Overview about Retrieval Augmented Generation (RAG)

Yuting Hu

01/12/2024

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

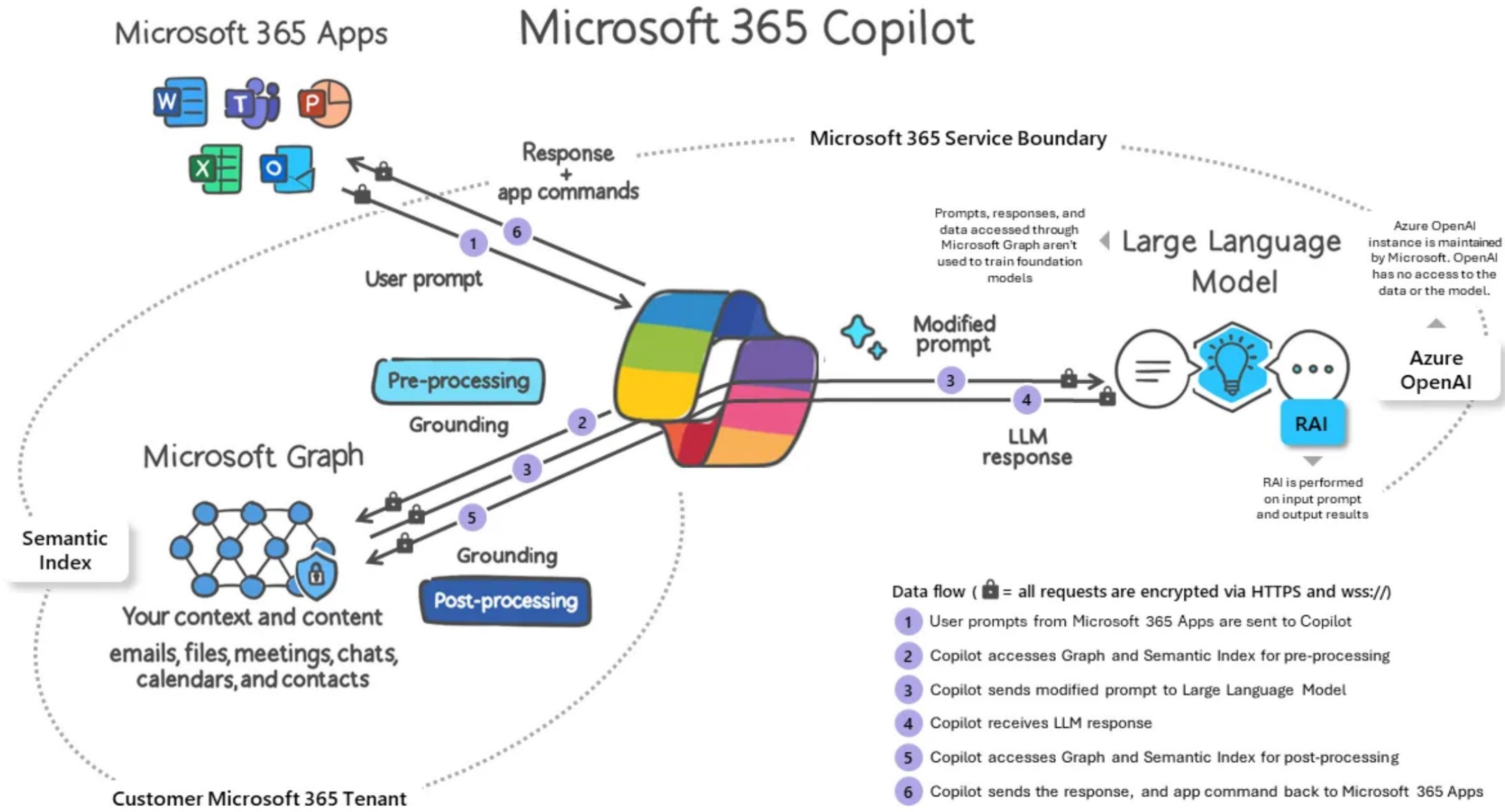
01 What is RAG?

Let's start from an example:

RAG in Microsoft 365 Copilot.

01 https://youtu.be/S7xTBa93TX8

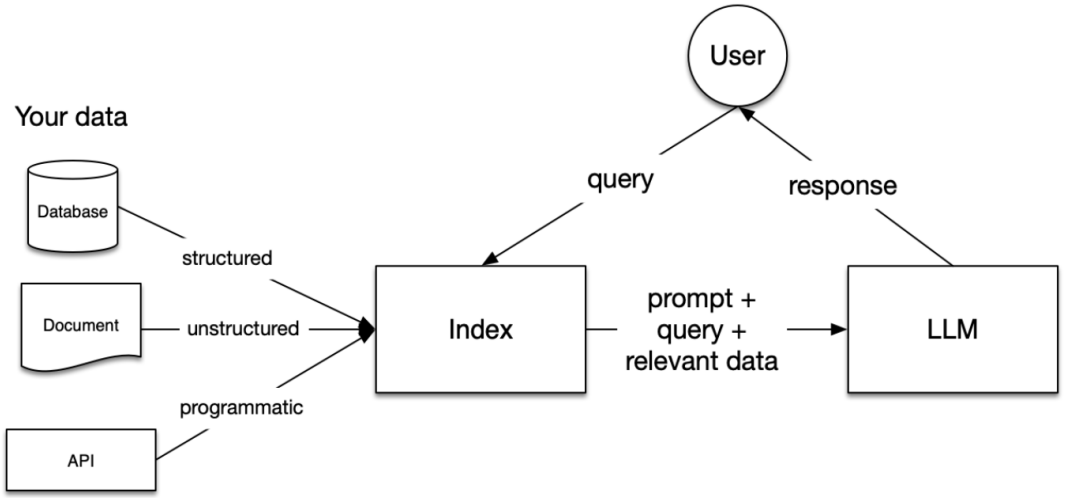
02 https://youtu.be/E5g20qmeKpg?si=bSVsCPqEtMn5Yzfs


Microsoft 365 Apps
Microsoft 365 Copilot
Microsoft 365 Service Boundary
Response + app commands
1
6
User prompt
Prompts, responses, and data accessed through Microsoft Graph aren't used to train foundation models
Large Language Model
Azure OpenAI instance is maintained by Microsoft. OpenAI has no access to the data or the model.
Modified prompt
3
4
LLM response
Azure OpenAI
RAI
RAI is performed on input prompt and output results
Pre-processing
Grounding
2
3
5
Grounding
Post-processing
Microsoft Graph
Semantic Index
Your context and content
emails, files, meetings, chats, calendars, and contacts
Customer Microsoft 365 Tenant
Data flow (= all requests are encrypted via HTTPS and wss://)
1 User prompts from Microsoft 365 Apps are sent to Copilot
2 Copilot accesses Graph and Semantic Index for pre-processing
3 Copilot sends modified prompt to Large Language Model
4 Copilot receives LLM response
5 Copilot accesses Graph and Semantic Index for post-processing
6 Copilot sends the response, and app command back to Microsoft 365 Apps

What is RAG?

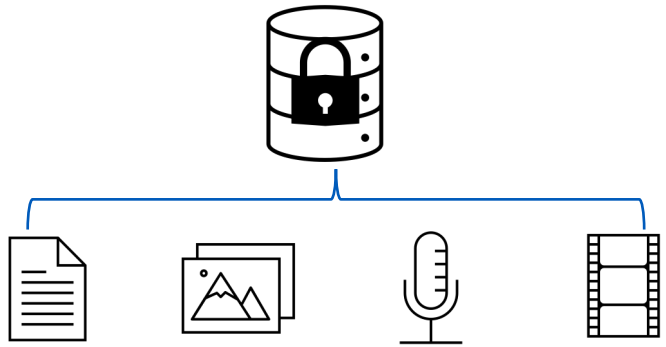
An architecture augments the capabilities of a Large Language Model (LLM) like ChatGPT by adding an information retrieval system that provides grounding data.

(An Open Book for LLMs)



Search + Prompting

External Database: used by LLMs to ground the generated answer. (eg. your private data)



Meet specific needs of users!

[Biggest business use-case of LLMs]

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

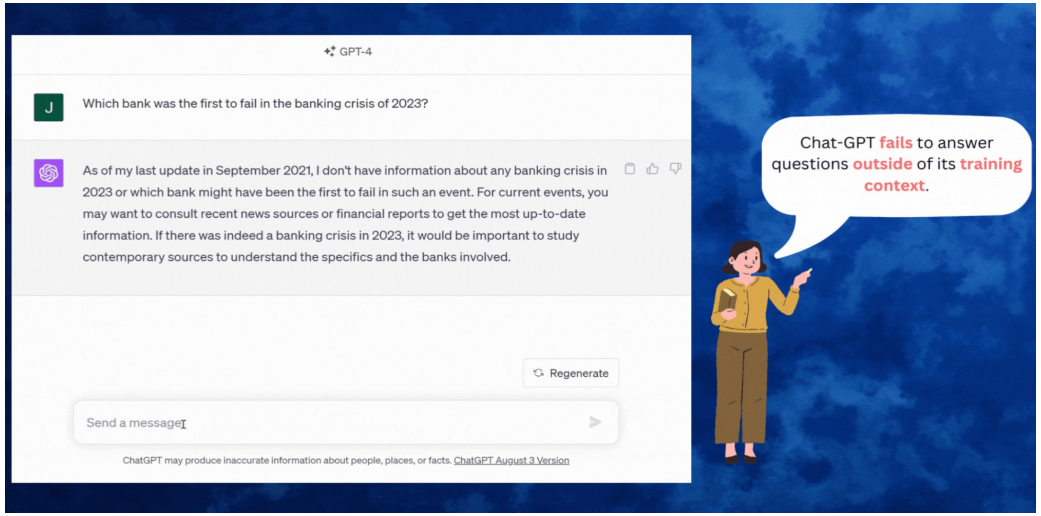
02 Why RAG is so important?

LLMs Hallucination

Why RAG is so important?

LLMs Challenges:

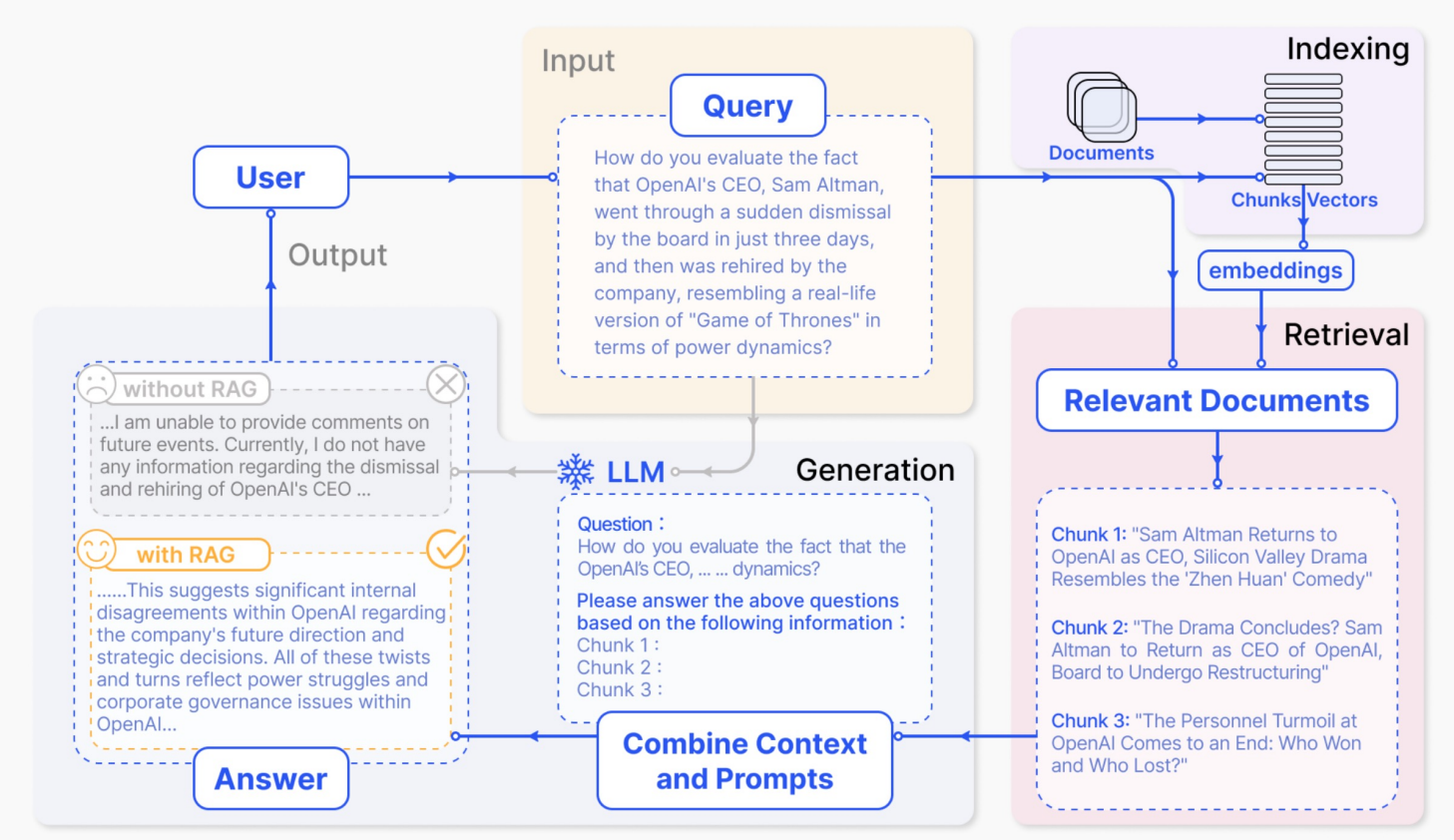
- Presenting false information when it does not have the answer.
- Presenting out-of-date or generic information when the user expects a specific, current response.
- Creating a response from non-authoritative sources.
- Creating inaccurate responses due to terminology confusion, wherein different training sources use the same terminology to talk about different things



LLMs Hallucinations (extend beyond training data or need up-to-date information)

LLMs Need to be Augmented!

RAG Benefits

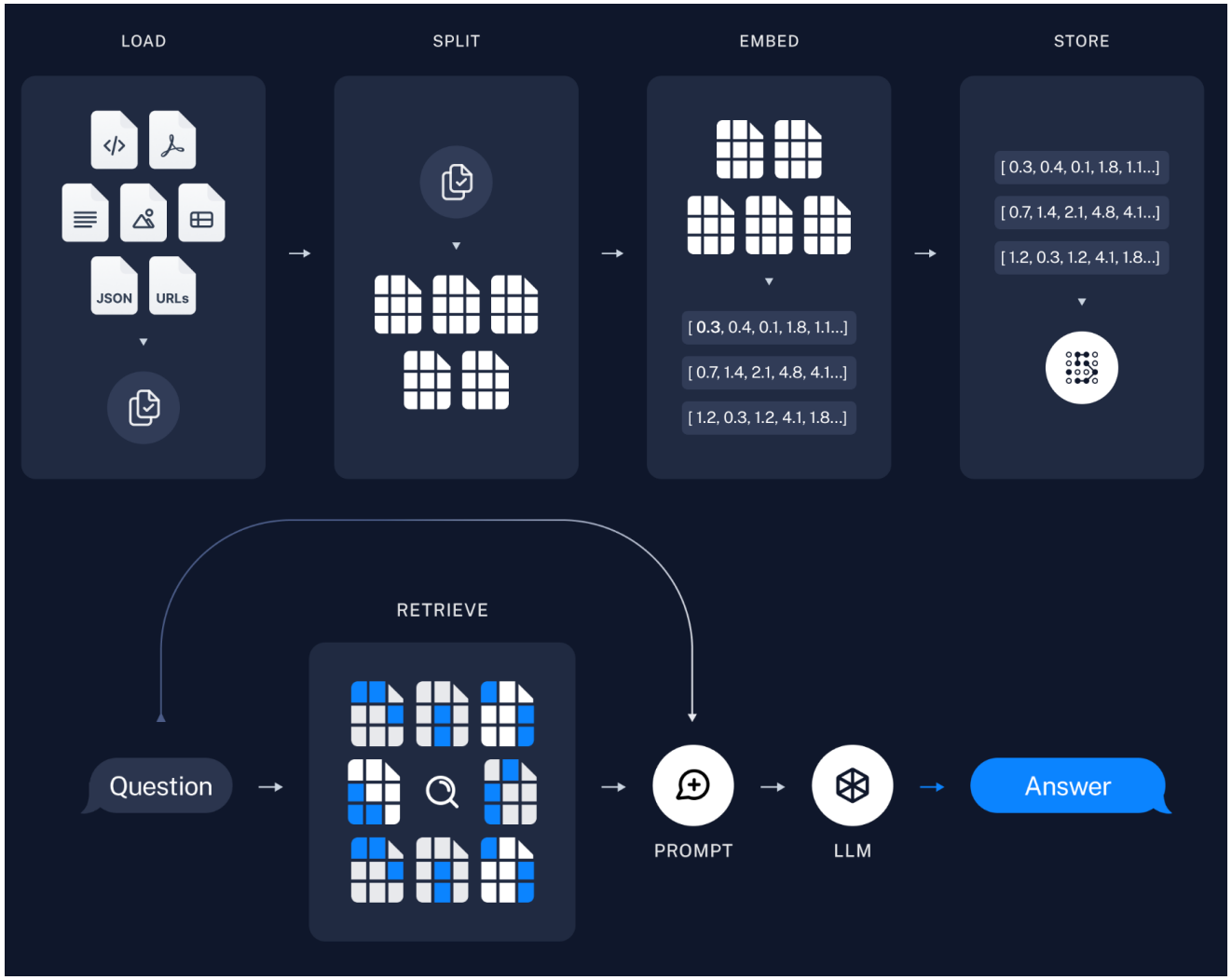


- Cost-effective implementation to introduce new data
- Provide current information
- Enhanced user trust
- More developer control

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

03 How does RAG work?

RAG Construction Workflow



Components

1)Offline Part

Load
Load documents

Split
Break document into smaller chunks

Store
Get embedding vectors from LLMs and save to vector store for indexing

2)Online Part

Retrieve
Indexing relevant chunks of user queries by vector similarity search (least cosine distance)

Generate
Augment prompt and get response from LLMs

The real world is much more complex …

Finite context window of transformer encoder

How to split documents into small, concise, meaningful chunks while support efficient index and hybrid index needs?

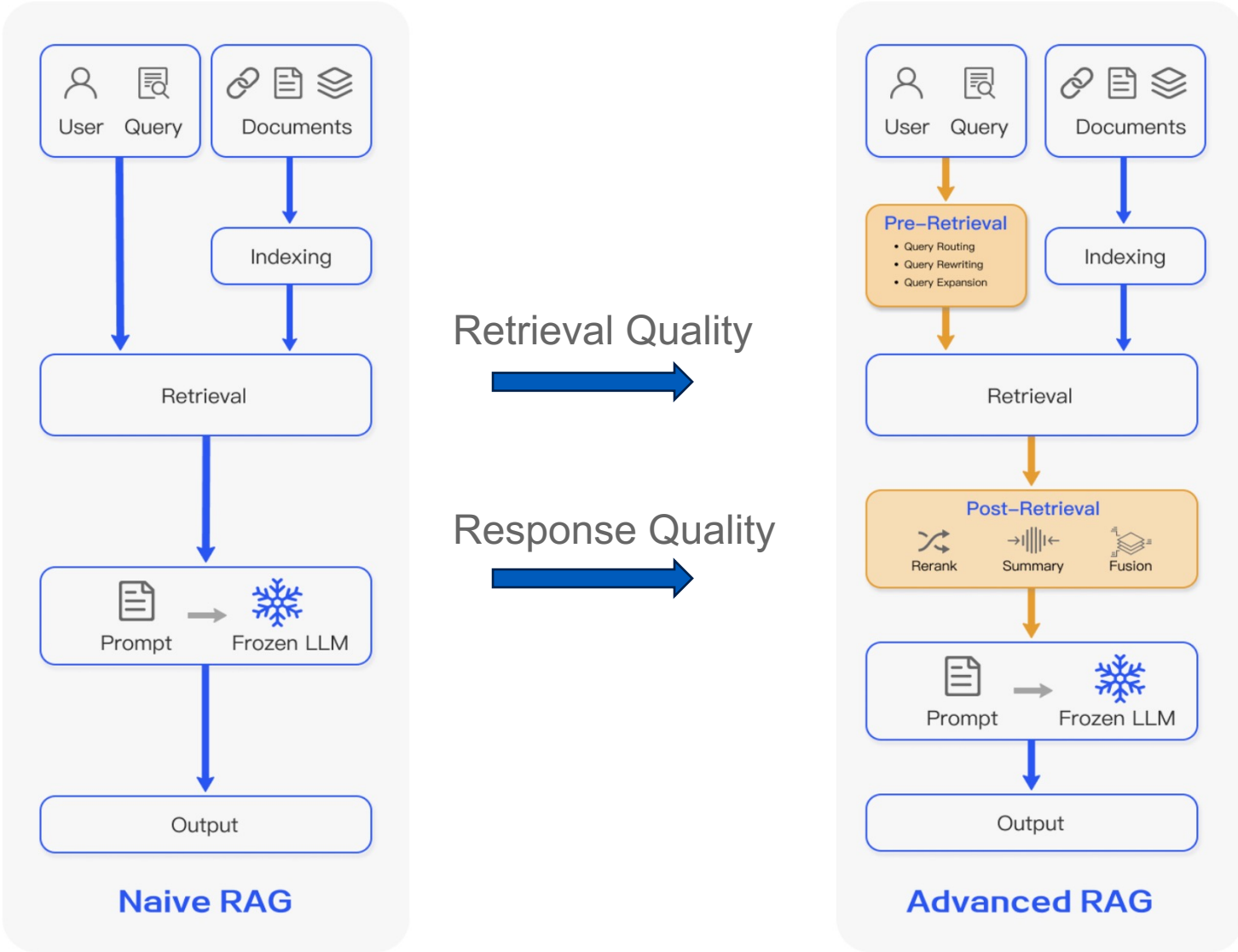
Flat index is not enough

Metafilter (search within dates or sources)

Efficient Index (target a small scope of relevant documents)

Context Enrichment (expand context from indexed chunks)

sentence window retrieval → expand context by sentences around the smaller retrieved chunk;
parent child chunks retrieval → split documents recursively into larger parent chunks, smaller child chunks;
fusion retrieval → combine the retrieved results with different similarity scores, post-reranking


User
Query
Documents
Indexing
Retrieval
Prompt
Frozen LLM
Output
Naive RAG
Retrieval Quality
Response Quality
User
Query
Documents
Pre-Retrieval
Query Routing
Query Rewriting
Query Expansion
Indexing
Retrieval
Post-Retrieval
Rerank
Summary
Fusion
Prompt
Frozen LLM
Output
Advanced RAG

From Simple to Advanced RAG

Table Stakes
Better Parsers
Chunk Sizes
Hybrid Search
Metadata Filters

Advanced Retrieval
Reranking
Recursive Retrieval
Embedded Tables
Small-to-big Retrieval

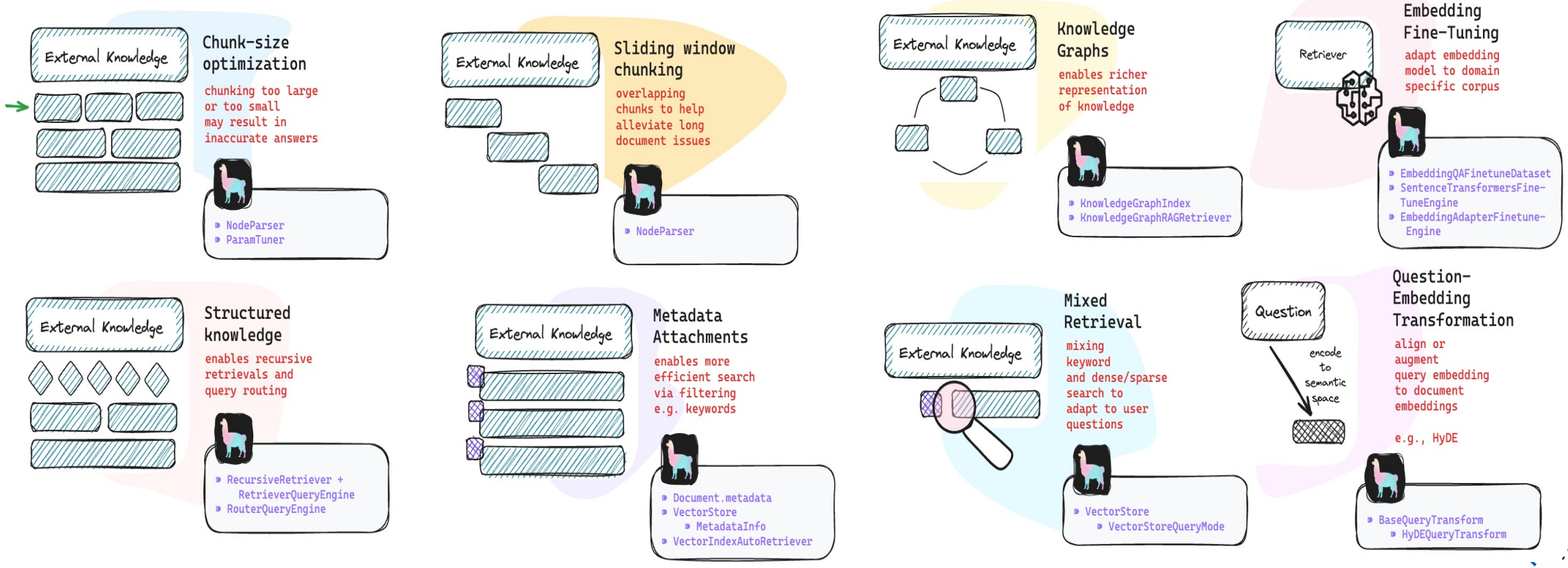
Fine-tuning
Embedding fine-tuning
LLM fine-tuning

Agentic Behavior
Routing
Query Planning
Multi-document Agents

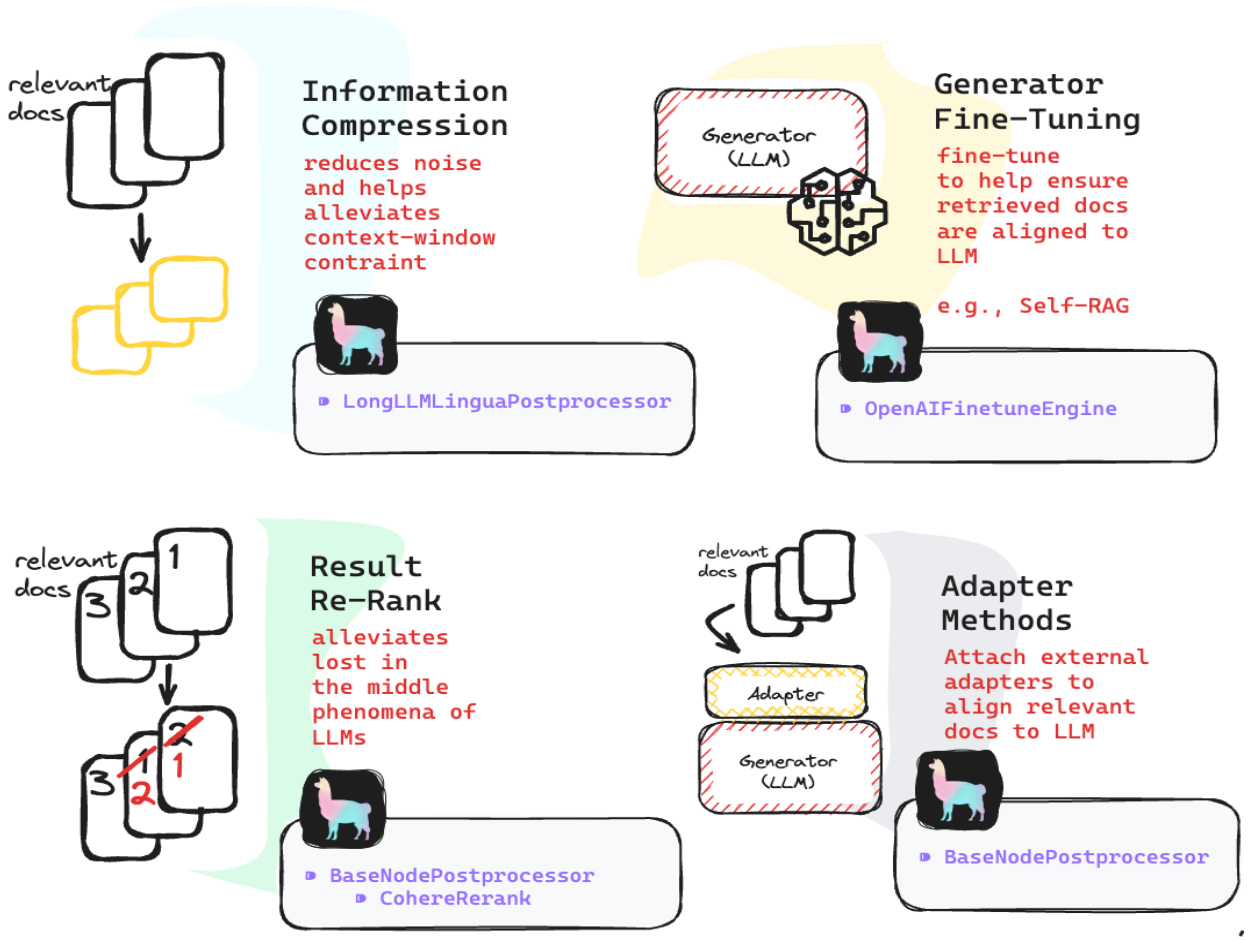
Less Expressive
Easier to Implement
Lower Latency/Cost

More Expressive
Harder to Implement
Higher Latency/Cost

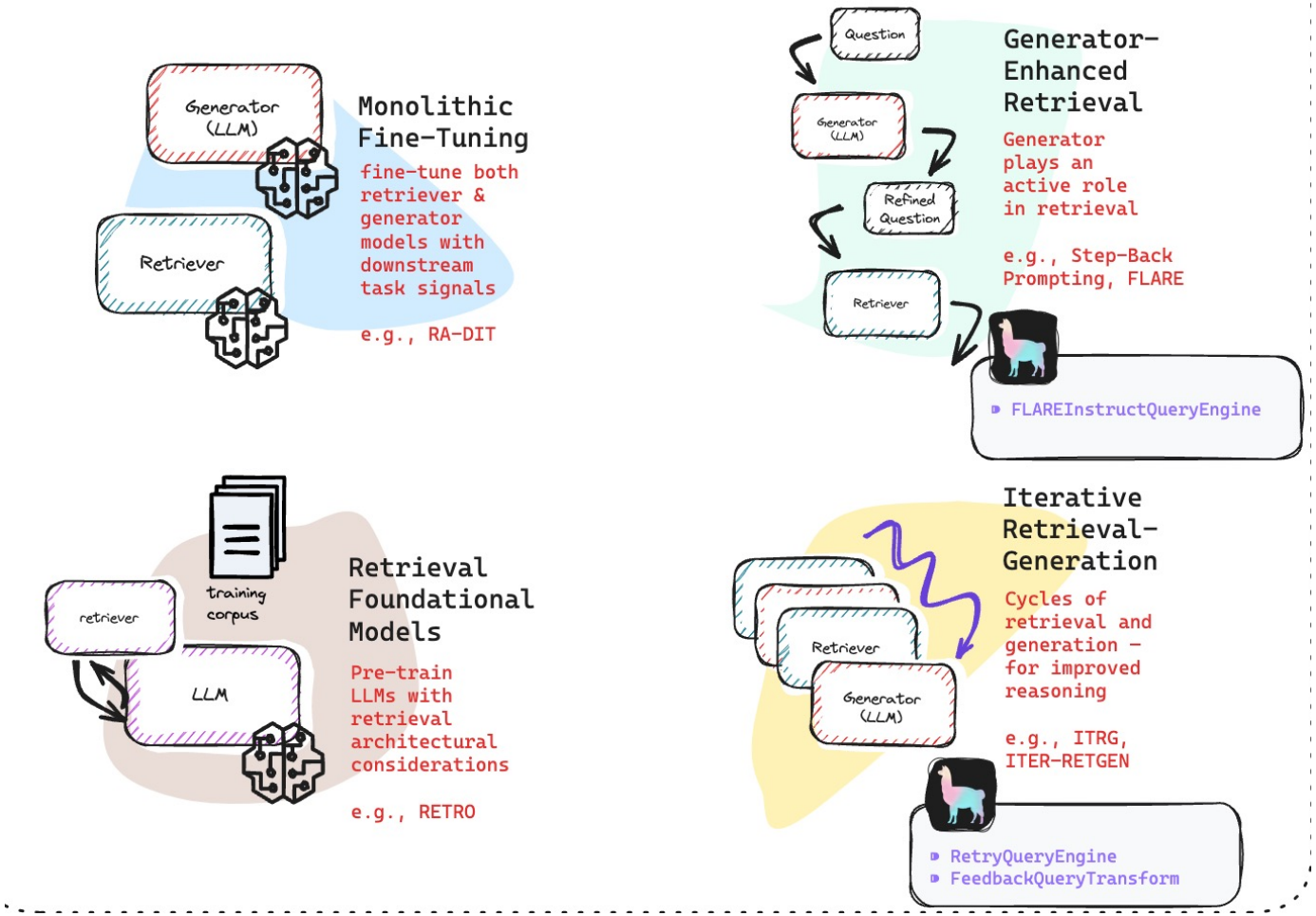
Techniques for Pre-retrieval in Advanced RAG



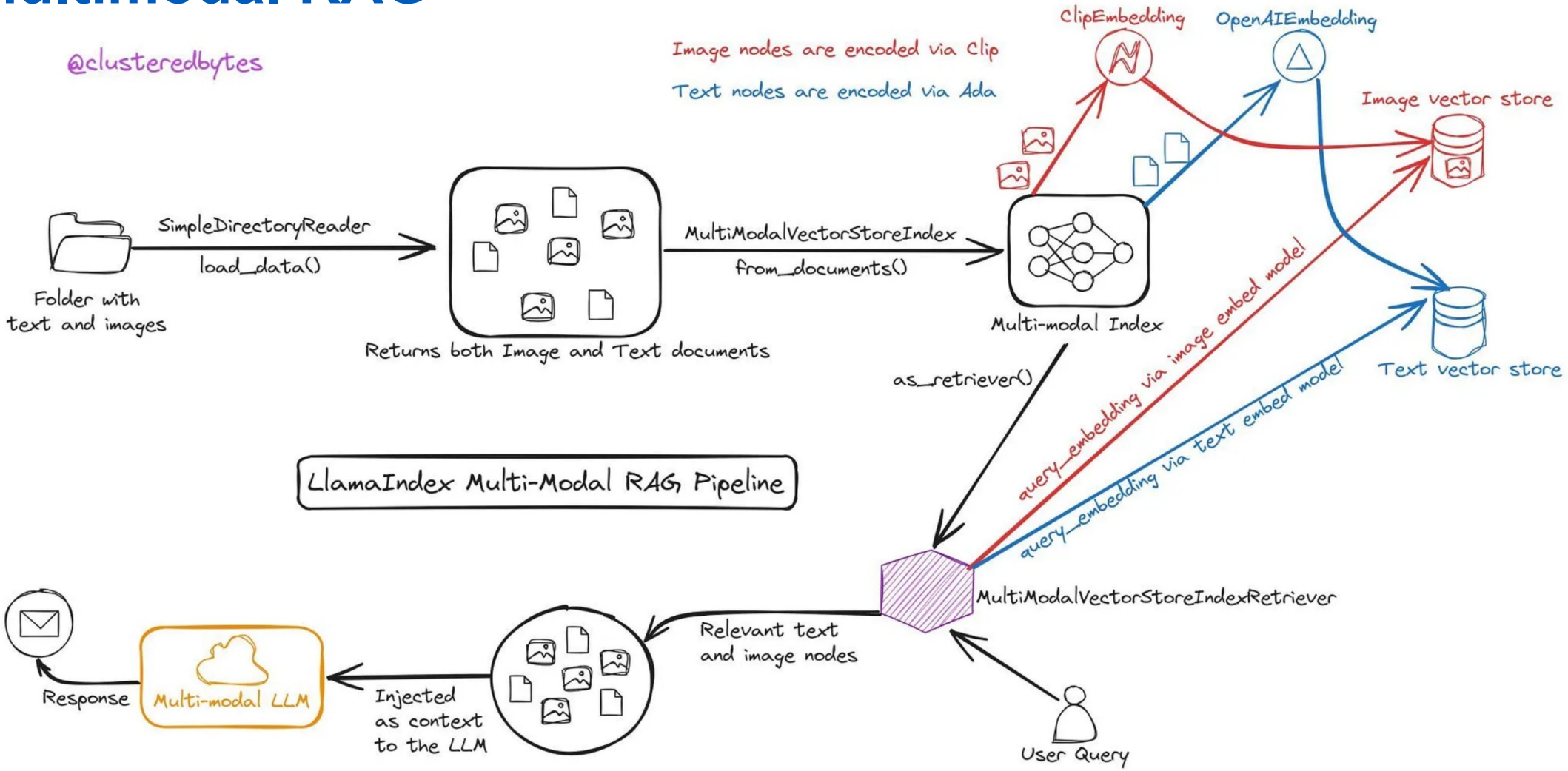
Techniques for Post-retrieval in Advanced RAG



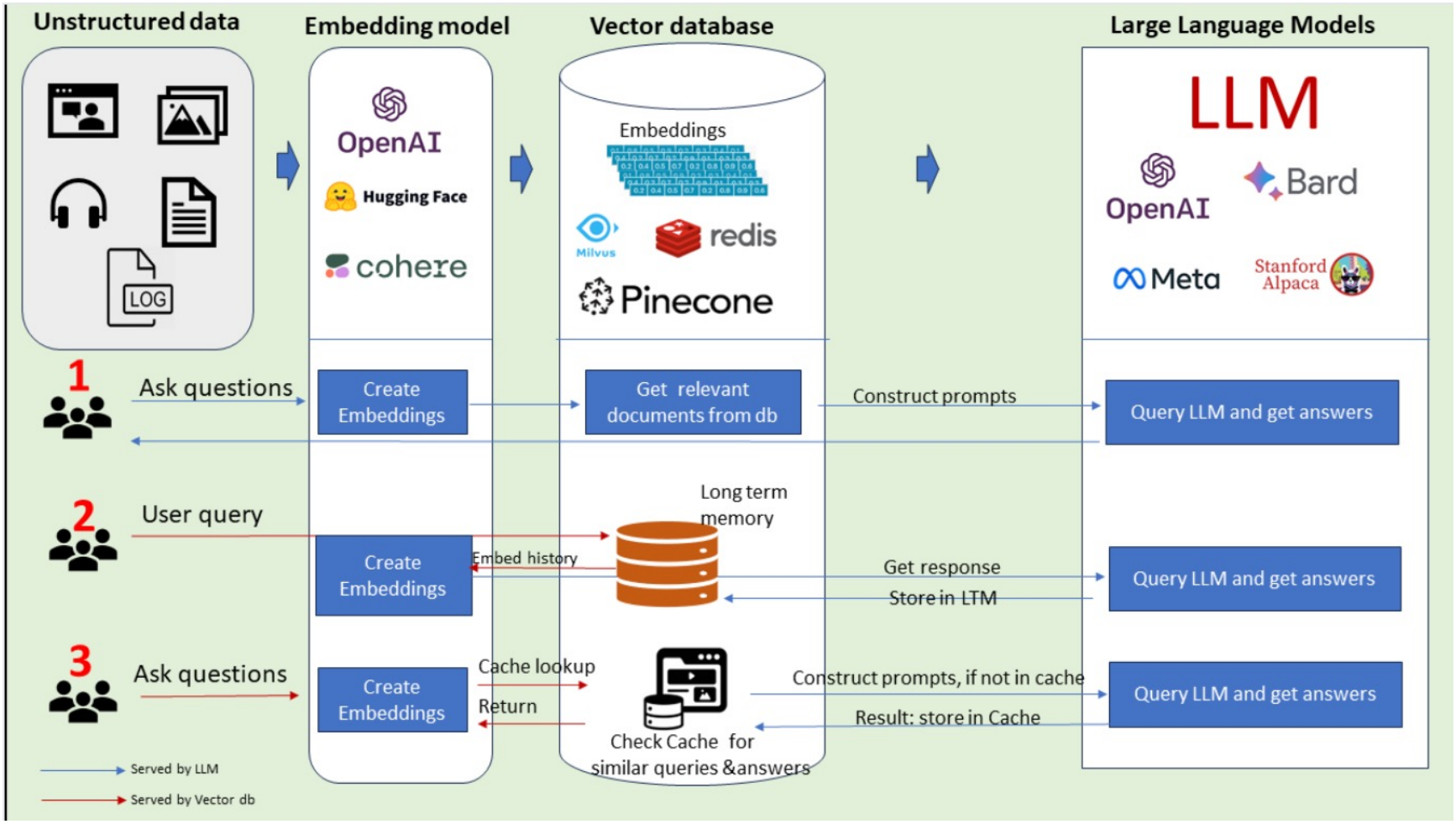
Techniques Combination in Advanced RAG



Multimodal RAG



3-Ways to Combine RAG+LLMs



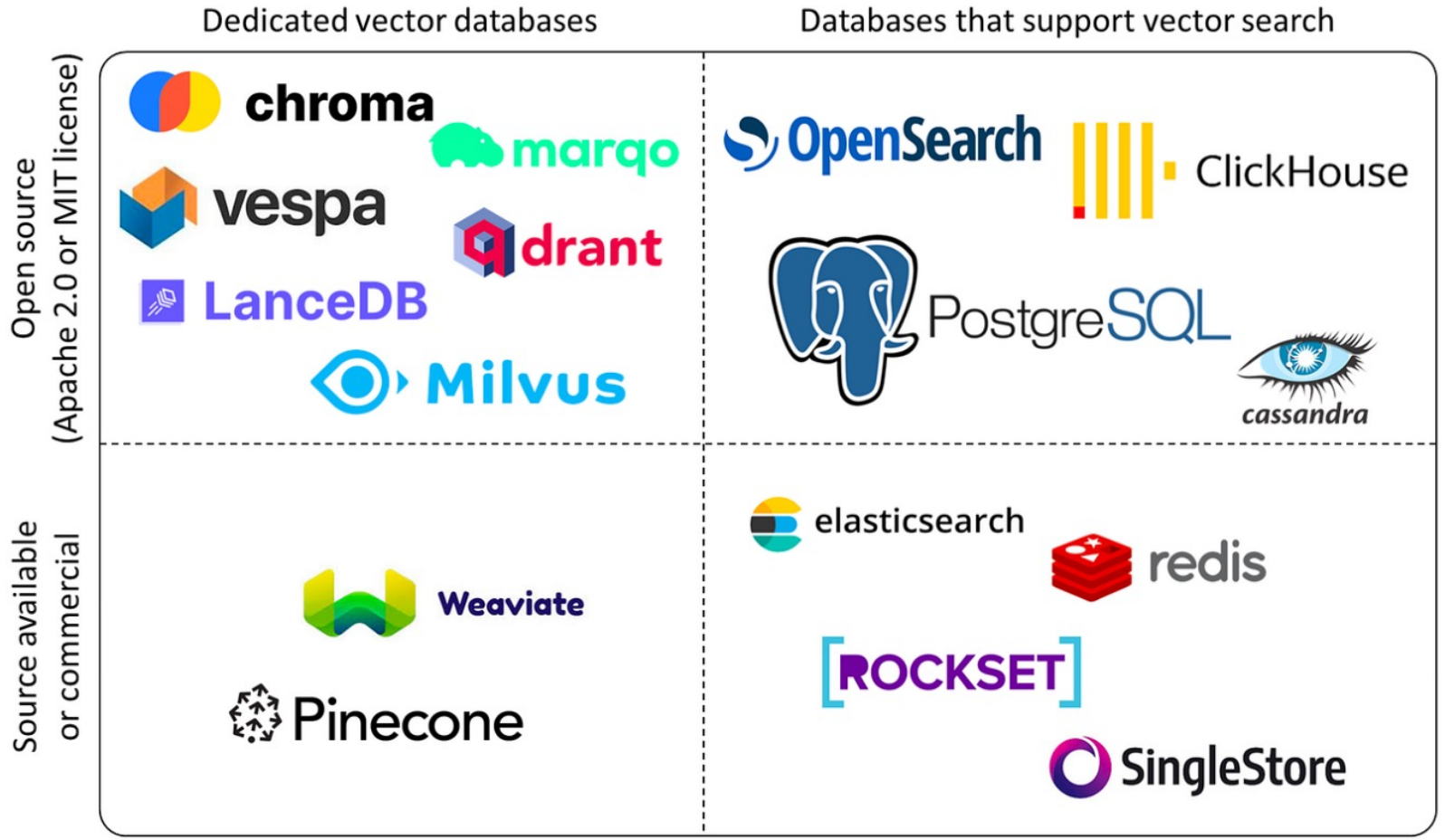
University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

04 RAG Ecological Chain

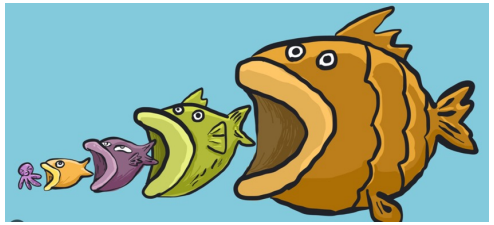
Companies In the Value Chain of RAG

Data Frameworks	Vector Databases	Knowledge Graph Infrastructure
Langchain	Pinecone	Neo4J
LlamaIndex	Marqo	NebulaGraph
Griptape	Qdrant	Amazon Neptune

Vector DB Providers



One Bad News for Vector DBs



OpenAI Assistants API

Complete Guide

Assistants API and Retrieval Tool

The Assistants API presently offers support for three distinct categories of tools, including code interpreters, retrieval, and function calling.

No Chunking, no embeddings, and no vector database are required. The advantage of using the assistants API is that you take a file, provide it to the assistant and can immediately start your RAG application. (**No Retrieval Quality Control!)**

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

05 RAG Recent Research

RAGs still has some issues ….

Relevance vs Similarity
computing similarity between document chunks and prompt does not always yield relevant contexts.

Optimize Chunk Size
striking a balance -— capturing all essential information without sacrificing speed.

Vector Search vs Knowledge Graph
knowledge graph can enforce consistent retrieval for certain key topics and concepts where the relationships are known.

Query Augmentation
fried chicken is more similiar to chicken soup or fried rice (varies by context)

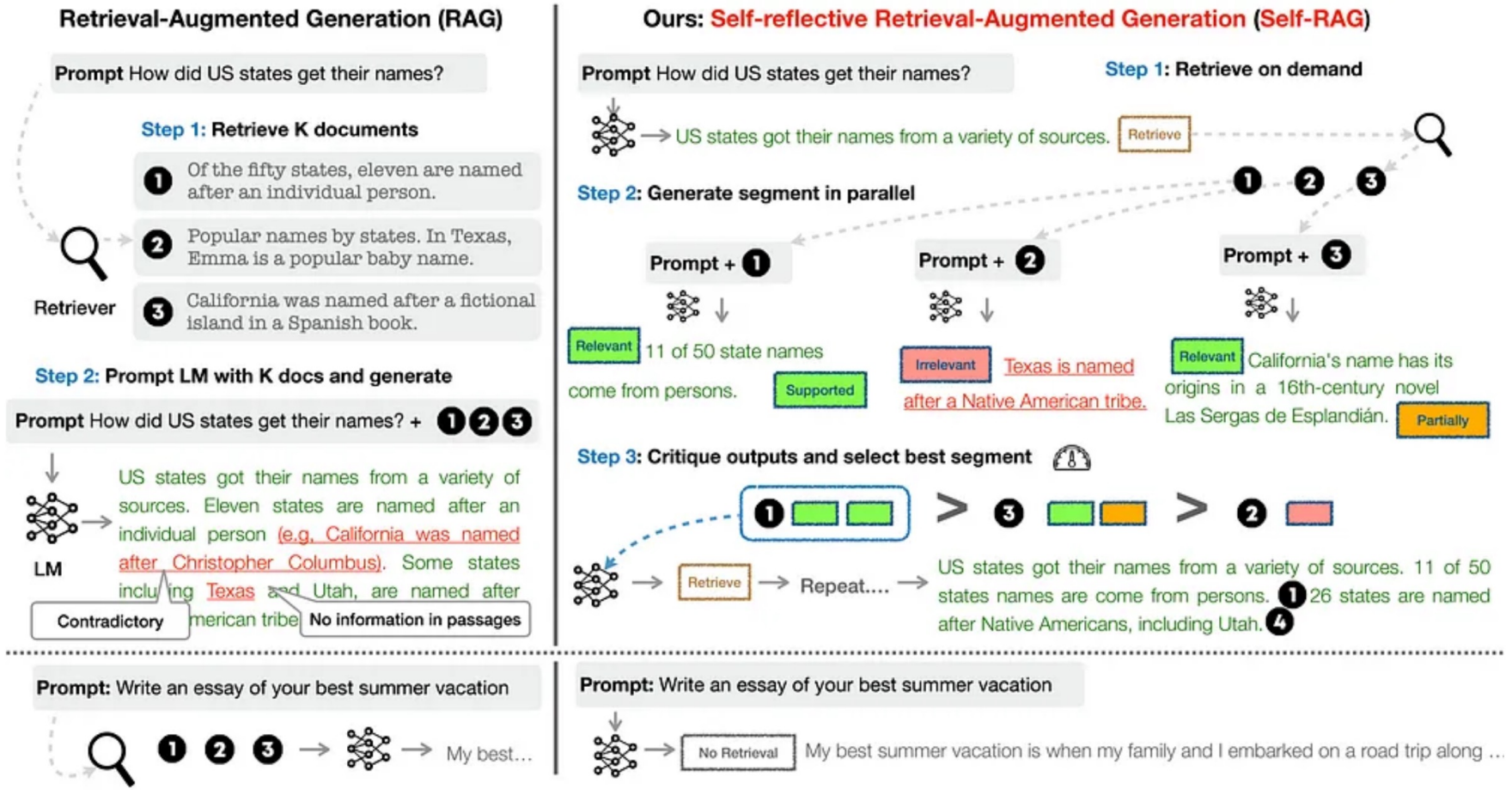
Query Planning
which city has the highest population? -> What is the population of Toronto? What is the population of Chicago?
(Query Understanding Layer @LLamaIndex)

Data Privacy and Security

Continuous Learning and Updating

Self-RAG

Self-RAG: Learning to Retrieve, Generate and Critique through Self-Reflections. 2024 ICLR Under Review



The authors develop a clever way for a fine-tuned LM (Llama2–7B and 13B) to output special tokens [Retrieval], [No Retrieval], [Relevant], [Irrelevant], [No support / Contradictory], [Partially supported], [Utility], etc. appended to LM generations to decide whether or not a context is relevant/irrelevant, the LM generated text from the context is supported or not, and the utility of the generation.

FLARE

Active Retrieval Augmented Generation. 2023 EMNLP

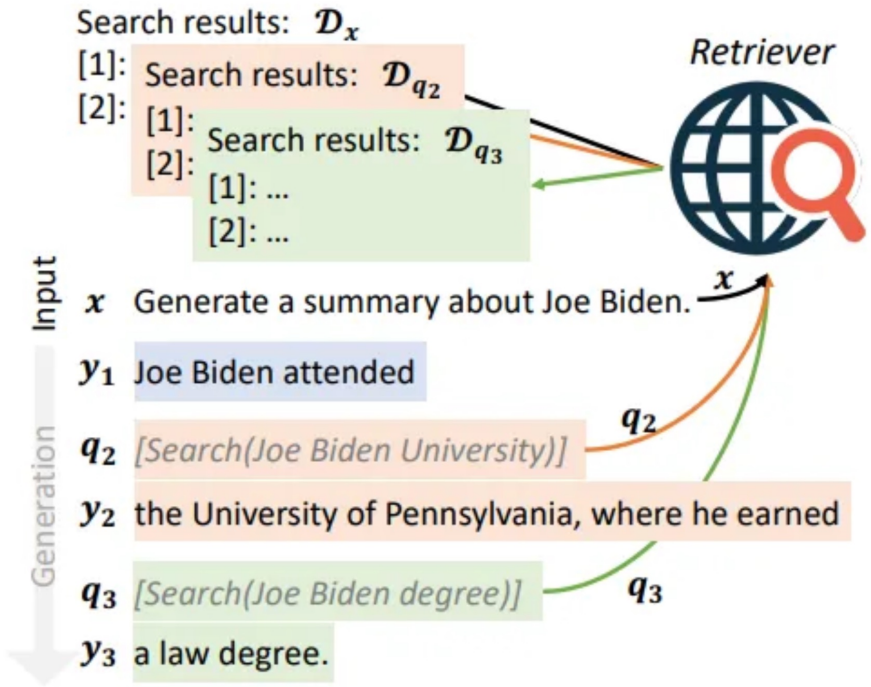
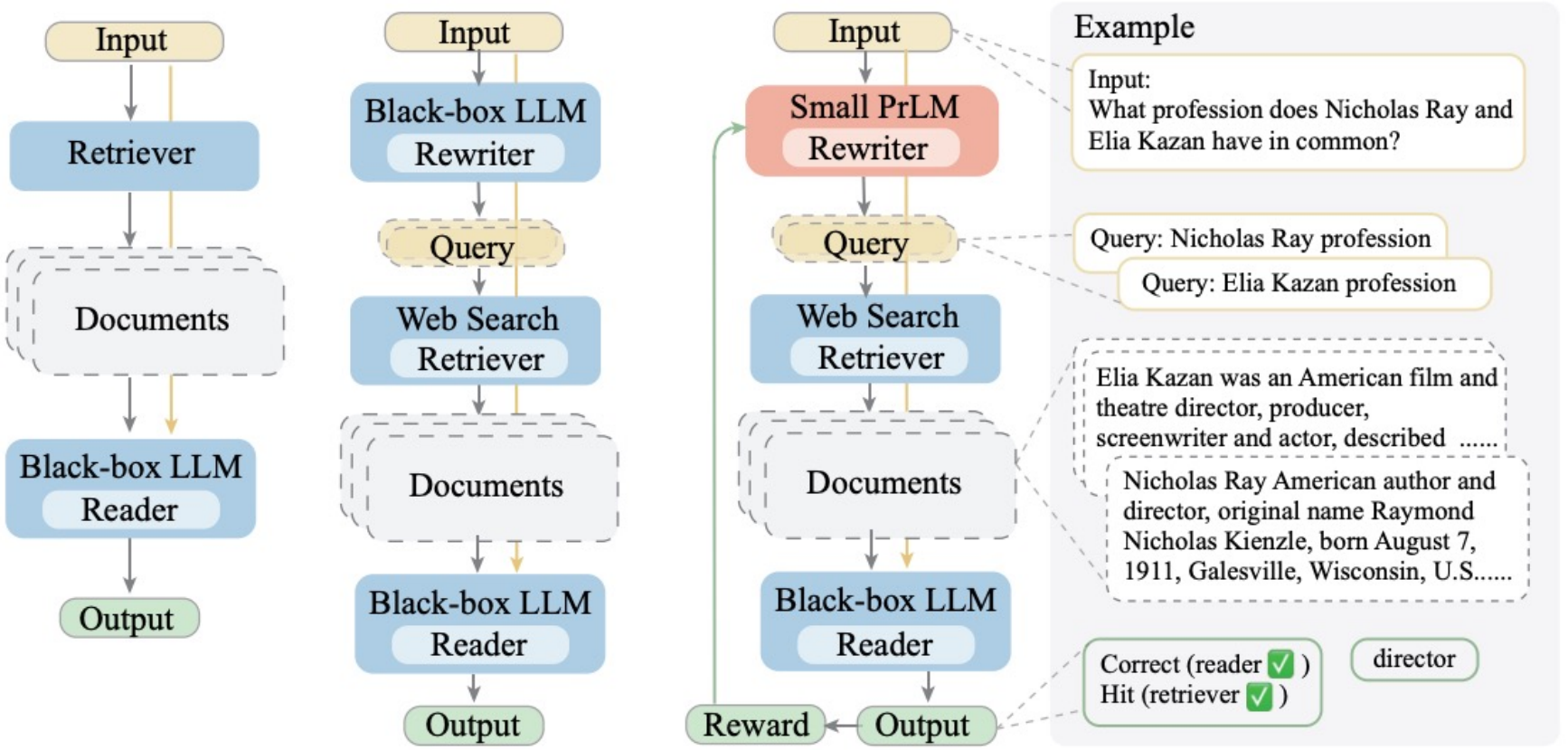


Figure 2: An illustration of forward-looking active retrieval augmented generation with retrieval instructions ($\text{FLARE}_{\text{instruct}}$). It iteratively generates search queries (shown in *gray italic*) to retrieve relevant information to aid future generations.

FLARE is used to handle cases where the userwant answers to be correct, and up to date — for which it makes sense to augment LLM knowledge with a real-time updated knowledge hub (the Internet).

In this workflow, first the user asks a question, and the LLM generates an initial partial sentence. This partial generation acts as a seed for an Internet search query (e.g. Joe Biden attended [X]). The result from this query is then integrated into the LLM response, and this process of continuous search and update continues, until the end of the generation.

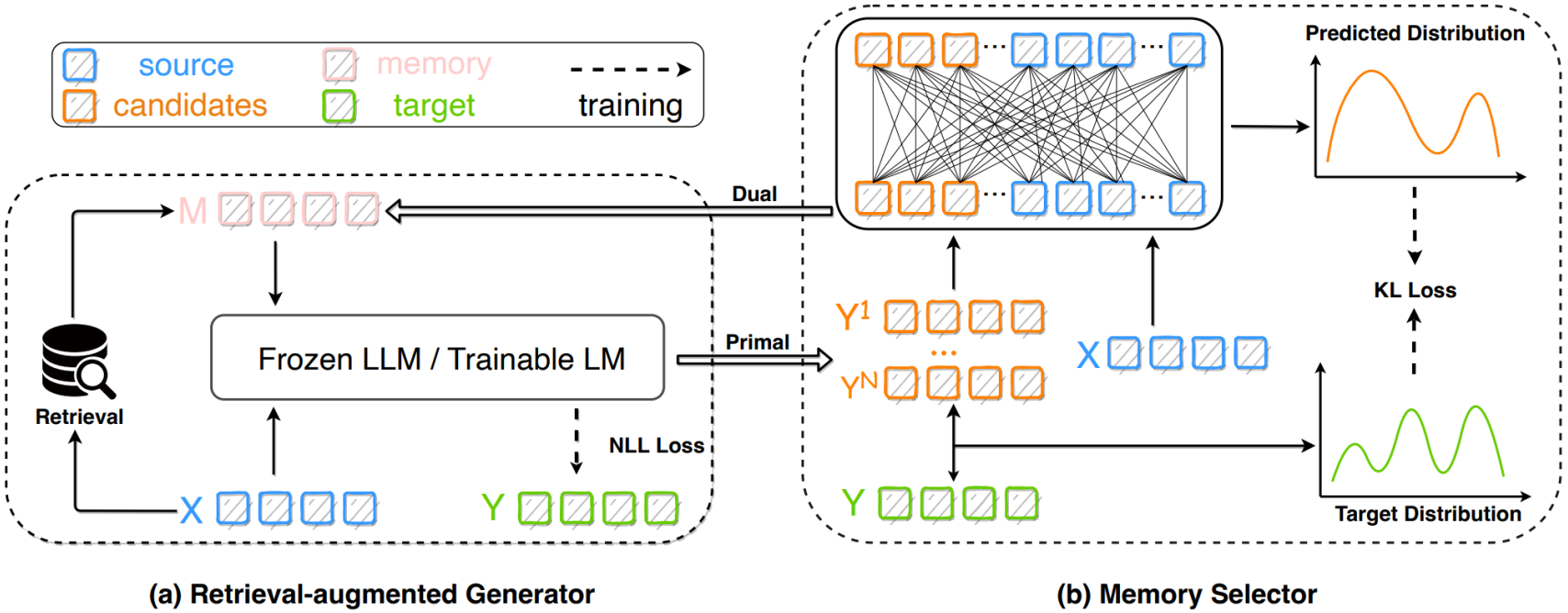
Query Rewriting for Retrieval-Augmented Large Language Models 2023 EMNLP



This work introduces a new framework, Rewrite-Retrieve-Read instead of the previous retrieve-then-read for the retrieval-augmented LLMs from the perspective of the query rewriting.

Lift Yourself Up: Retrieval-augmented Text Generation with Self-Memory

2023 NeurIPS



This work introduces a self-mem framework, iteratively employing a retrieval-augmented generator to create an unbounded memory pool and using a memory selector to choose one output as memory for the subsequent generation round. This enables the model to leverage its own output, referred to as self-memory, for improved generation.

References

Papers:

- Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf)
- Retrieval-Augmented Generation for Large Language Models: A Survey (https://arxiv.org/pdf/2312.10997.pdf)
- Self-RAG: Learning to Retrieve, Generate and Critique through Self-Reflections (https://selfrag.github.io/)
- Active Retrieval Augmented Generation (https://aclanthology.org/2023.emnlp-main.495.pdf)
- Query Rewriting for Retrieval-Augmented Large Language Models (https://aclanthology.org/2023.emnlp-main.322.pdf)
- Lift Yourself Up: Retrieval-augmented Text Generation with Self-Memory (https://openreview.net/pdf?id=lYNSvp51a7)

Blogs:

- https://www.anyscale.com/blog/a-comprehensive-guide-for-building-rag-based-llm-applications-part-1
- https://neo4j.com/developer-blog/fine-tuning-retrieval-augmented-generation/
- https://medium.com/dphi-tech/implement-rag-with-knowledge-graph-and-llama-index-6a3370e93cdd
- https://medium.com/enterprise-rag/injecting-knowledge-graphs-in-different-rag-stages-a3cd1221f57b
- https://medium.com/llamaindex-blog/a-cheat-sheet-and-some-recipes-for-building-advanced-rag-803a9d94c41b

References

Blogs:

- https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6
- https://python.langchain.com/docs/use_cases/question_answering/
- https://docs.llamaindex.ai/en/stable/getting_started/concepts.html
- https://weaviate.io/blog/multimodal-rag
- https://docs.llamaindex.ai/en/stable/examples/query_engine/knowledge_graph_rag_query_engine.html
- https://dxiaochuan.medium.com/optimizing-retrieval-for-rag-applications-enhancing-contextual-knowledge-in-llms-79ebcafe5f6e
- https://www.pinecone.io/learn/retrieval-augmented-generation/
- https://medium.com/emalpha/innovations-in-retrieval-augmented-generation-8e6e70f95629
- https://www.pinecone.io/learn/vector-database/
- https://www.datacamp.com/blog/the-top-5-vector-databases
- https://wandb.ai/cosmo3769/RAG/reports/A-Gentle-Introduction-to-Retrieval-Augmented-Generation-RAG---Vmlldzo1MjM4Mjk1

References

Blogs:

- https://medium.com/emalpha/innovations-in-retrieval-augmented-generation-8e6e70f95629
- https://medium.com/enterprise-rag/a-first-intro-to-complex-rag-retrieval-augmented-generation-a8624d70090f
- https://www.arcus.co/blog/rag-at-planet-scale
- https://medium.com/enterprise-rag/injecting-knowledge-graphs-in-different-rag-stages-a3cd1221f57b
- https://medium.com/programmers-journey/three-open-source-rag-tools-you-need-to-know-about-331c3f28ab22
- https://medium.com/@zilliz_learn/do-we-still-need-vector-databases-for-rag-with-openais-releasing-of-its-built-in-retrieval-9bc960a1df45
- https://docs.llamaindex.ai/en/stable/examples/agent/openai_assistant_query_cookbook.html

Videos:

- https://www.youtube.com/watch?v=TRjq7t2Ms5I&ab_channel=AIEngineer