context Question Answering (LCQA), a challenging task, aims to reason over long-context documents to yield accurate answers to questions. Existing long-context Large Language Models (LLMs) for LCQA often struggle with the "*lost in the middle*" issue. Retrieval-Augmented Generation (RAG) mitigates this issue by providing external factual evidence. However, its chunking strategy disrupts the global long-context information, and its low-quality retrieval in long contexts hinders LLMs from identifying effective factual details due to substantial noise. To this end, we propose LongRAG, a general, dual-perspective, and robust LLM-based RAG system paradigm for LCQA to enhance RAG's understanding of complex long-context knowledge (i.e., global information and factual details). We design LongRAG as a plug-and-play paradigm, facilitating adaptation to various domains and LLMs. Extensive experiments on three multi-hop datasets demonstrate that LongRAG significantly outperforms long-context LLMs (up by 6.94%), advanced RAG (up by 6.16%), and Vanilla RAG (up by 17.25%). Furthermore, we conduct quantitative ablation studies and multi-



Zaoh et al, 2024
https://arxiv.org/abs/2410.18050

LongRAG

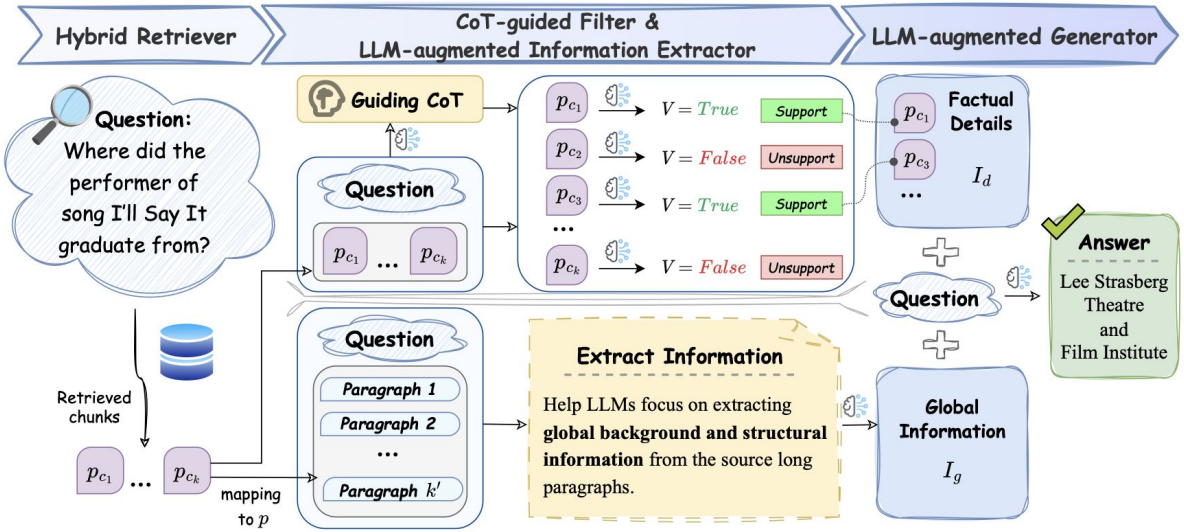


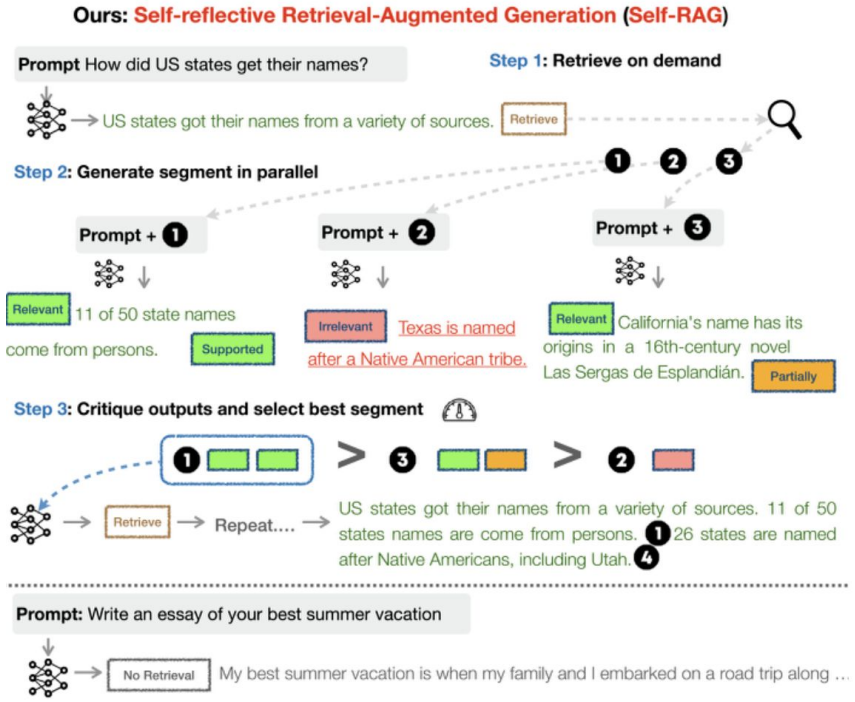
Figure 2: **An overview of LongRAG.** Our system involves four sub-components: Hybrid Retriever receives a question and retrieves the top-k most relevant chunks p_c; CoT-guided Filter generates global key clues to analyze their relevance one by one, obtaining a set of "True" chunks as I_d; Meanwhile, LLM-augmented Information Extractor sequentially maps p_c to the source long-context paragraph p to extract effective global information I_g; LLM-augmented Generator promotes knowledge interaction between I_g and I_d to generate the final answer.

Self-RAG

"Our framework trains a single arbitrary LM that adaptively retrieves passages on-demand (e.g., can retrieve multiple times during generation, or completely skip retrieval), and generates and reflects on retrieved passages and its own generations using special tokens"

You need to build a system that continuously improves itself, so how do you make it learn from its own outputs without supervision?

- Uses self-supervised feedback to critique responses.
- Refines embeddings and retrieval strategies.
- Reduces dependence on annotated training data.

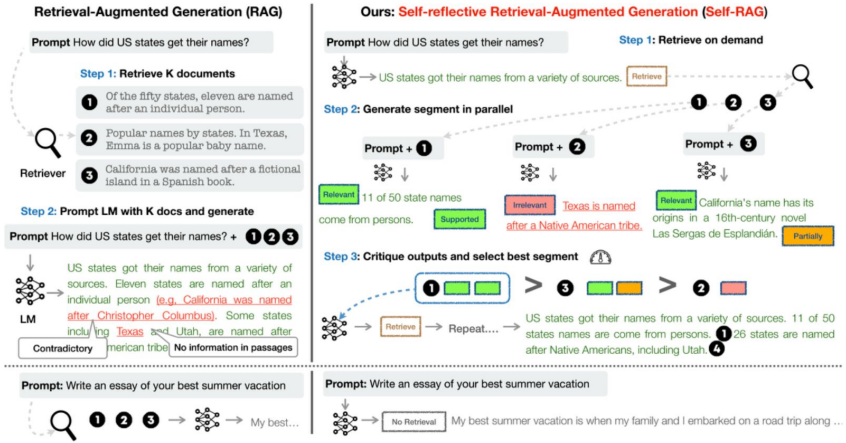


Self-RAG: Learning to Retrieve, Generate and Critique through Self-Reflections

Akari Asai[1] Zeqiu Wu[1] Yizhong Wang[1], Avirup Sil[2], Hannaneh Hajishirzi[1,3],
[1]University of Washington, [2]IBM AI Research, [3]Allen Institute for AI

arXiv | Code | Model (7B) | Model (13B) | Data | HF Space

SELF-RAG learns to retrieve, generate and critique to enhance LM's output quality and factuality, outperforming ChatGPT and retrieval-augmented LLama2 Chat on six tasks.



https://selfrag.github.io

Summary

The issue: Factual inaccuracies of versatile LLMs
Despite their remarkable capabilities, large language models (LLMs) often produce responses containing factual inaccuracies due to their sole reliance on the parametric knowledge they encapsulate. They often generate hallucinations, especially in long-tail, their knowledge gets obsolete, and lacks attribution

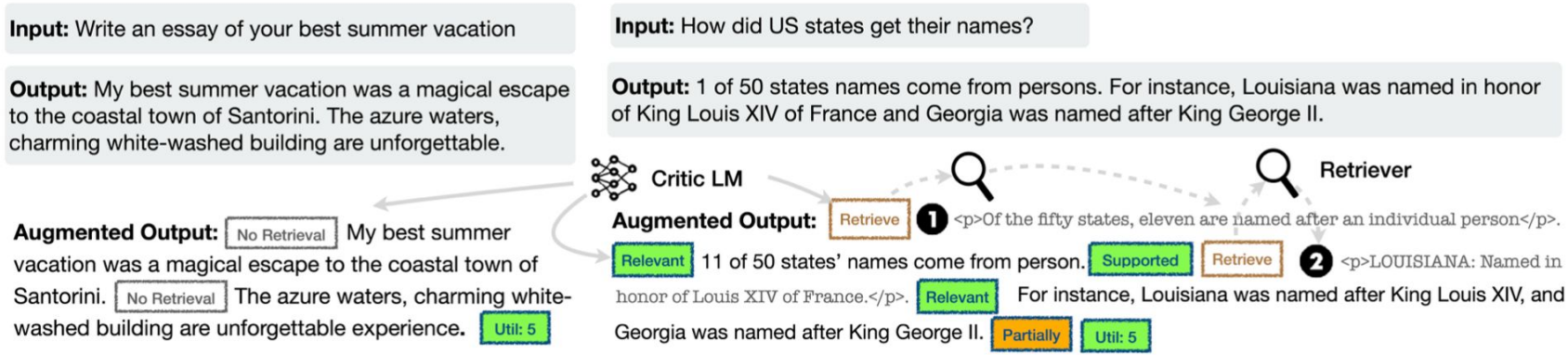
FLORIDA ATLANTIC

Training of Self-RAG

Self-RAG training consists of three models, a Retriever, a Critic and a Generator.

1. Trains the **Critic** and augment diverse instruction-output data with retrieved passages by the **Retriever** as well as reflection tokens (See Figure below).

2. Train the **Generator** LM using a standard next token prediction objective to learns to generate natural continuations as well as special tokens to retrieve or critique its own generations.



Category B – Accuracy-Focused Models

*You need to build a system for domains where **errors are costly**, so how do you guarantee precise retrieval and minimize mistakes?*

Use Case: Accurate Medical Assistant

Deploying a medical assistant that references clinical guidelines and avoids hallucinating treatments.

- Retrieves authoritative medical documents and guidelines.
- Provides evidence-grounded, traceable responses.
- Corrects errors through feedback or validation loops.

Why this type of RAG is needed?

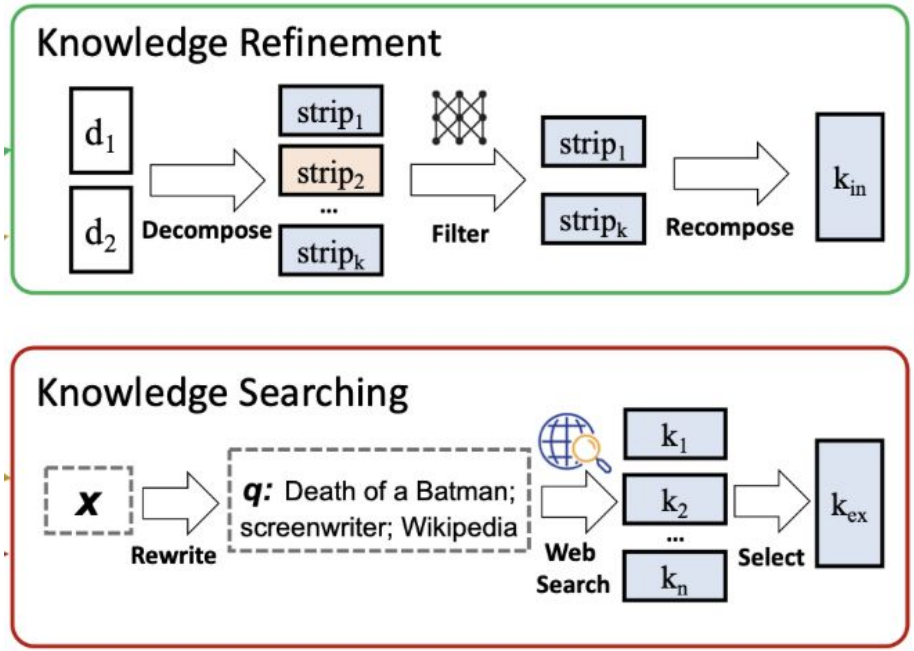
- Clinical tasks demand verifiable, high-precision outputs.
- Compliance requires provenance and auditability.
- A general-purpose baseline RAG lacks sufficient accuracy.

Corrective RAG

Adds correction loops to improve accuracy.

You need to build a system for medicine or law, so how do you ensure it corrects mistakes and avoids propagating errors?

- Introduces a correction loop that re-checks outputs.
- Feedback improves retrieval and generation quality.
- Increases trust in high-stakes environments.



Yan et al, 2024

https://arxiv.org/abs/2401.15884

Corrective Retrieval Augmented Generation

Shi-Qi Yan[1*], **Jia-Chen Gu**[2*], **Yun Zhu**[3], **Zhen-Hua Ling**[1]
[1]National Engineering Research Center of Speech and Language Information Processing, University of Science and Technology of China, Hefei, China
[2]Department of Computer Science, University of California, Los Angeles
[3]Google DeepMind
yansiki@mail.ustc.edu.cn, gujc@ucla.edu, yunzhu@google.com, zhling@ustc.edu.cn

Abstract

Large language models (LLMs) inevitably exhibit hallucinations since the accuracy of generated texts cannot be secured solely by the parametric knowledge they encapsulate. Although retrieval-augmented generation (RAG) is a practicable complement to LLMs, it relies heavily on the relevance of retrieved documents, raising concerns about how the model behaves if retrieval goes wrong. To this end, we propose the **C**orrective **R**etrieval **A**ugmented **G**eneration (CRAG) to improve the robustness of generation. Specifically, a lightweight retrieval evaluator is designed to assess the overall quality of retrieved documents for a query, returning a confidence degree based on which different knowledge retrieval actions can be triggered. Since retrieval from static and limited corpora can only return sub-optimal documents, large-scale web searches are utilized as an extension for augmenting the retrieval results. Besides, a decompose-then-recompose algorithm is designed for retrieved documents to selectively focus on key information and filter out irrelevant information in

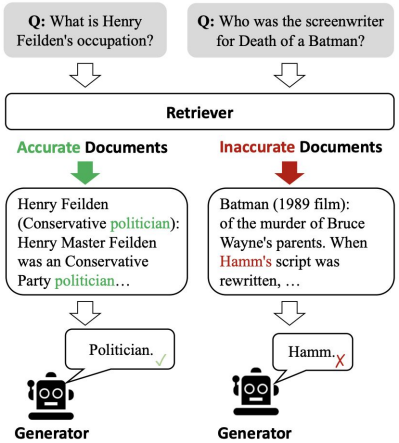
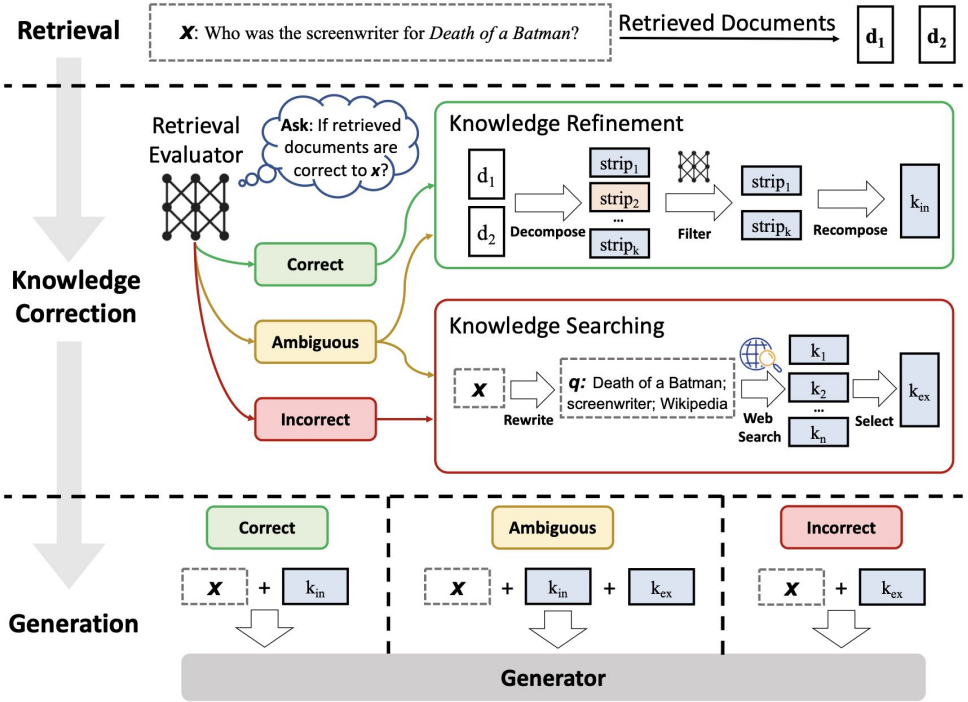


Figure 1: The examples show that a low-quality retriever is prone to introducing a substantial amount of irrelevant information, impeding the generators from acquiring accurate knowledge and potentially misleading them.

Corrective RAG

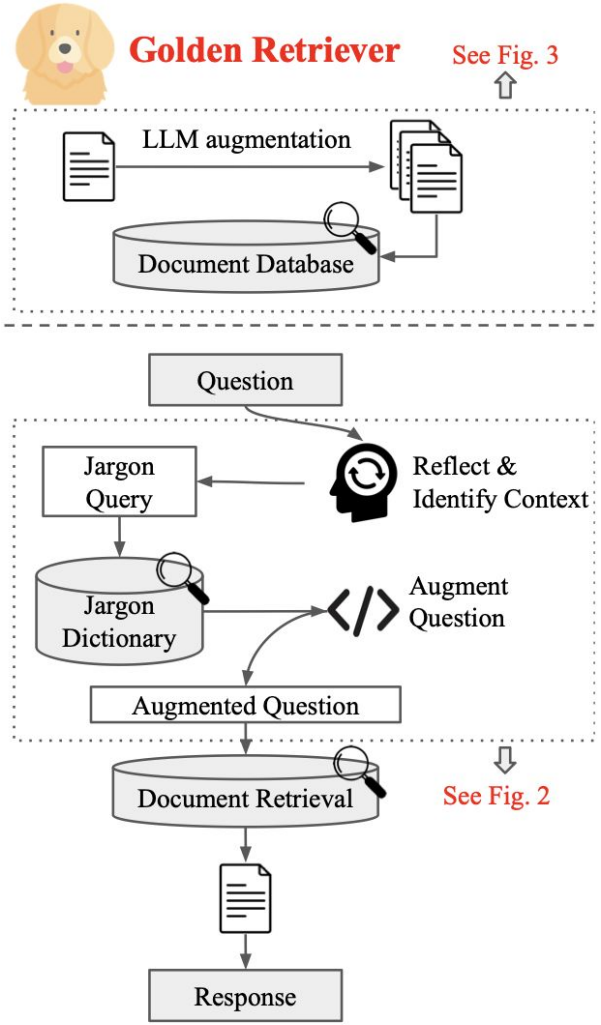


Golden-Retriever RAG

Designed for maximum retrieval precision.

You need to build a fact-checking system, so how do you guarantee that only highly reliable sources are retrieved?

- Applies stricter retrieval filters and thresholds.
- Sacrifices breadth for reliability.
- Ensures compliance and auditing accuracy.



An et al, 2024

https://arxiv.org/abs/2408.00798

FLORIDA ATLANTIC

Golden-Retriever: High-Fidelity Agentic Retrieval Augmented Generation for Industrial Knowledge Base

Zhiyu An[σμ1] **Xianzhong Ding**[γ] **Yen-Chun Fu**[μ] **Cheng-Chung Chu**[μ] **Yan Li**[μ] **Wan Du**[σ]
σ : University of California, Merced, CA, USA
μ : Western Digital Corporation, CA, USA
γ : Lawrence Berkeley National Laboratory, CA, USA
{zan7, wdu3}@ucmerced.edu

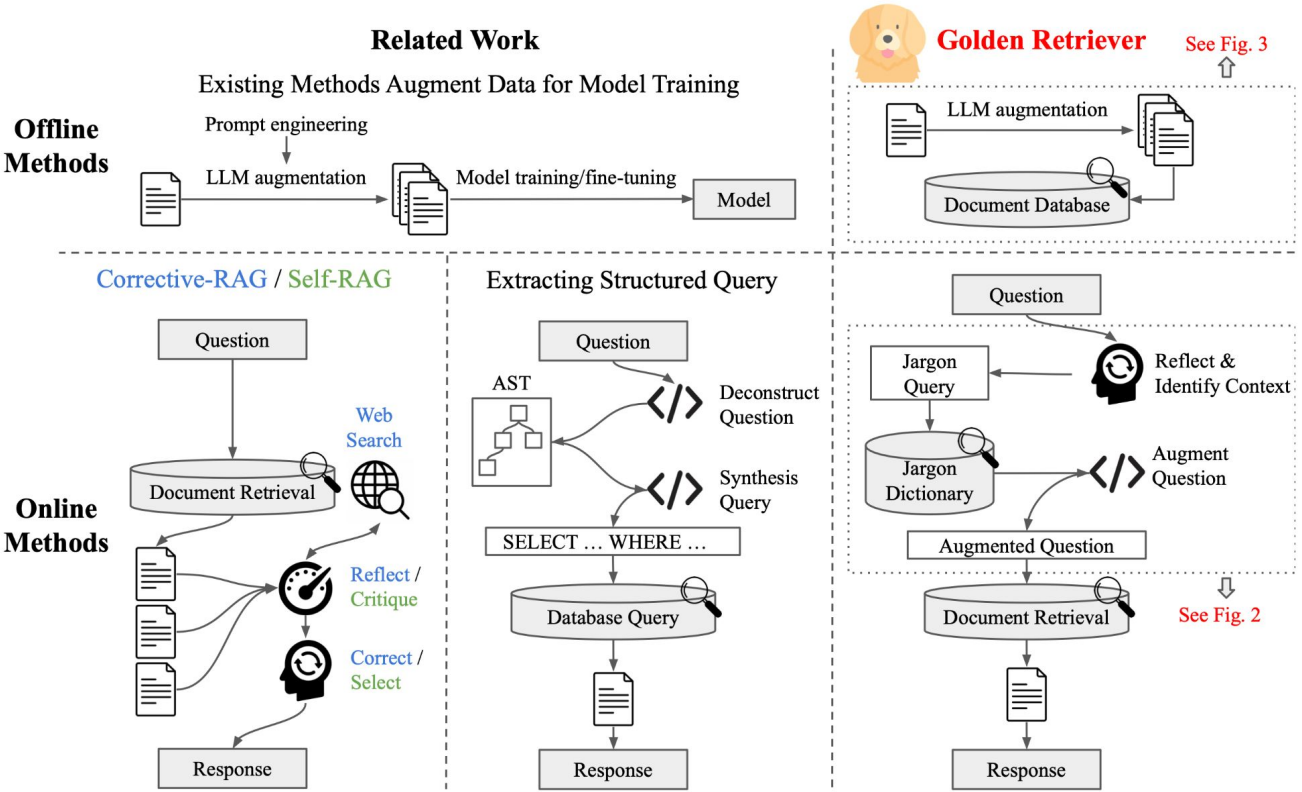
Abstract

This paper introduces Golden-Retriever, designed to efficiently navigate vast industrial knowledge bases, overcoming challenges in traditional LLM fine-tuning and RAG frameworks with domain-specific jargon and context interpretation. Golden-Retriever incorporates a reflection-based question augmentation step before document retrieval, which involves identifying jargon, clarifying its meaning based on context, and augmenting the question accordingly. Specifically, our method extracts and lists all jargon and abbreviations in the input question, determines the context against a predefined list, and queries a jargon dictionary for extended definitions and descriptions. This comprehensive augmentation ensures the RAG framework retrieves the most relevant documents by providing clear context and resolving ambiguities, significantly improving retrieval accuracy. Evaluations using three open-source LLMs on a domain-specific question-answer dataset demonstrate Golden-Retriever's superior performance, providing a robust solution for efficiently integrating and querying industrial knowledge bases.

(Petroni et al., 2019; Hu et al., 2021). To make a pre-trained LLM incorporate a company's domain-specific knowledge, we may fine-tune it over the company's proprietary documents. However, fine-tuning is computationally expensive, generalize poorly to new knowledge due to the Reversal Curse (Berglund et al., 2023), and limited in capacity, as it may overwrite old knowledge (Roberts et al., 2020; Zhai et al., 2024).

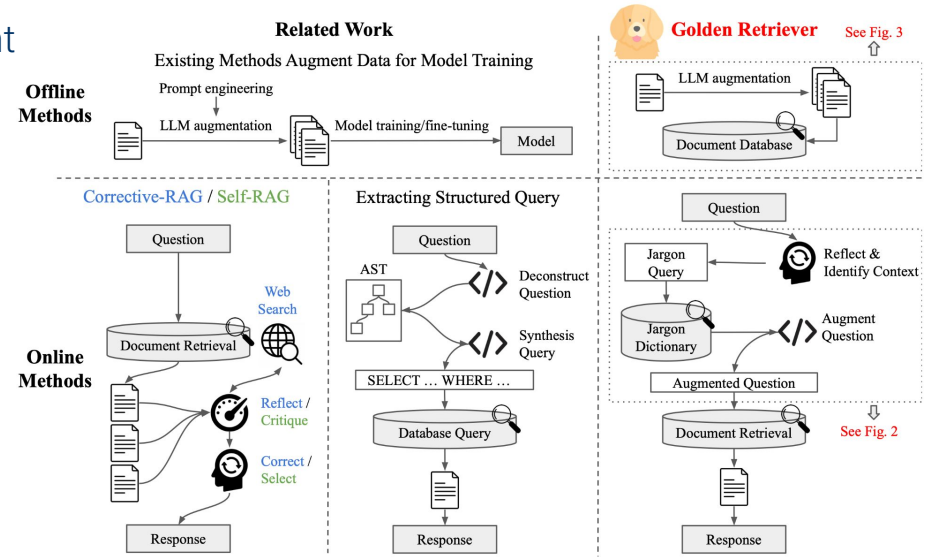
Retrieval Augmented Generation (RAG) (Lewis et al., 2020) offers a flexible and scalable approach for utilizing large document collections. RAG consists of an embedding model, a document database, and a LLM. During offline preparation, RAG embeds document chunks into the document database that retains semantic information. When an user asks a question, RAG first retrieves relevant document chunks according to semantic similarity. Then, the retrieved chunks are incorporated into prompts for the LLM, which then generates an answer. The output of RAG is the answer generated by the LLM based on the document chunks. This allows dynamic updates of knowledge base for an LLM without retraining it.

Golden Retriever RAG

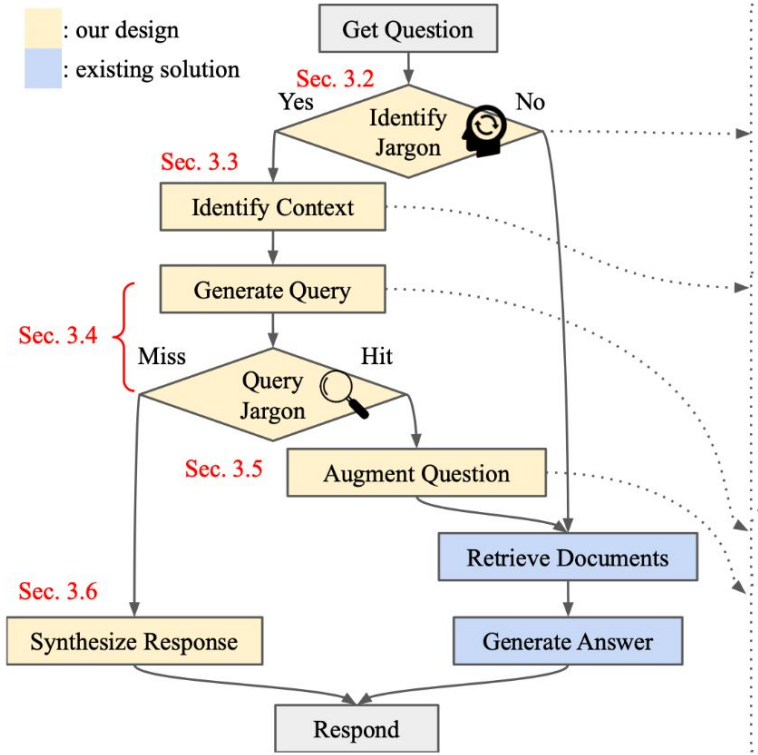


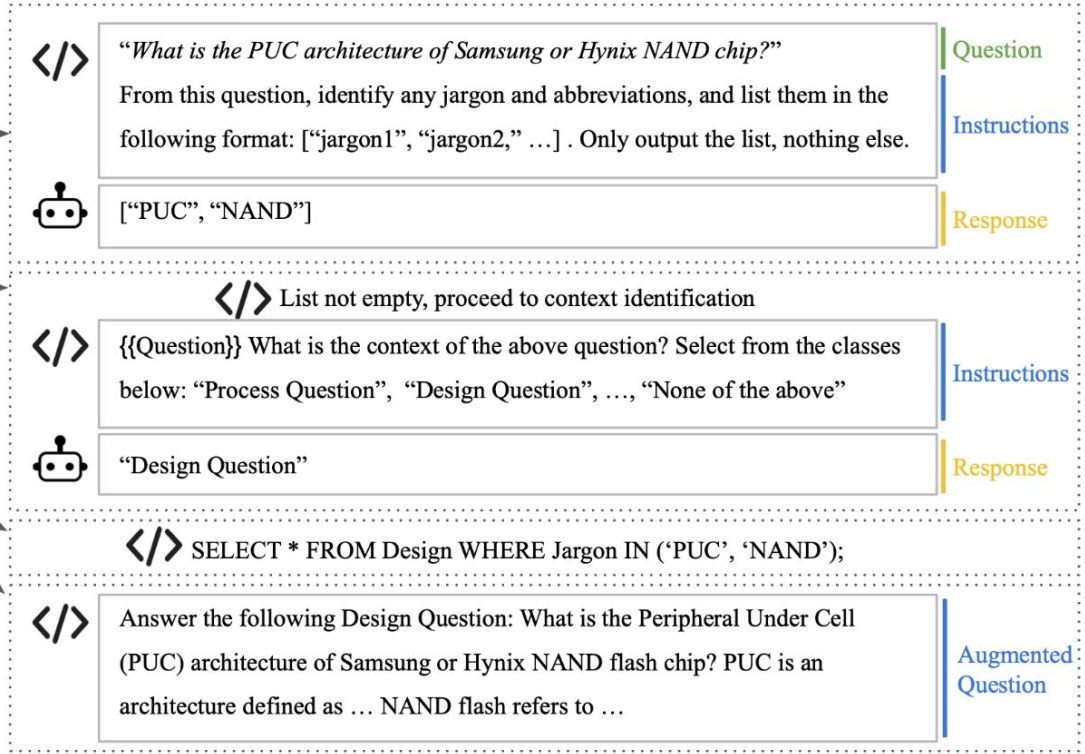
How does Golden Retriever RAG work?

- Offline part focuses on enhancing the document database to improve the relevance of retrieved documents
- the online process involves identifying jargons and abbreviations within the user's question.
- determine the context in which the question is asked, s the meaning of terms can vary significantly across different contexts.



Golden Retriever RAG



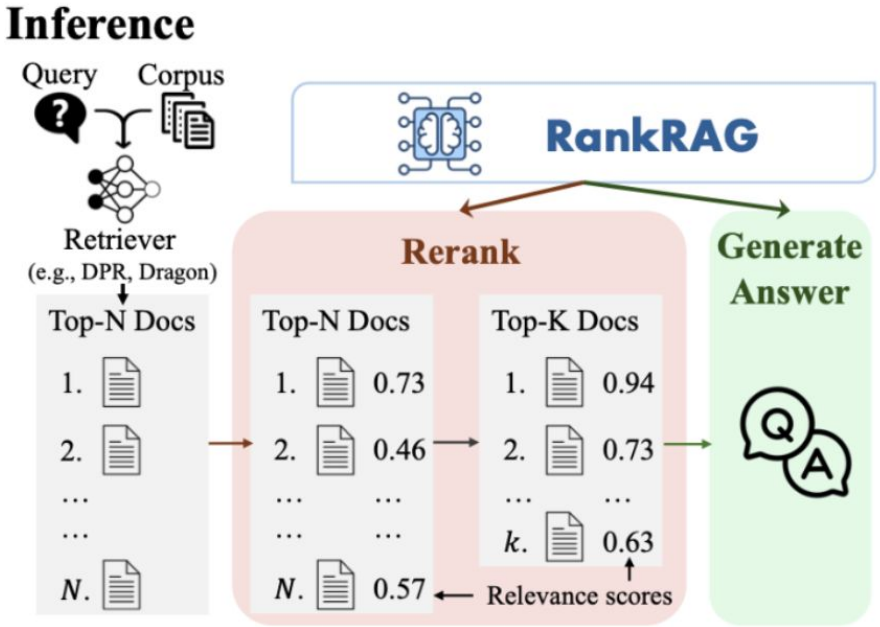


RankRAG

"We instruction-tune the LLM to simultaneously capture the relevance between the question and context and utilize the retrieved context for answer generation."

You need to build a search or recommendation system, so how do you make sure the most relevant results appear first?

- Uses ranking models to refine retrieved documents.
- Prioritizes the most relevant passages.
- Improves grounding quality and precision.



RankRAG

Yu et al, 2024
https://arxiv.org/abs/2407.02485

FLORIDA ATLANTIC

RankRAG: Unifying Context Ranking with Retrieval-Augmented Generation in LLMs

Yue Yu * Georgia Tech
Wei Ping * NVIDIA
Zihan Liu NVIDIA
Boxin Wang NVIDIA
Jiaxuan You NVIDIA
Chao Zhang Georgia Tech
Mohammad Shoeybi NVIDIA
Bryan Catanzaro NVIDIA

Abstract

Large language models (LLMs) typically utilize the top-k contexts from a retriever in retrieval-augmented generation (RAG). In this work, we propose a novel instruction fine-tuning framework RankRAG, which instruction-tunes a single LLM for the dual purpose of context ranking and answer generation in RAG. In particular, the instruction-tuned LLMs work surprisingly well by adding a small fraction of ranking data into the training blend, and outperform existing expert ranking models, including the same LLM exclusively fine-tuned on a large amount of ranking data. For generation, we compare our model with many strong baselines, including GPT-4-0613, GPT-4-turbo-2024-0409, and ChatQA-1.5, an open-sourced model with the state-of-the-art performance on RAG benchmarks. Specifically, our Llama3-RankRAG significantly outperforms Llama3-ChatQA-1.5 and GPT-4 models on nine knowledge-intensive benchmarks. In addition, it also performs comparably to GPT-4 on five RAG benchmarks in the biomedical domain without instruction fine-tuning on biomedical data, demonstrating its superb capability for generalization to new domains.

1 Introduction

Retrieval-augmented generation (RAG) (Lewis et al., 2020; Izacard & Grave, 2021; Lin et al., 2024; Wang et al., 2024) is a widely used technique for customizing large language models (LLMs) to handle long-tail knowledge (Mallen et al., 2023; Asai et al., 2024b), provide up-to-date information (Kasai et al., 2023), and adapt to specific domains and tasks (Xiong et al., 2024) without modifying the model weights. In general, a dense embedding based retriever (Karpukhin et al., 2020; Lin et al., 2023; Wang et al., 2022) first retrieves top-k chunked contexts from a collection documents or external database for a given question. Then, LLM reads the top-k contexts to generate the answer.

RankRAG

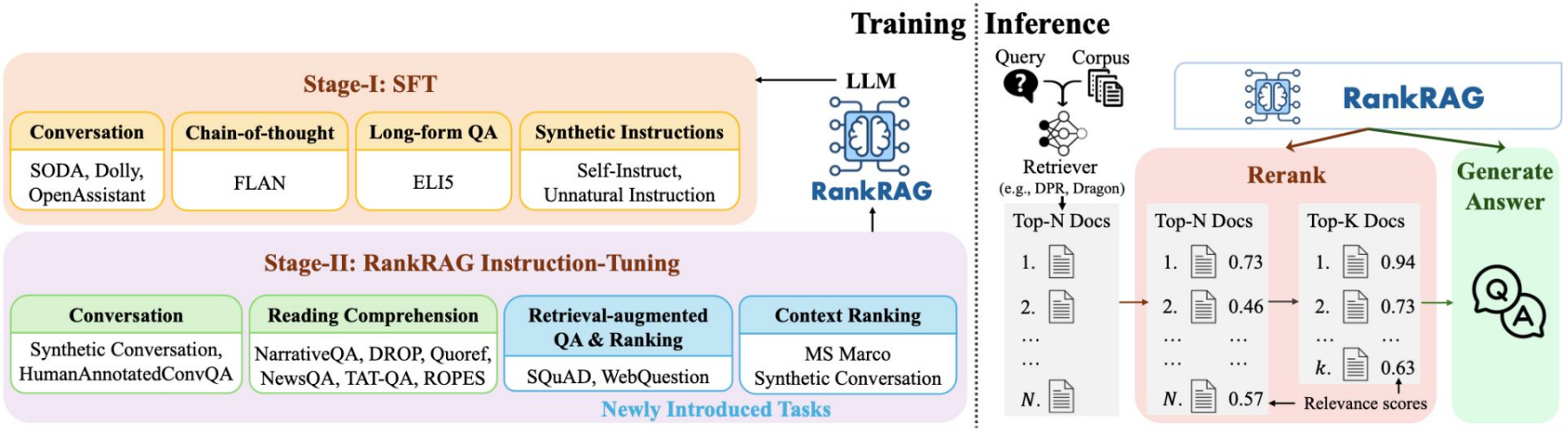


Figure 2: Two-stage instruction tuning framework for RankRAG.

Category C – Efficiency-Focused Models

*You need to build a system for **real-time applications with high throughput**, so how do you optimize for speed, scalability, and low cost*

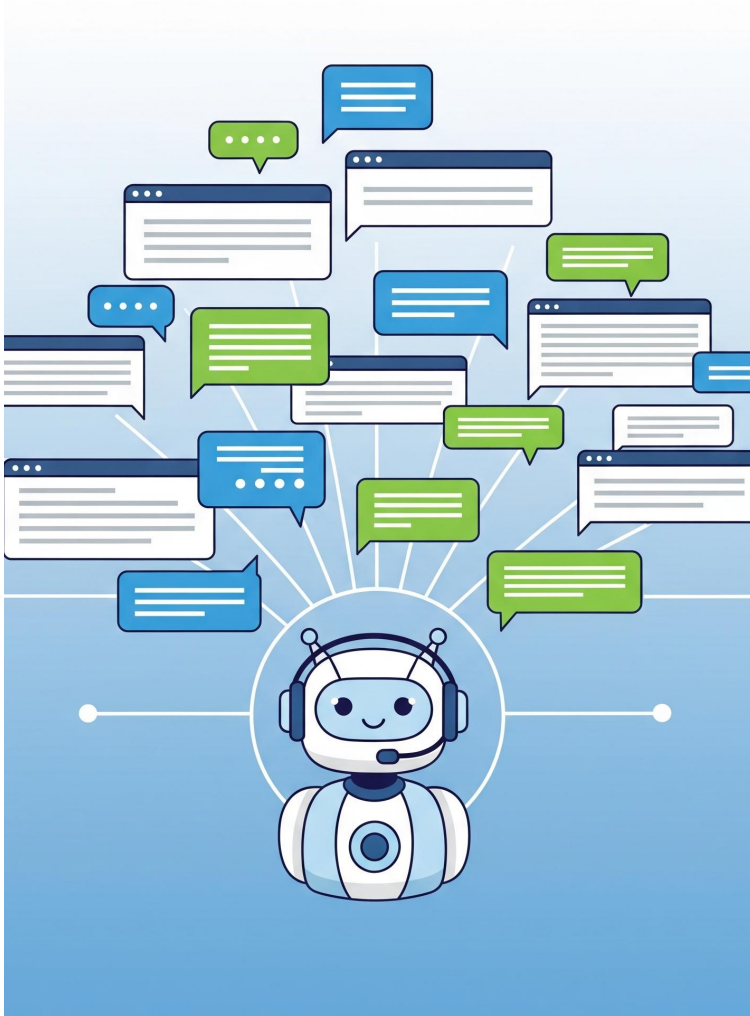
Use Case: Customer Support Chatbot

Running a customer support chatbot that answers instantly while serving thousands of concurrent users.

- Responds to queries in real time with consistent accuracy.
- Handles large-scale, simultaneous interactions reliably.
- Reduces human agent workload by automating routine questions.

Why this type of RAG is needed?

- Customer service requires fast, always-available responses.
- High-volume environments demand scalable, efficient systems.
- A general-purpose baseline RAG cannot support concurrency at enterprise scale.

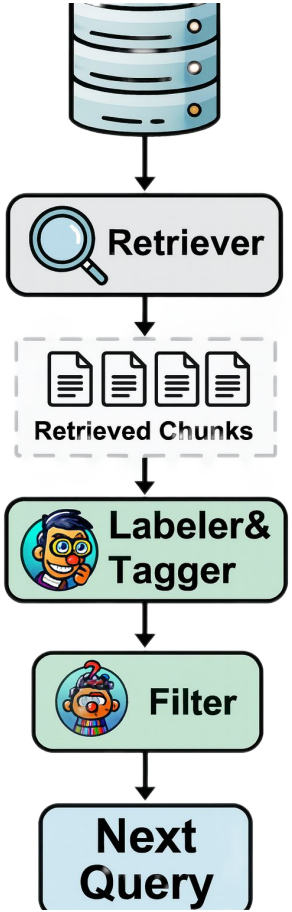


EfficientRAG

Streamlined for low latency and cost efficiency.

You need to build a real-time chatbot, so how do you support instant responses while controlling computational cost??

- Optimizes retrieval indexing for speed.
- Balances accuracy with response latency.
- Designed for high-throughput, interactive systems.



EfficientRAG

Zhuang et al, 2024
https://arxiv.org/abs/2408.04259

FLORIDA ATLANTIC

EfficientRAG: Efficient Retriever for Multi-Hop Question Answering

Ziyuan Zhuang[1*], Zhiyang Zhang[1*], Sitao Cheng[1], Fangkai Yang[2‡], Jia Liu[1], Shujian Huang[1], Qingwei Lin[2], Saravan Rajmohan[2], Dongmei Zhang[2], Qi Zhang[2]
[1] State Key Laboratory for Novel Software Technology, Nanjing University [2] Microsoft
ziyuan.zhuang@smail.nju.edu.cn

Abstract

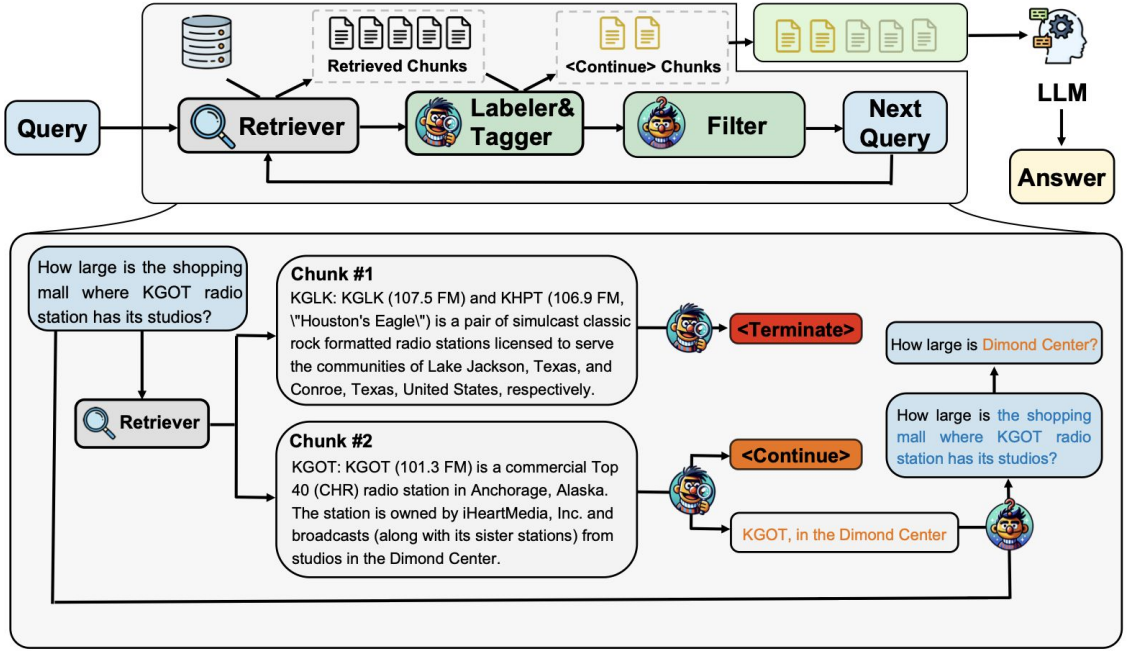
Retrieval-augmented generation (RAG) methods encounter difficulties when addressing complex questions like multi-hop queries. While iterative retrieval methods improve performance by gathering additional information, current approaches often rely on multiple calls of large language models (LLMs). In this paper, we introduce EfficientRAG, an efficient retriever for multi-hop question answering. EfficientRAG iteratively generates new queries without the need for LLM calls at each iteration and filters out irrelevant information. Experimental results demonstrate that EfficientRAG surpasses existing RAG methods on three open-domain multi-hop question-answering datasets. The code is available in aka.ms/efficientrag.

1 Introduction

Large-language models (LLMs) have shown remarkable performance in numerous applications and tasks (OpenAI, 2023; Jiang et al., 2023a; Touvron et al., 2023b). However, LLMs lack knowledge underrepresented in their training data, especially in domain-specific settings, and still face *of motion?"*. However, one-round RAG methods could fail in complex questions where more information is required beyond the first-round retrieved information, *e.g.*, multi-hop questions (Yang et al., 2018a; Trivedi et al., 2022a; Ho et al., 2020b). In order to deal with complex multi-hop questions, recent works propose to obtain required information through multi-round retrievals or reasonings, such as rewriting or generating queries for the following multi-round retrievals (Khattab et al., 2022; Ma et al., 2023; Shao et al., 2023; Jiang et al., 2023b), interleaving multiple retrieval and reasoning steps (Trivedi et al., 2023), multi-rounds of self-asking (Press et al., 2023). However, such iterative retrieval approaches have the following limitations: (1) they require multiple LLM calls concerning rewriting or generating new queries for the next round of retrieval, thus increasing the latency and cost. (2) they require dedicated prompting and few-shot examples that might need updating across different scenarios.

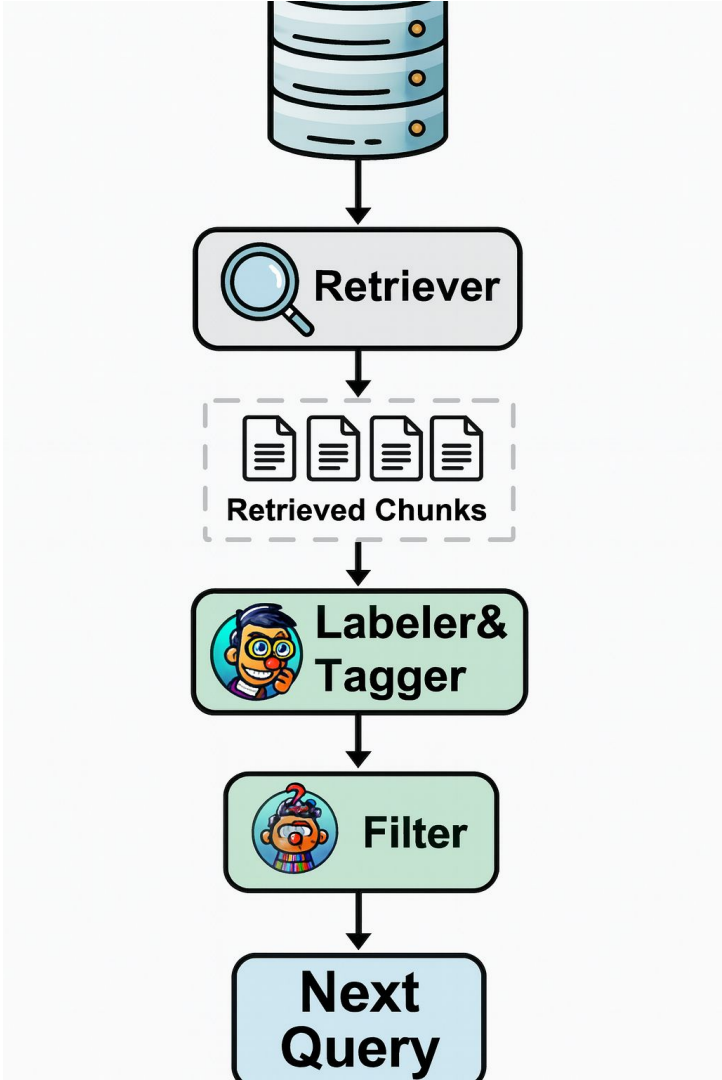
In this paper, we are inspired by the intuition that the types of relations in multi-hop questions

EfficientRAG



How does Efficient RAG work?

1. Retrieving relevant chunks from the knowledge base.
2. Each chunk is tagged as either <Terminate> or <Continue>.
3. From the <Continue> chunks, preserved tokens are annotated (e.g., “KGOT in the Dimond Center”).
4. The Filter processes a combination of:
   a. the original question, and
   b. the annotated tokens from the preserved chunks.
5. This iterative process repeats until:
   a. all chunks are tagged <Terminate>, or
   b. the maximum number of iterations is reached.

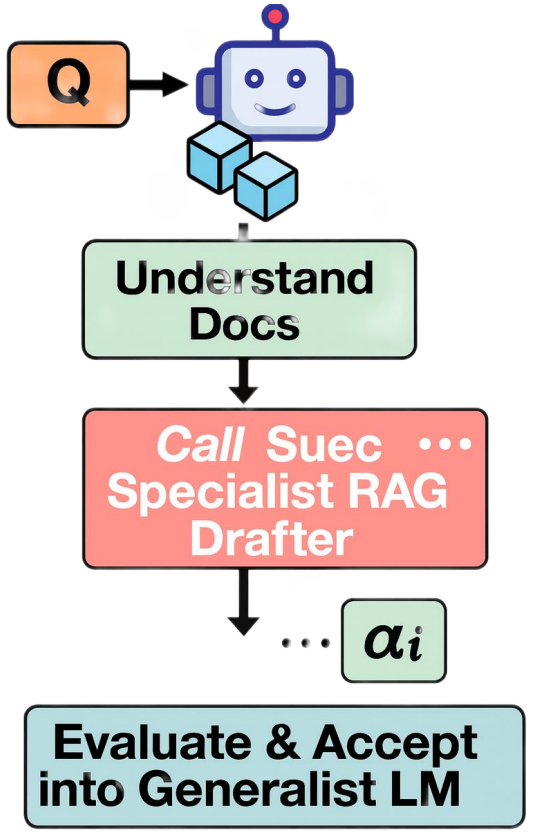


Speculative RAG

Retrieves knowledge proactively to anticipate needs.

You need to build a predictive assistant, so how do you prefetch relevant knowledge before the user finishes asking?

- Anticipates likely queries and retrieves in advance.
- Reduces wait time with proactive retrieval.
- Ideal for predictive typing or code completion.



Speculative RAG

Wang et al, 2024
https://arxiv.org/abs/2407.08223

FLORIDA ATLANTIC

SPECULATIVE RAG: ENHANCING RETRIEVAL AUGMENTED GENERATION THROUGH DRAFTING

Zilong Wang[1*] **Zifeng Wang**[2] **Long T. Le**[2] **Huaixiu Steven Zheng**[3]
Swaroop Mishra[3] **Vincent Perot**[3] **Yuwei Zhang**[1] **Anush Mattapalli**[4]
Ankur Taly[4] **Jingbo Shang**[1] **Chen-Yu Lee**[2] **Tomas Pfister**[2]
[1]University of California, San Diego [2]Google Cloud AI Research
[3]Google DeepMind [4]Google Cloud AI

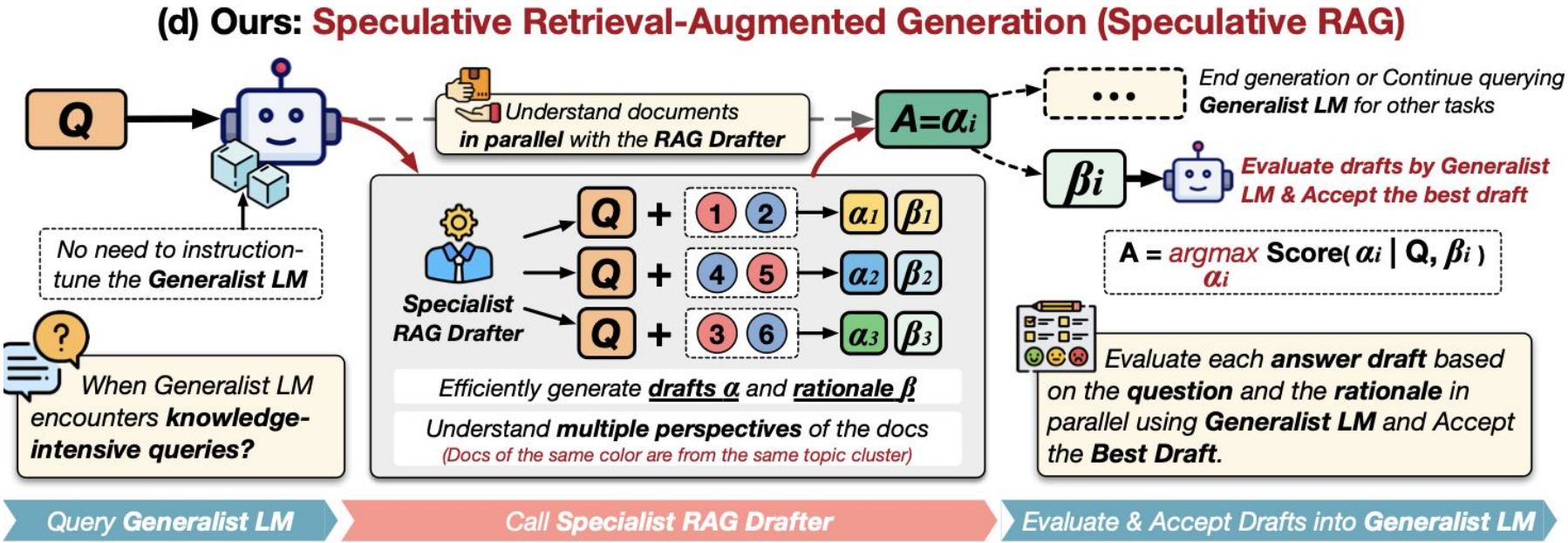
ABSTRACT

Retrieval augmented generation (RAG) combines the generative abilities of large language models (LLMs) with external knowledge sources to provide more accurate and up-to-date responses. Recent RAG advancements focus on improving retrieval outcomes through iterative LLM refinement or self-critique capabilities acquired through additional instruction tuning of LLMs. In this work, we introduce SPECULATIVE RAG – a framework that leverages a larger generalist LM to efficiently verify multiple RAG drafts produced in parallel by a smaller, distilled specialist LM. Each draft is generated from a distinct subset of retrieved documents, offering diverse perspectives on the evidence **while reducing input token counts per draft**. This approach enhances comprehension of each subset and mitigates potential **position bias over long context**. Our method accelerates RAG by delegating drafting to the smaller specialist LM, with the larger generalist LM performing a **single** verification pass over the drafts. Extensive experiments demonstrate that SPECULATIVE RAG achieves **state-of-the-art performance with reduced latency** on TriviaQA, MuSiQue, PopQA, PubHealth, and ARC-Challenge benchmarks. It notably enhances accuracy by up to 12.97% while reducing latency by 50.83% compared to conventional RAG systems on PubHealth.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable success in question answering tasks (Brown et al., 2020; Achiam et al., 2023; Team et al., 2023). Trained on massive datasets, LLMs leverage their extensive parametric memory to generate seemingly plausible responses to user queries (Kojima et al., 2022; Kamalloo et al., 2023). However, when faced with knowledge-intensive questions demanding up-to-date information or obscure facts (Petroni et al., 2021), LLMs struggle with factual inaccuracies and produce hallucinated contents (Huang et al., 2023).

Speculative RAG

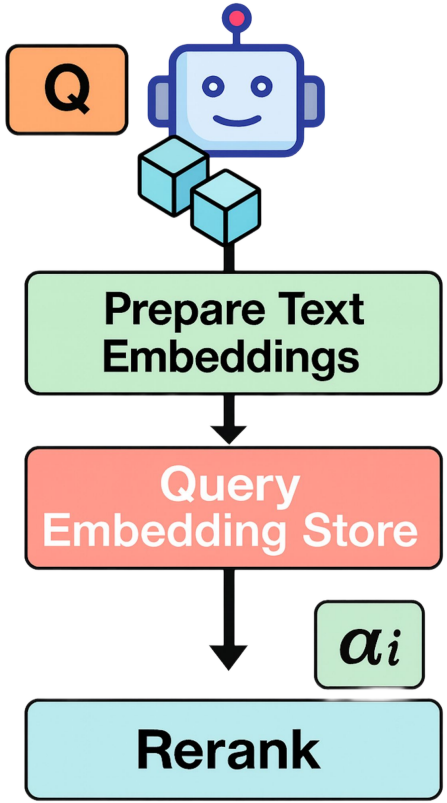


Multi-Head RAG

Runs parallel retrieval pipelines for diverse perspectives.

You need to build a system for complex multi-domain queries, so how do you gather multiple perspectives at once?

- Executes multiple retrieval processes simultaneously.
- Aggregates diverse sources into one response.
- Supports research and multidimensional analysis.



Multi-Head RAG

Besta et al, 2024
https://arxiv.org/abs/2406.05085

FLORIDA ATLANTIC

Multi-Head RAG: Solving Multi-Aspect Problems with LLMs

Maciej Besta* ETH Zurich
Ales Kubicek ETH Zurich
Robert Gerstenberger ETH Zurich
Marcin Chrapek ETH Zurich
Roman Niggli ETH Zurich

Patrik Okanovic ETH Zurich
Yi Zhu ETH Zurich
Patrick Iff ETH Zurich
Michal Podstawski NASK National Research Institute

Lucas Weitzendorf ETH Zurich
Mingyuan Chi ETH Zurich
Joanna Gajda Cledar
Piotr Nyczyk Cledar
Jürgen Müller BASF SE

Hubert Niewiadomski Cledar / IDEAS Research Institute
Torsten Hoefler ETH Zurich

Abstract

Retrieval Augmented Generation (RAG) enhances the abilities of Large Language Models (LLMs) by enabling the retrieval of documents into the LLM context to provide more accurate and relevant responses. Existing RAG solutions do not focus on queries that may require fetching multiple documents with substantially different contents. Such queries occur frequently, but are challenging because the embeddings of these documents may be distant in the embedding space, making it hard to retrieve them all. This paper introduces Multi-Head RAG (MRAG), a novel scheme designed to address this gap with a simple yet powerful idea: leveraging activations of Transformer's multi-head attention layer, instead of the decoder layer, as keys for fetching multi-aspect documents. The driving observation is that different attention heads learn to capture different data aspects. Harnessing the corresponding activations results in embeddings that represent various facets of data items and queries, improving the retrieval accuracy for complex queries. We provide an evaluation methodology and metrics, multi-aspect datasets, and real-world use cases to demonstrate MRAG's effectiveness. We show MRAG's design advantages over 18 RAG baselines, empirical improvements of up to 20% in retrieval success ratios, and benefits for downstream LLM generation. MRAG can be seamlessly integrated with existing RAG frameworks and benchmarks.

Website & code: `https://github.com/spcl/MRAG`

Multi-Head RAG

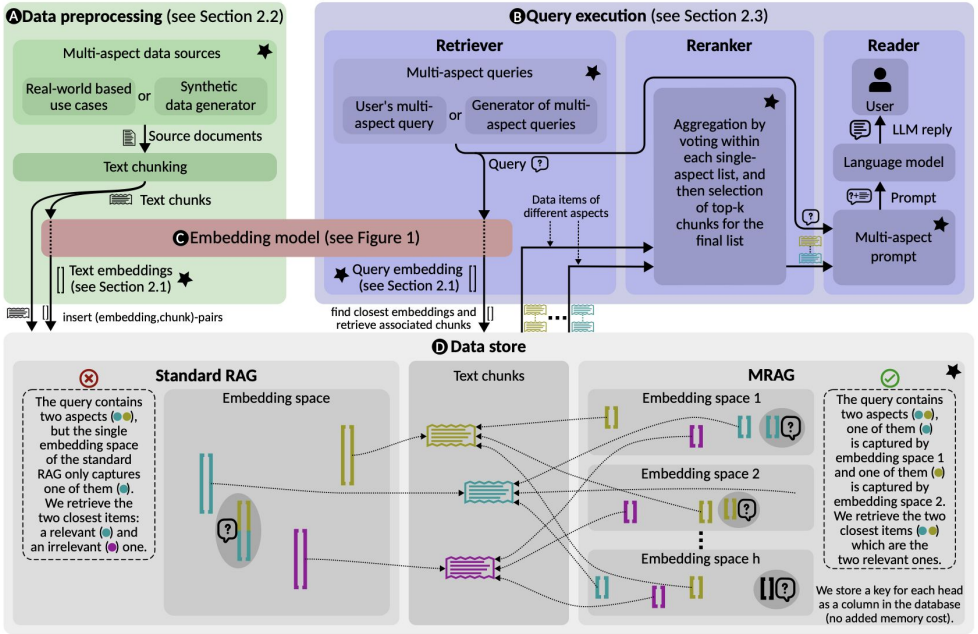


Figure 2: Overview of the MRAG pipeline, consisting of two parts: data preparation A and query execution B. The embedding model C and the data store D are used by both parts. The data store D contains text embeddings [] linking to text chunks ▭ reflecting three different aspects (cyan, magenta, yellow). Blocks marked by a star ★ are a novelty of this work.

Category D – Adaptive & Modular Models

You need to build a system for enterprises with diverse domains and structured knowledge, so how do you adapt pipelines and integrate multiple knowledge sources?

Use Case: HR policy Assistant

Developing an enterprise AI assistant that retrieves HR policies, financial records, and engineering documentation in one unified interface.

- Aggregates content from HR, Finance, and Engineering repositories.
- Returns citation-backed answers with links to original docs.
- Enforces role-based access control (RBAC) and data governance.

Why this type of RAG is needed?

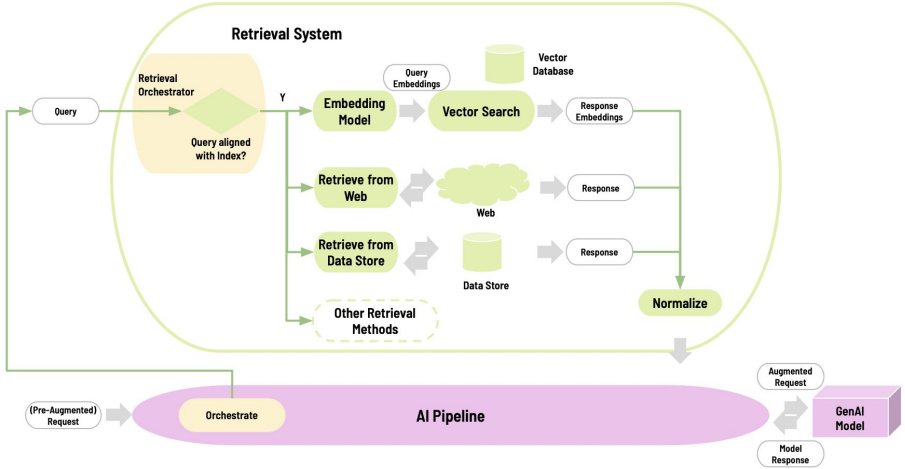
- Enterprise knowledge is siloed across many tools and formats.
- Compliance and audits require provenance, permissions, and traceability.
- Different teams need domain-aware retrieval

Adaptive RAG

Dynamically tuned for different domains or users.

You need to build a cross-domain system, so how do you adapt retrieval and generation to each domain or user profile?

- Fine-tunes embeddings and retrievers to domains.
- Learns user- or task-specific preferences.
- Supports personalization and industry-specific needs.



Adaptive RAG

Jeong et al, 2024
https://arxiv.org/abs/2403.14403

Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity

Soyeong Jeong[1] **Jinheon Baek**[2] **Sukmin Cho**[1] **Sung Ju Hwang**[1,2] **Jong C. Park**[1*]
School of Computing[1] Graduate School of AI[2]
Korea Advanced Institute of Science and Technology[1,2]
{starsuzi,jinheon.baek,nelllpic,sjhwang82,jongpark}@kaist.ac.kr

Abstract

Retrieval-Augmented Large Language Models (LLMs), which incorporate the non-parametric knowledge from external knowledge bases into LLMs, have emerged as a promising approach to enhancing response accuracy in several tasks, such as Question-Answering (QA). However, even though there are various approaches dealing with queries of different complexities, they either handle simple queries with unnecessary computational overhead or fail to adequately address complex multi-step queries; yet, not all user requests fall into only one of the simple or complex categories. In this work, we propose a novel adaptive QA framework that can dynamically select the most suitable strategy for (retrieval-augmented) LLMs from the simplest to the most sophisticated ones based on the query complexity. Also, this selection process is operationalized with a classifier, which is a smaller LM trained to predict the complexity level of incoming queries with automatically collected labels, obtained from actual predicted outcomes of models and inherent inductive biases in datasets. This approach offers a balanced strategy, seamlessly adapting between the iterative and single-step

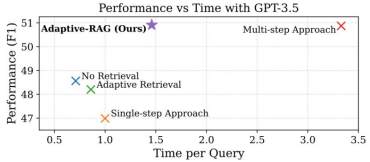
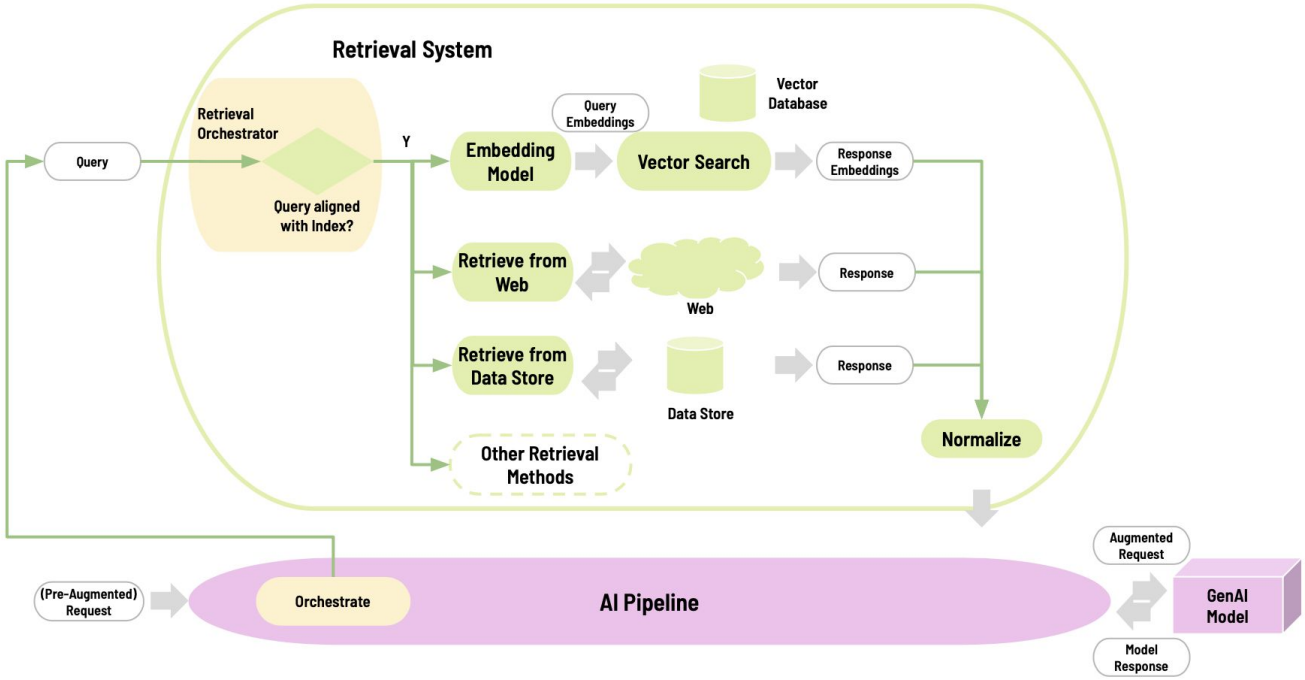


Figure 1: QA performance (F1) and efficiency (Time/Query) for different retrieval-augmented generation approaches. We use the GPT-3.5-Turbo-Instruct as the base LLM.

answering (QA) (Yang et al., 2018; Kwiatkowski et al., 2019). However, they still generate factually incorrect answers since their knowledge solely relies on their parametric memory (Kasai et al., 2022; Mallen et al., 2023). Meanwhile, memorizing all the (ever-changing) world knowledge may not be possible. To address this problem, retrieval-augmented LLMs (Borgeaud et al., 2022; Izacard et al., 2023; Shi et al., 2023), which incorporate non-parametric knowledge into LLMs with additional retrieval modules, have gained much increasing attention. Specifically, these models access a knowledge base, which serves as an extensive repository of information across various subjects and disciplines, to retrieve information relevant to

Adaptive RAG



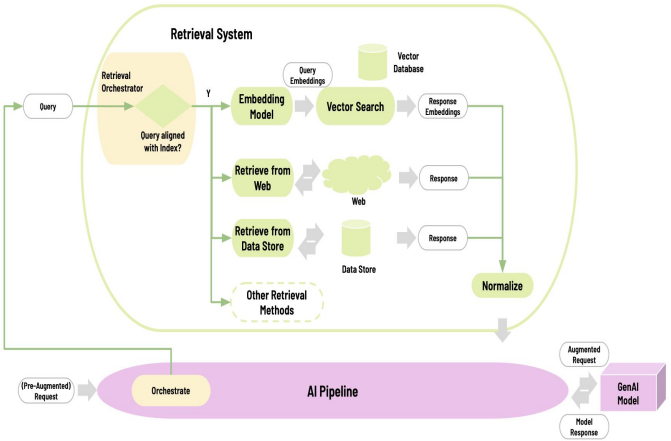
Adaptive RAG

Difference from Standard RAG

- Adaptive (MRAG) uses multiple embedding spaces, retrieving all relevant aspects, not irrelevant ones.
- Improves accuracy by covering multi-faceted queries comprehensively.

Query execution

- **Retriever**: Handles user's multi-aspect query or generates multi-aspect queries automatically.
- **Reranker**: Aggregates candidate chunks from different aspects, selects the top-k most relevant.
- **Reader**: Uses a language model to generate the final response based on the selected chunks and multi-aspect prompt.

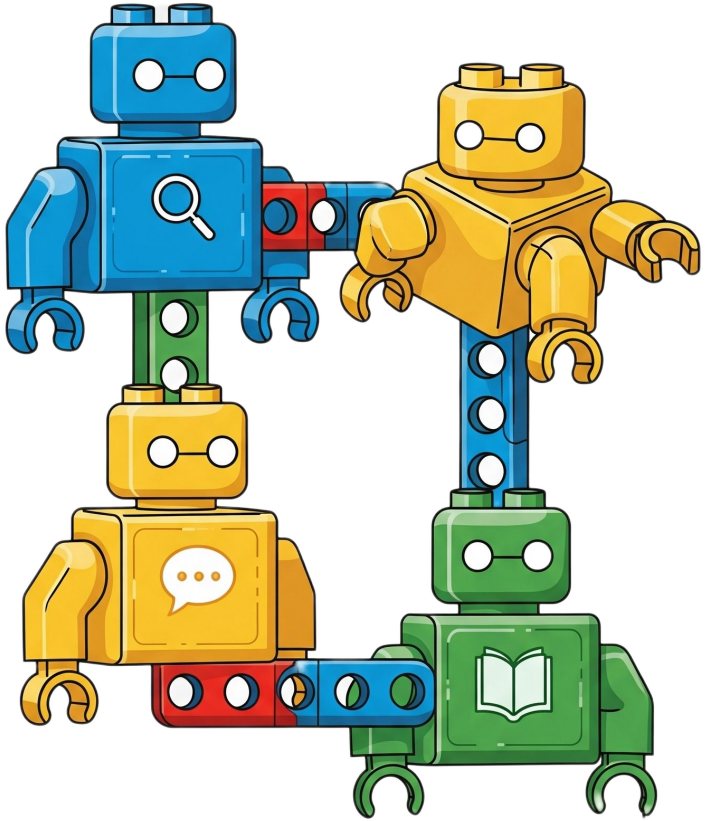


Modular RAG

Architected with interchangeable components.

You need to build an evolving system, so how do you design it with swappable modules instead of fixed components?

- Uses pluggable retrievers, rankers, and LLMs.
- Supports flexible system upgrades.
- Fits enterprise-scale conversational agents



Modular RAG

Gao et al, 2024

https://arxiv.org/abs/2407.21059

FLORIDA ATLANTIC

Modular RAG: Transforming RAG Systems into LEGO-like Reconfigurable Frameworks

Yunfan Gao, Yun Xiong, Meng Wang, Haofen Wang

***Abstract*—Retrieval-augmented Generation (RAG) has markedly enhanced the capabilities of Large Language Models (LLMs) in tackling knowledge-intensive tasks. The increasing demands of application scenarios have driven the evolution of RAG, leading to the integration of advanced retrievers, LLMs and other complementary technologies, which in turn has amplified the intricacy of RAG systems. However, the rapid advancements are outpacing the foundational RAG paradigm, with many methods struggling to be unified under the process of "retrieve-then-generate". In this context, this paper examines the limitations of the existing RAG paradigm and introduces the modular RAG framework. By decomposing complex RAG systems into independent modules and specialized operators, it facilitates a highly reconfigurable framework. Modular RAG transcends the traditional linear architecture, embracing a more advanced design that integrates routing, scheduling, and fusion mechanisms. Drawing on extensive research, this paper further identifies prevalent RAG patterns—linear, conditional, branching, and looping—and offers a comprehensive analysis of their respective implementation nuances. Modular RAG presents innovative opportunities for the conceptualization and deployment of RAG systems. Finally, the paper explores the potential emergence of new operators and paradigms, establishing a solid theoretical foundation and a practical roadmap for the continued evolution and practical deployment of RAG technologies.**

***Index Terms*—Retrieval-augmented generation, large language model, modular system, information retrieval**

I. Introduction

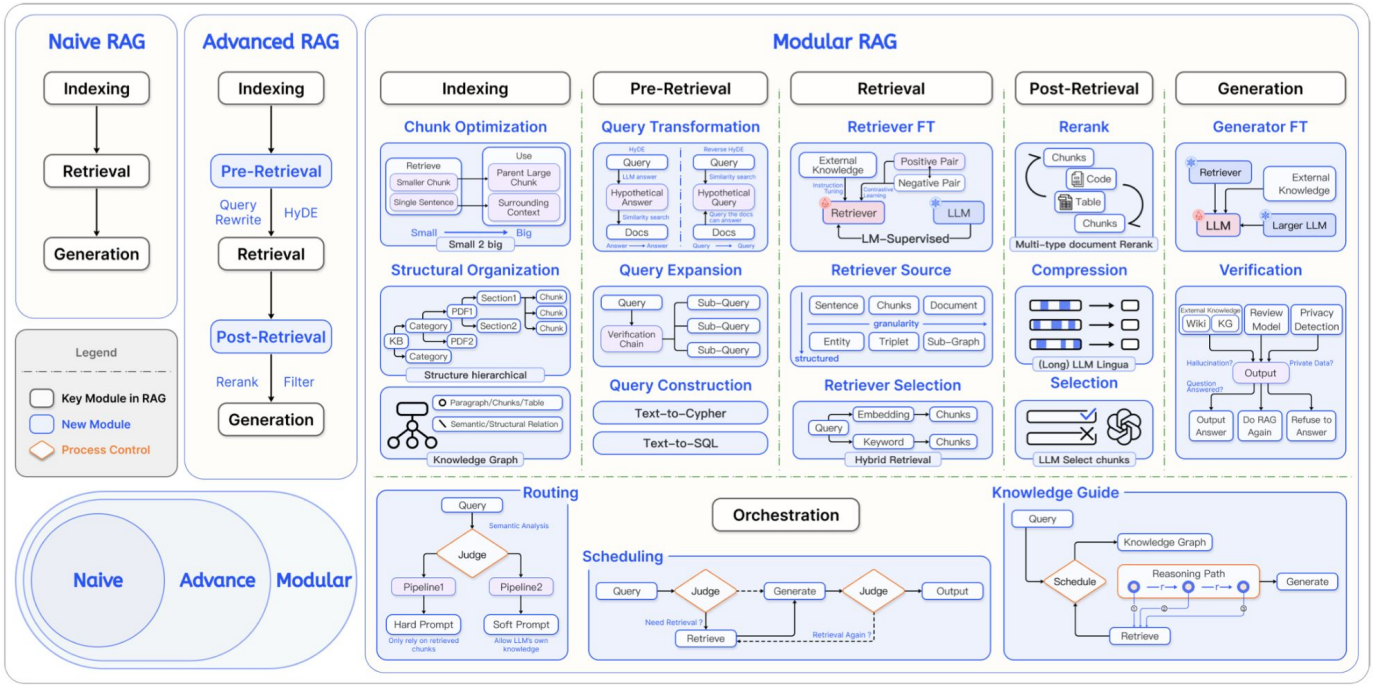
LARGE Language Models (LLMs) have demonstrated remarkable capabilities, yet they still face numerous challenges, such as hallucination and the lag in information updates [1]. Retrieval-augmented Generation (RAG), by accessing external knowledge bases, provides LLMs with important contextual information, significantly enhancing their perfor-

limitations of Naive RAG have become increasingly apparent. As depicted in Figure 1, it predominantly hinges on the straightforward similarity of chunks, result in poor performance when confronted with complex queries and chunks with substantial variability. The primary challenges of Naive RAG include: *1) Shallow Understanding of Queries.* The semantic similarity between a query and document chunk is not always highly consistent. Relying solely on similarity calculations for retrieval lacks an in-depth exploration of the relationship between the query and the document [8]. *2) Retrieval Redundancy and Noise.* Feeding all retrieved chunks directly into LLMs is not always beneficial. Research indicates that an excess of redundant and noisy information may interfere with the LLM's identification of key information, thereby increasing the risk of generating erroneous and hallucinated responses. [9]

To overcome the aforementioned limitations, Advanced RAG paradigm focuses on optimizing the retrieval phase, aiming to enhance retrieval efficiency and strengthen the utilization of retrieved chunks. As shown in Figure 1 ,typical strategies involve pre-retrieval processing and post-retrieval processing. For instance, query rewriting is used to make the queries more clear and specific, thereby increasing the accuracy of retrieval [10], and the reranking of retrieval results is employed to enhance the LLM's ability to identify and utilize key information [11].

Despite the improvements in the practicality of Advanced RAG, there remains a gap between its capabilities and real-world application requirements. On one hand, as RAG technology advances, user expectations rise, demands continue to evolve, and application settings become more complex. For instance, the integration of heterogeneous data and the new demands for system transparency, control, and maintainability.

Modular RAG

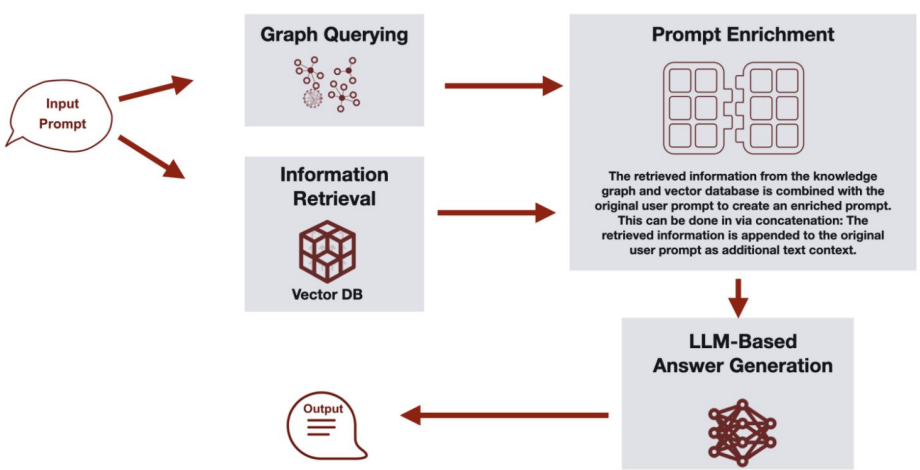

Naive RAG
Indexing
Retrieval
Generation
Advanced RAG
Indexing
Pre-Retrieval
Query Rewrite
HyDE
Retrieval
Post-Retrieval
Rerank
Filter
Generation
Legend
Key Module in RAG
New Module
Process Control
Naive
Advance
Modular
Modular RAG
Indexing
Chunk Optimization
Retrieve
Smaller Chunk
Single Sentence
Use
Parent Large Chunk
Surrounding Context
Small
Big
Small 2 big
Structural Organization
PDF1
PDF2
Category
Section1
Section2
Chunk
KB
Structure hierarchical
Paragraph/Chunks/Table
Semantic/Structural Relation
Knowledge Graph
Pre-Retrieval
Query Transformation
HyDE
Query
LLM answer
Hypothetical Answer
Similarity search
Docs
Reverse HyDE
Similarity search
Query the docs can answer
Query Expansion
Query
Verification Chain
Sub-Query
Query Construction
Text-to-Cypher
Text-to-SQL
Retrieval
Retriever FT
External Knowledge
Positive Pair
Negative Pair
Retriever
LLM
LM-Supervised
Retriever Source
Sentence
Chunks
Document
granularity
Entity
Triplet
Sub-Graph
structured
Retriever Selection
Query
Embedding
Keyword
Hybrid Retrieval
Post-Retrieval
Rerank
Chunks
Code
Table
Multi-type document Rerank
Compression
(Long) LLM Lingua
Selection
LLM Select chunks
Generation
Generator FT
Retriever
External Knowledge
LLM
Larger LLM
Verification
Wiki
KG
Review Model
Privacy Detection
Output
Output Answer
Do RAG Again
Refuse to Answer
Routing
Query
Semantic Analysis
Judge
Pipeline1
Pipeline2
Hard Prompt
Soft Prompt
Orchestration
Scheduling
Query
Judge
Generate
Output
Retrieve
Knowledge Guide
Query
Knowledge Graph
Schedule
Reasoning Path
Generate
Retrieve

Graph RAG

Incorporates structured knowledge graphs into retrieval.

You need to build a system for structured domains like drug discovery, so how do you exploit entity relationships instead of relying only on flat text?

- Connects knowledge graphs to retrieval pipelines.
- Captures relationships among entities.
- Enables reasoning in scientific and relational domains.



Graph RAG

Edge et al, 2024
https://arxiv.org/abs/2404.16130

FLORIDA ATLANTIC

From Local to Global: A GraphRAG Approach to Query-Focused Summarization

Darren Edge[1†] **Ha Trinh**[1†] **Newman Cheng**[2] **Joshua Bradley**[2] **Alex Chao**[3]

Apurva Mody[3] **Steven Truitt**[2] **Dasha Metropolitansky**[1] **Robert Osazuwa Ness**[1]

Jonathan Larson[1]

[1]Microsoft Research
[2]Microsoft Strategic Missions and Technologies
[3]Microsoft Office of the CTO

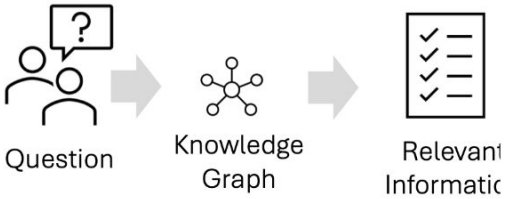
{daedge,trinhha,newmancheng,joshbradley,achao,moapurva,
steventruitt,dasham,robertness,jolarso}@microsoft.com

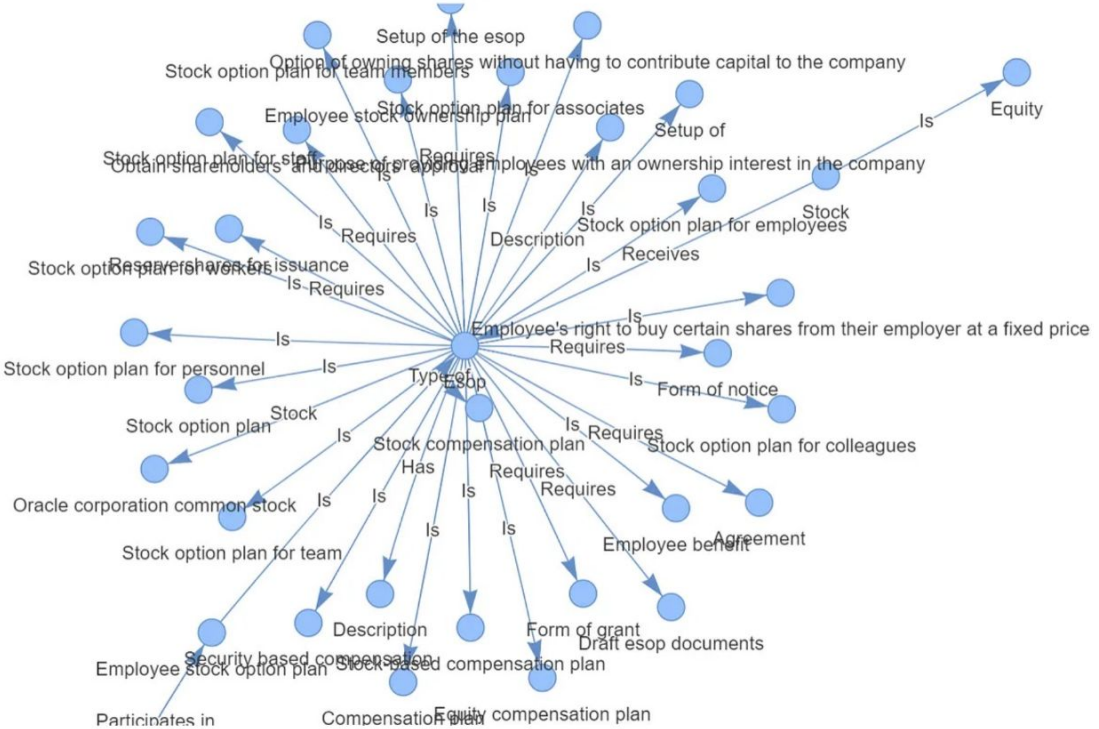
[†]These authors contributed equally to this work

Abstract

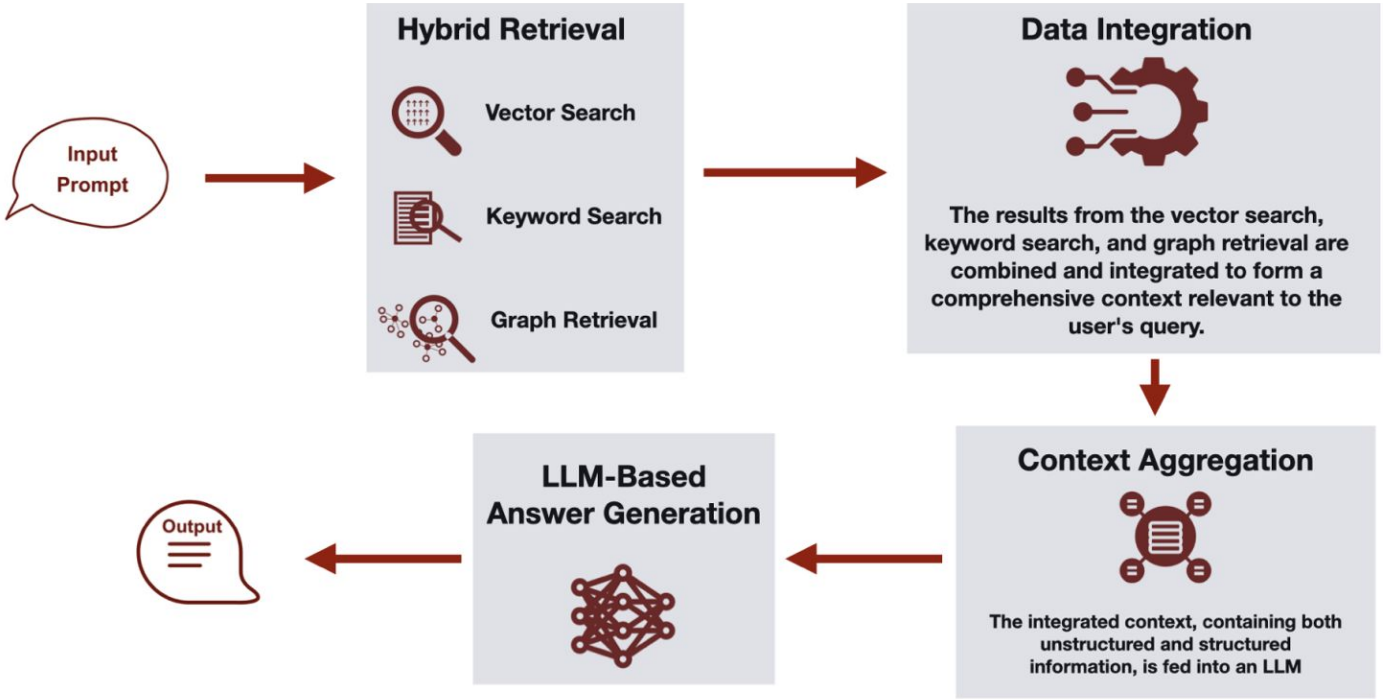
The use of retrieval-augmented generation (RAG) to retrieve relevant information from an external knowledge source enables large language models (LLMs) to answer questions over private and/or previously unseen document collections. However, RAG fails on global questions directed at an entire text corpus, such as "What are the main themes in the dataset?", since this is inherently a query-focused summarization (QFS) task, rather than an explicit retrieval task. Prior QFS methods, meanwhile, do not scale to the quantities of text indexed by typical RAG systems. To combine the strengths of these contrasting methods, we propose *GraphRAG*, a graph-based approach to question answering over private text corpora that scales with both the generality of user questions and the quantity of source text. Our approach uses an LLM to build a graph index in two stages: first, to derive an entity knowledge graph from the source documents, then to pre-generate community summaries for all groups of closely related entities. Given a question, each community summary is used to generate a partial response, before all partial responses are again summarized in a final response to the user. For a

About Knowledge Graphs





Graph-Enhanced Hybrid Retrieval



https://gradientflow.com/graphrag-design-patterns-challenges-recommendations/

Exercises

Exercise 4 - RAG Lab

Objective:
Implement a RAG pipeline that retrieves information from a document and uses a GenAI model to synthesize responses.

Go to Canvas -> Assignments

COT 6930 - Generative Intelligence and Software Development Lifecycles

Dr. Fernando Koch
kochf@fau.edu
http://www.fernandokoch.me