香港城市大學（東莞）
City University of Hong Kong
(Dongguan)

Celebrating the Establishment in Year 2024

Lecture 11: Advanced topics and recent trends

Retrieval Augmented Generation (RAG)

CS6493 Natural Language Processing

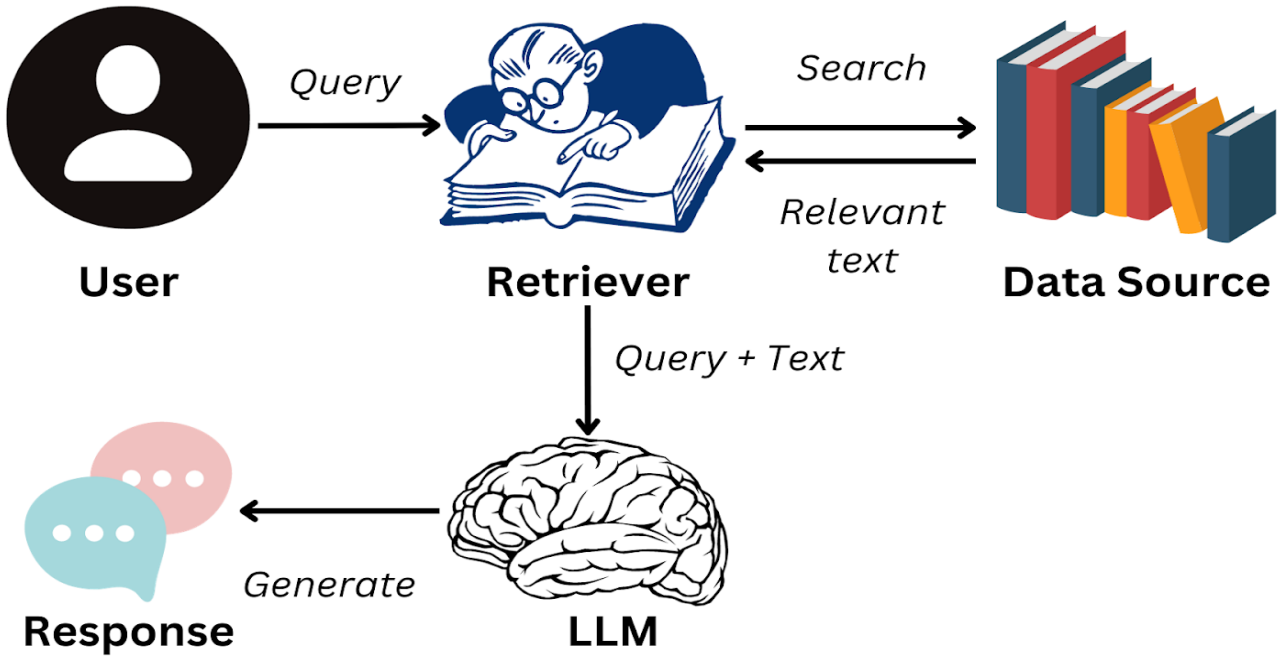
Instructor: Linqi Song

Content

- What’s RAG?
- What / When / How to retrieve?

What’s RAG?

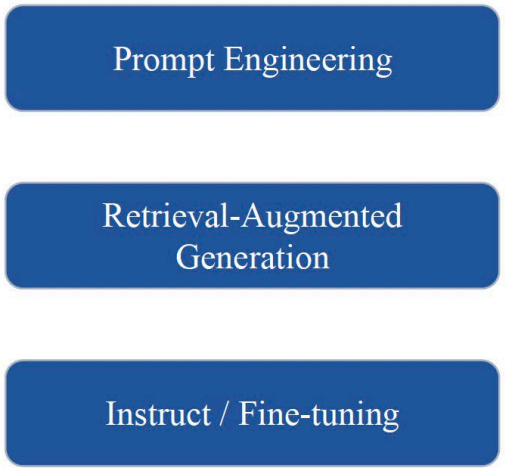
- RAG is a paradigm that enhances LLMs by integrating external knowledge bases.
- It employs a synergistic approach, combining information retrieval mechanisms and generation techniques to bolster the NLP performance.

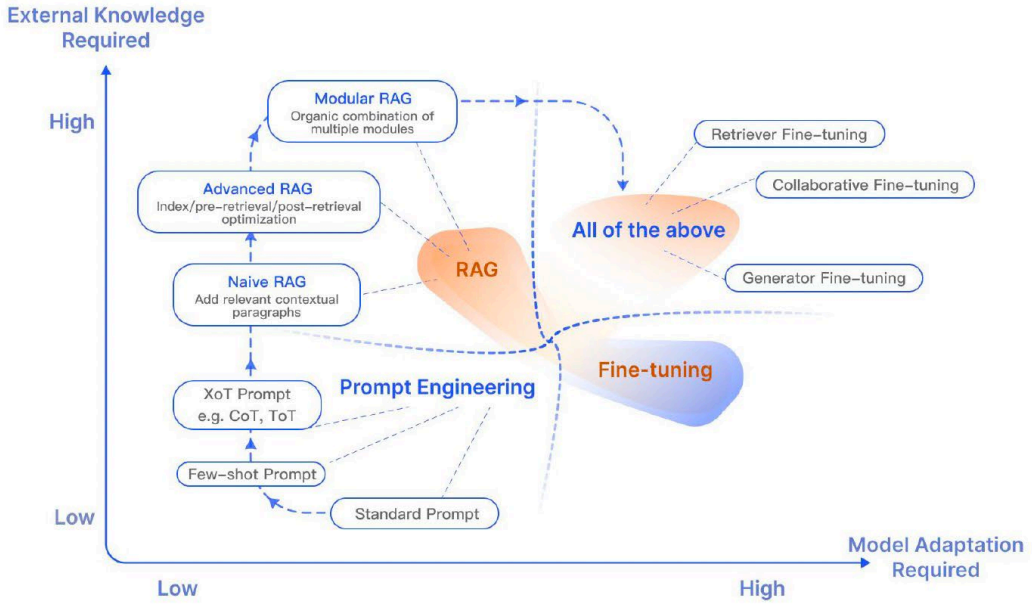


Why RAG?

- Tackle with up-to-date and long-tail knowledge
  - broaden LLM's ability
- No retraining for task-specific/domain-specific applications
  - the high costs associated with training and inference
- Reduce Hallucination
  - mitigate this problem by grounding generated content, ensuring the responses it generates are well-grounded

Ways to optimize LLMs.





A typical case of RAG

Where is RAG being used?

Scenarios where RAG is applicable:

- Long-tail distribution of data
- Frequent knowledge updates
- Answers requiring verification and traceability
- Specialized domain knowledge
- Data privacy preservation

Q&A
RETRO (Borgeaud et al2021)
REALM (Gu et al, 2020)
ATLAS (lzacard et al, 2023)

Fact Checking
RAG (Lewis et al, 2020)
ATLAS (lzacard et al, 2022)
Evi. Generator (Asai et al, 2022a)

Dialog
BlenderBot3 (Shuster et al.2022)
Internet-augmented generation
(Komeili et a., 2022)

Summary
FLARE (Jiang et al, 2023)

Machine Translation
kNN-MT (Khandelwal et al., 2020)TRIME-MT (Zhong et al., 2022)

Code Generation
DocPrompting (Zhou et al., 2023
Natural ProverWelleck et al., 2022)

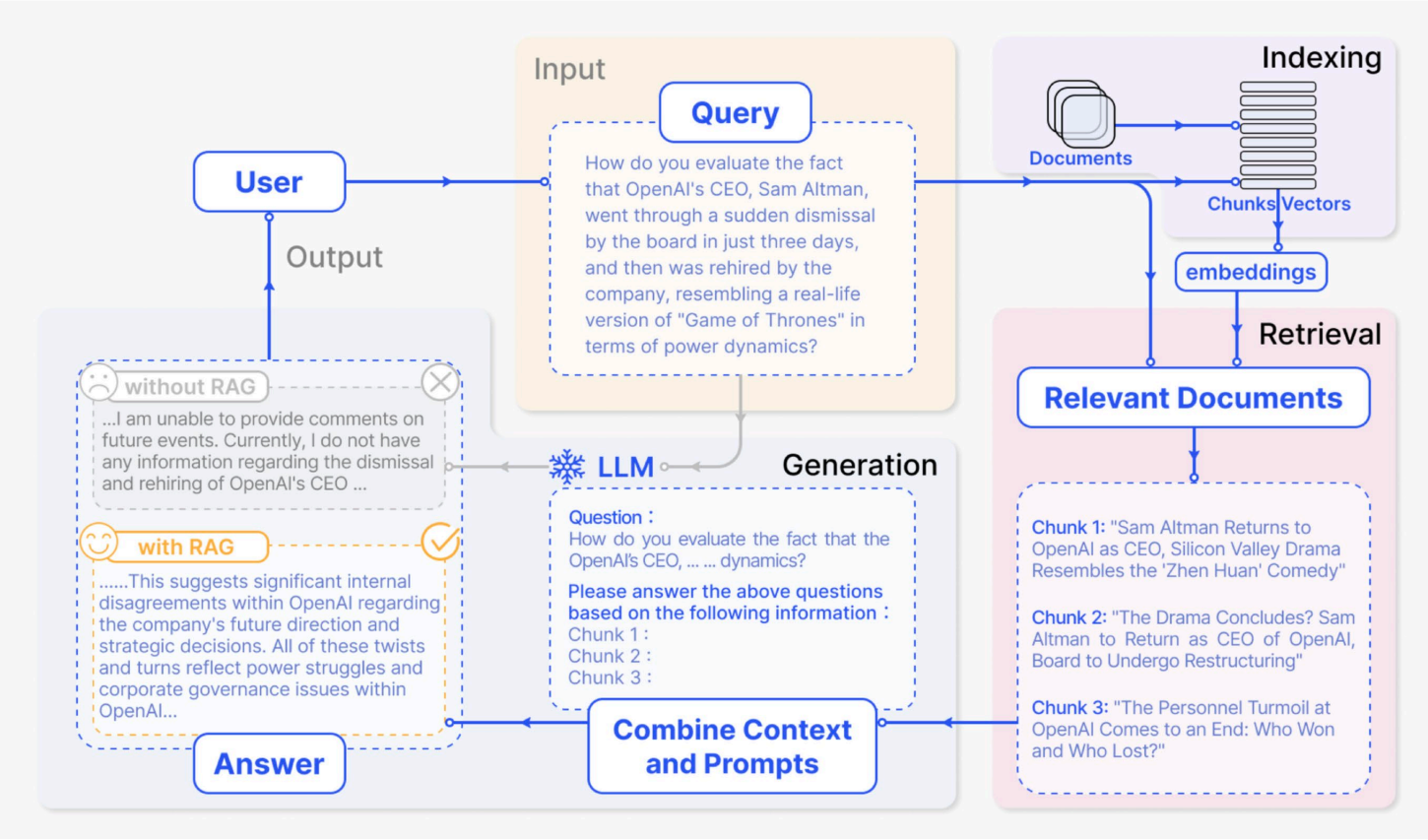
Natural Language Inference
kNN-Prompt (Shi et al., 2022)
NPM (Min et al., 2023)

Sentiment analysis
kNN-Prompt (Shi et al., 2022)NPM (Min et al., 2023)

Commonsense reasoning
Raco (Yu et al, 2022)

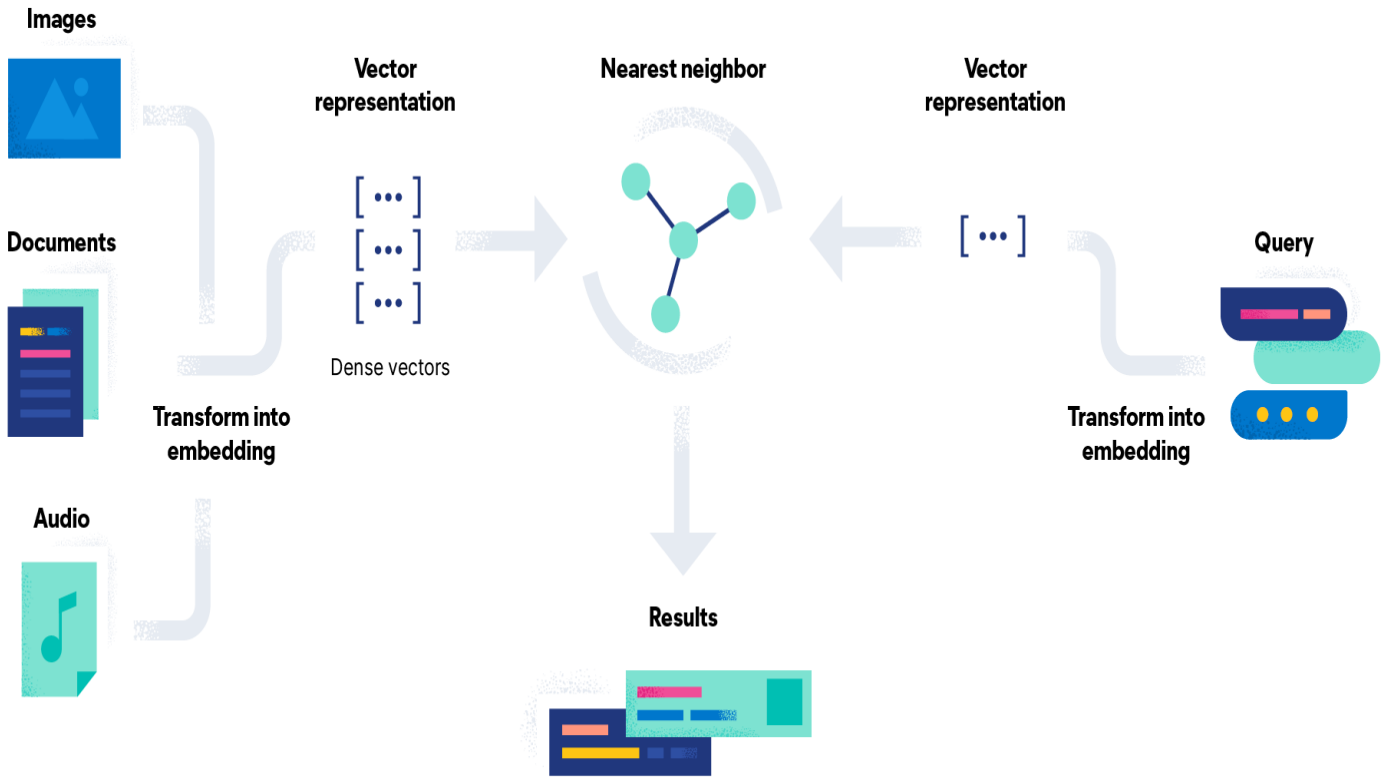
RAG architecture

- Vector Database
  - Embedding
  - Indexing
  - Querying (Retrieve)
  - Post-process
- Generator
  - LLM (like GPT, BART, T5)



Vecctor Database

- A vector database is a database that stores information as vectors, which are numerical representations of data objects, also known as vector embeddings.
- Vector embeddings are a numerical representation of a subject, word, image, or any other piece of data. Vector embeddings — also known as embeddings — are generated by large language models and other AI models.
- A vector database is different from a vector search library or vector index: it is a data management solution that enables metadata storage and filtering, is scalable, allows for dynamic data changes, performs backups, and offers security features.
- A vector database works by using algorithms to index and query vector embeddings.

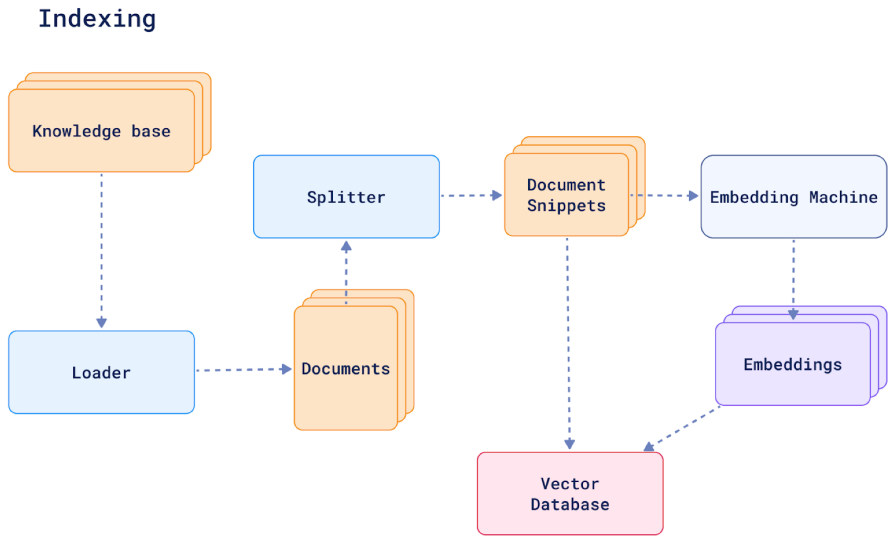


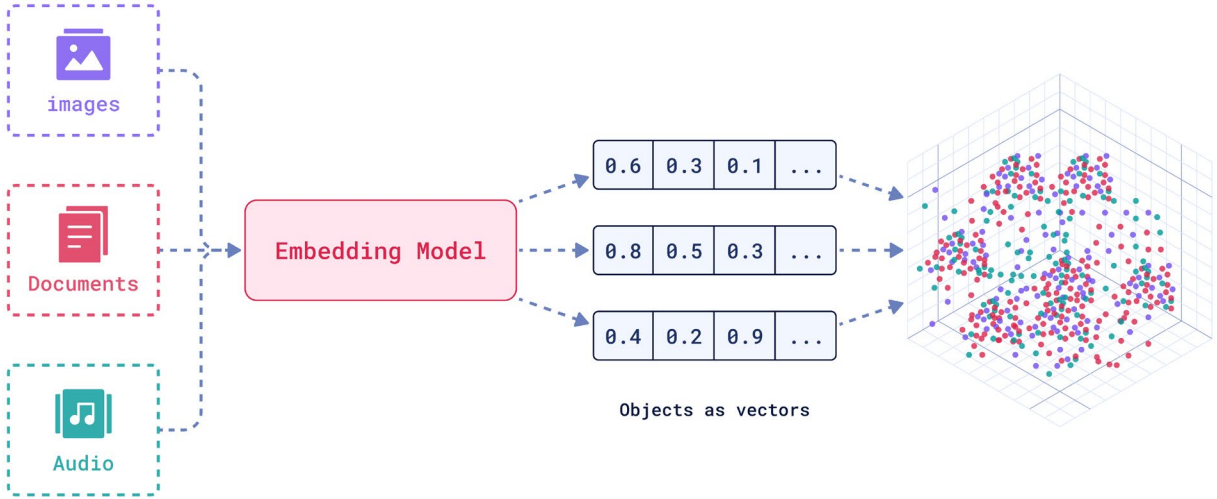
Vector database-embedding

Start with a loader that gathers documents containing your data. These documents could be anything from articles and books to web pages and social media posts.

Next, a splitter divides the documents into smaller chunks, typically sentences or paragraphs. This is because RAG models work better with smaller pieces of text. In the diagram, these are document snippets.

Each text chunk is then fed into an embedding machine (TF-IDF, BM25, BERT, GPT). This machine uses complex algorithms to convert the text into vector embeddings.





Vector database-indexing

- All the generated vector embeddings are stored in a knowledge base of indexed information. This supports efficient retrieval of similar pieces of information when needed.
  - Hashing: A hashing algorithm, such as the locality-sensitive hashing (LSH) algorithm, is best suited to an approximate nearest neighbor search because it enables speedy results, and generates approximate results. LSH uses hash tables — think of a Sudoku puzzle — to map nearest neighbors. A query will be hashed into a table and then compared to a set of vectors in the same table to determine similarity.
  - Quantization: A quantization technique, such as product quantization (PQ), will break up vectors into smaller parts and represent those parts with code, and then put the parts back together. The result is a code representation of a vector and its components. The ensemble of these codes is referred to as a codebook. When queried, a vector database that uses quantization will break the query down into code, and then match it against the codebook to find the most similar code to generate results.
  - Graph-based: A graph algorithm, such as the Hierarchical Navigable Small World (HNSW) algorithm uses nodes to represent vectors. It clusters the nodes and draws lines or edges between similar nodes, creating hierarchical graphs. When a query is launched, the algorithm will navigate the graph hierarchy to find nodes containing the vectors that are most similar to the query vector.

Indexing optimization

Chunk Optimization

Small-2-Big

Embeding at sentence level expand the window during generation process.

Slidingwindow

liding chunk covers the entire text, avoiding semantic ambiguity

Summary

Retrieve documents through summaries, then retrieve text blocks from the documents.

Adding Metadata

Example | Page | Time | Type | Document Title

Metadata Filtering/Enrichment

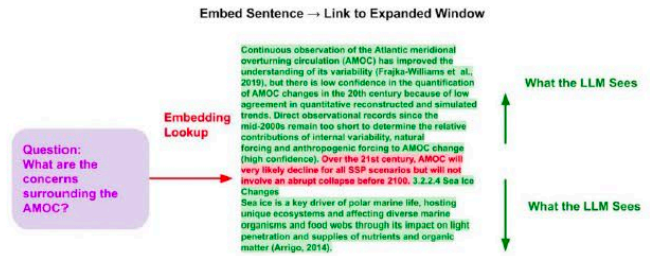
Pseudo Metadata Generation

Enhance retrieval by gener-ating a hypothetical document for the incoming query and creating qu-estions that the text block can answer.

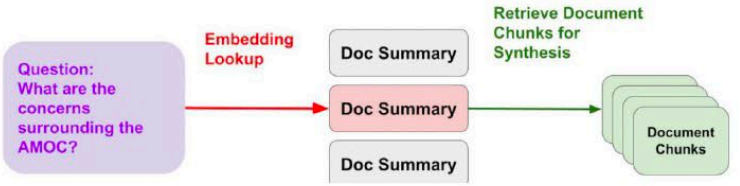
Metadata filter

Dissect and annotate the document. During the query, infer metadata filters in addition to semantic queries

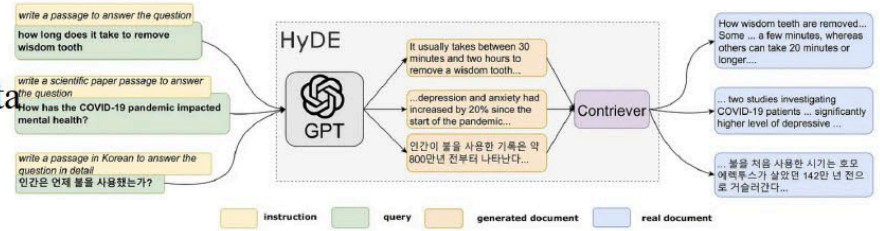
Small-2-Big



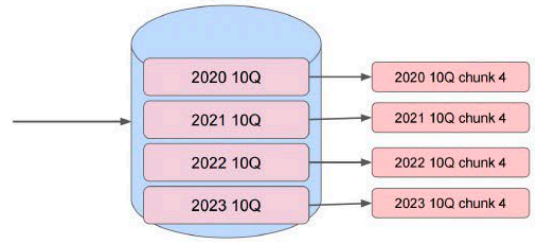
Abstract



Pseudo Metadata



Metadata filter



Naïve RAG

Step1 Indexing

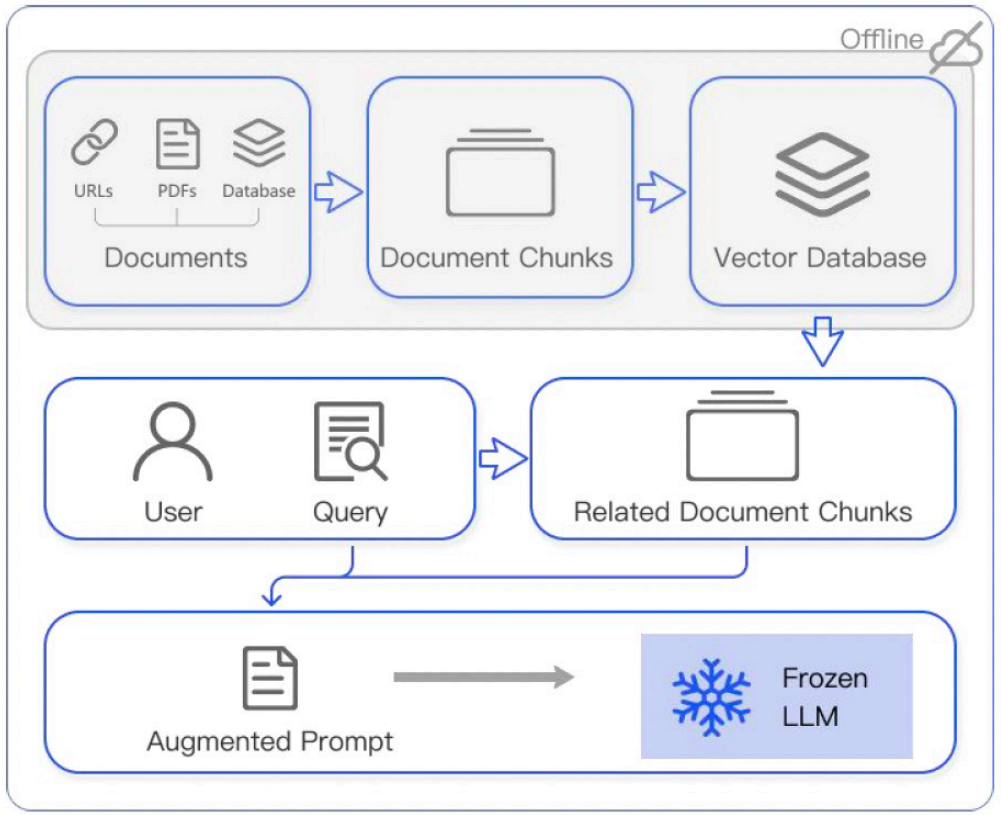
1. Divide the document into even chunks, each chunk being a piece of the original text.
2. Using the encoding model to generate an embedding for each chunck.
3. Store the Embedding of each block in the vector database.

Step2 Retrival

Retrieve the k most relevant documents using vector similarity search.

Step3 Generation

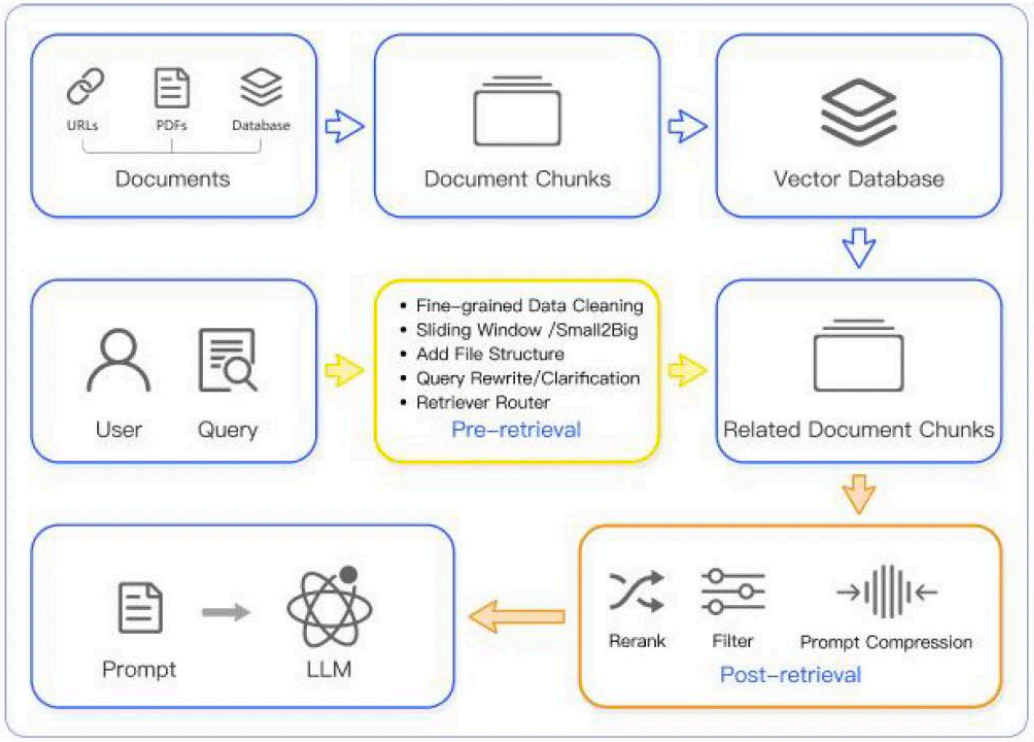
The original query and the retrieved text are combined and input into a LLM to get the final answer



Advanced RAG

Index Optimization → Pre-Retrieval Process → Retrieval → Post-Retrieval Process→ Genaration

- **Optimizing Data Indexing:** sliding window, fine-grained segmentation、adding metadata
- **Pre-Retrieval Process:** retrieve routes, summaries, rewriting, and confidence judgment
- **Post-Retrieval Process:** reorder, filter content retrieval



Main issues in RAG – what/when/how

What to retrieve ?

- Token
- Phrase
- Chunk
- Paragraph
- Entity
- Knowledge graph

When to retrive ?

- Single search
- Each token
- Every N tokens
- Adaptive search

How to use the retrieved information ?

- Input/Data Layer
- Model/Intermediate Layer
- Output/Prediction Layer

Other Issues

Augmentation stage:

- Pre-training
- Fine-tuning
- Inference

Retrieval choice:

- BERT
- Roberta
- BGE
- .....

Model Collaboration

Scale selectionz

Generation choice:

- GPT
- Llama
- T5
- .....

What to retrieve?

coarse

Retrieval granularity

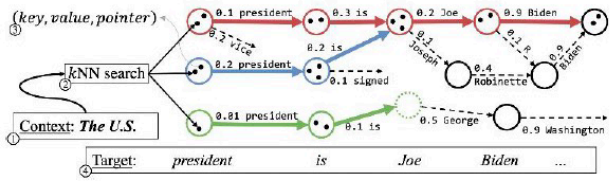
meticulous

low

level of structuration

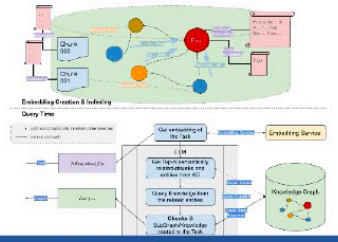
High

Chunk | In-Context RAG 2023



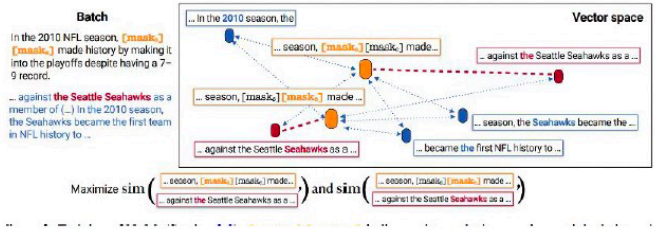
The search is broad, recalling a large amount of information, but with low accuracy, high coverage but includes much redundant information.

Knowledge Graph | 2023

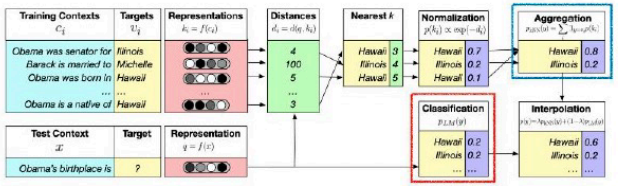


Richer semantic and structured information, but the retrieval efficiency is lower and is limited by the quality of KG.

Phrase | NPM 2023

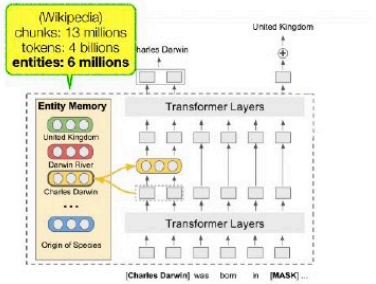


Token | KNN-LMM 2019



It excels in handling long-tail and cross-domain issues with high computational efficiency, but it requires significant storage.

Entity | EasE 2022



When to retrieve?

High efficiency, but low relevance of the retrieved documents

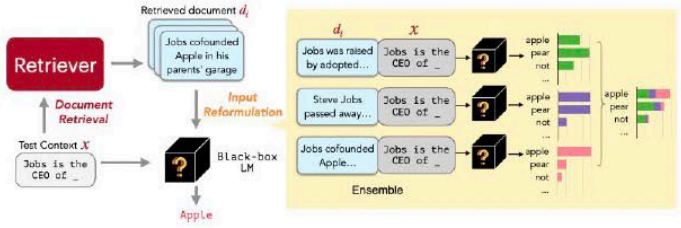
Balancing efficiency and information might not yield the optimal solution

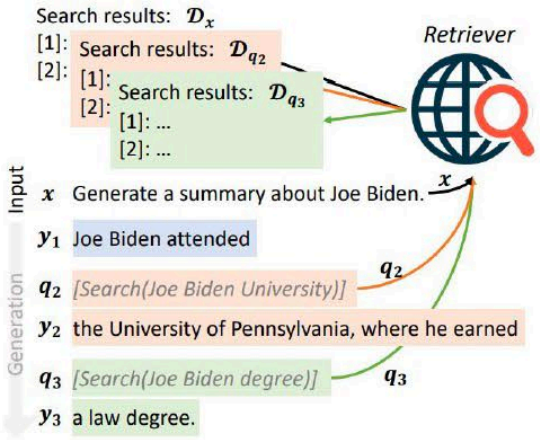
A large amount of information with low efficiency and redundant information.

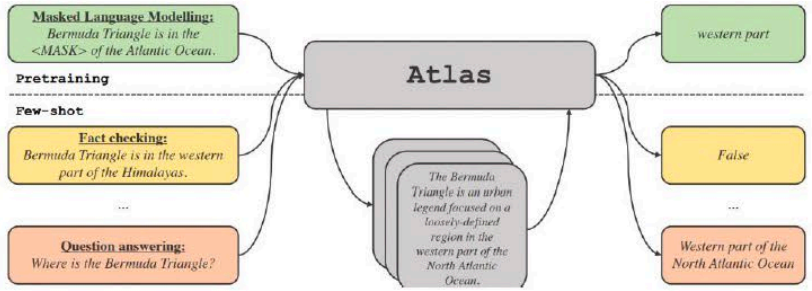
Once | Replug 2023

Adaptive | Flare 2023

Every N Tokens | Atlas 2023







Conducting once search during the reasoning process.

Adaptively conduct the search.

Retrieve once for every N tokens generated.

Low

Retrieval frequency

High

Retrieval with tokens - Toolformer

- Toolformer (Schick et al. 2023) generates tokens that trigger retrieval (or other tools)
- Training is done in an iterative manner - generate and identify successful retrievals

The New England Journal of Medicine is a registered trademark of **[QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society]** the MMS.

Out of 1400 participants, 400 (or **[Calculator(400 / 1400) → 0.29]** 29%) passed the test.

The name derives from "la tortuga", the Spanish word for **[MT("tortuga") → turtle]** turtle.

The Brown Act is California's law **[WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.]** that requires legislative bodies, like city councils, to hold their meetings open to the public.

Retrieval with uncertainty

- FLARE (Jiang et al. 2023) tries to generate content, then does retrieval if LM certainty is low.

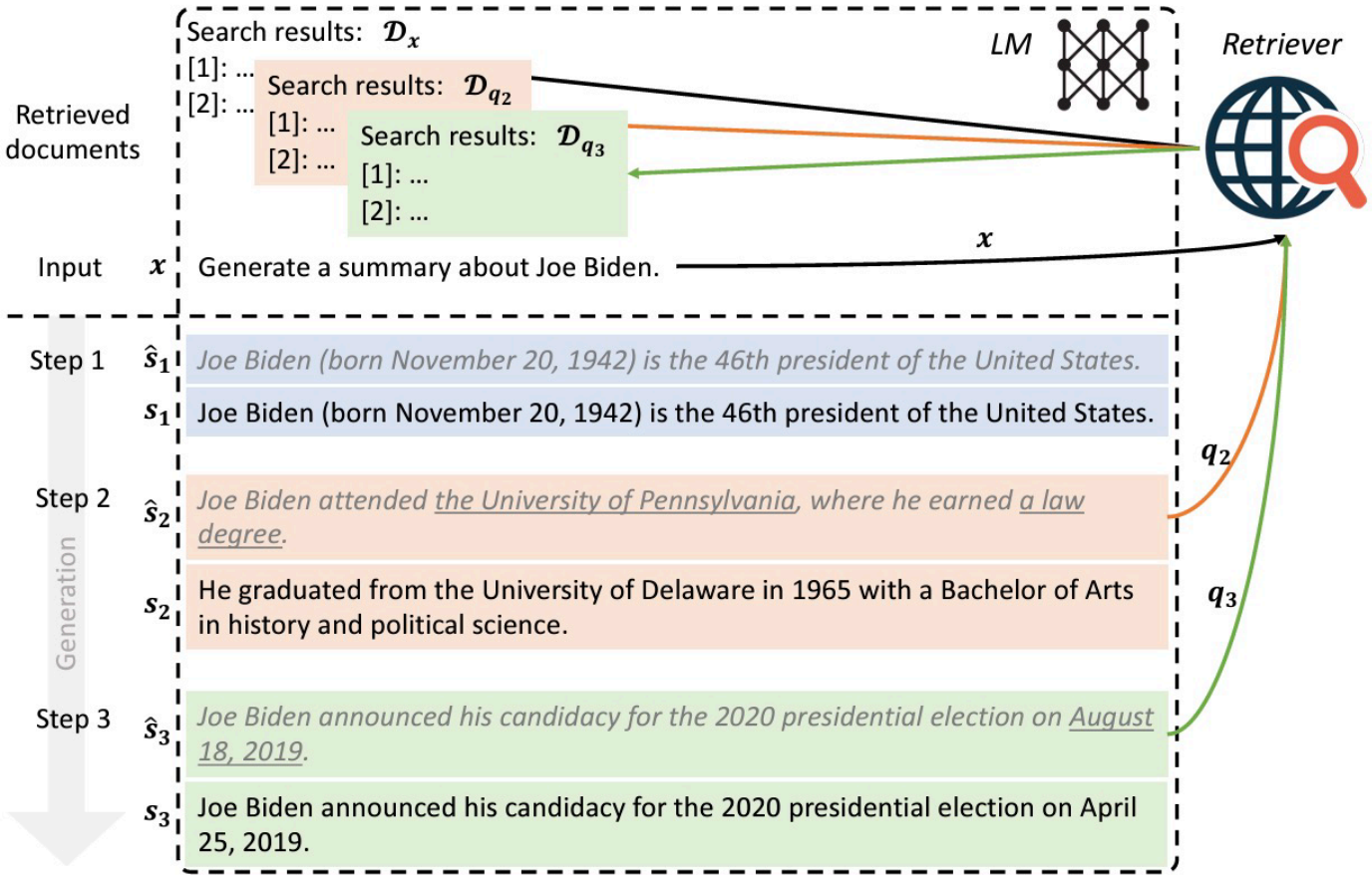


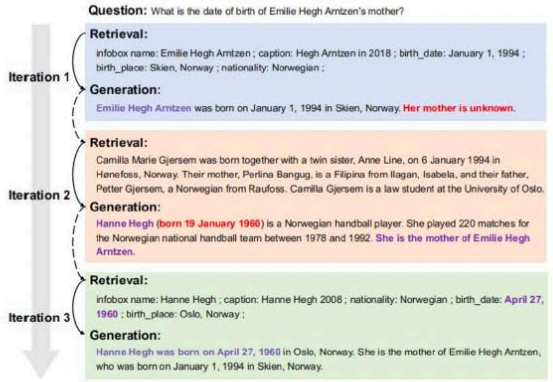
Figure 1: An illustration of forward-looking active retrieval augmented generation (FLARE). Starting with the user input $\boldsymbol{x}$ and initial retrieval results $\mathcal{D}_{\boldsymbol{x}}$, FLARE iteratively generates a temporary next sentence (shown in *gray italic*) and check whether it contains low-probability tokens (indicated with underline). If so (step 2 and 3), the system retrieves relevant documents and regenerates the sentence.

Iterative retrieval and adaptive retrieval

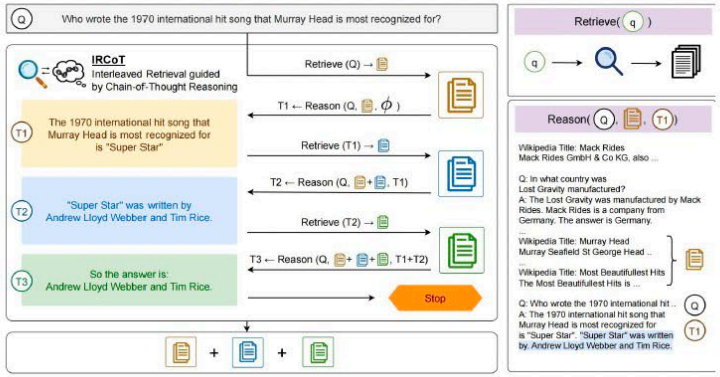
Iterative

Iteratively Retrieving from the Corpus to Acquire More Detailed and In-depth Knowledge

ITER [Feng et al., 2023]



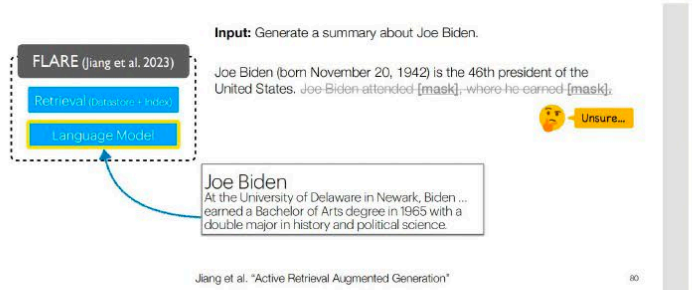
IRCOT [Trivedi et al.,2022]



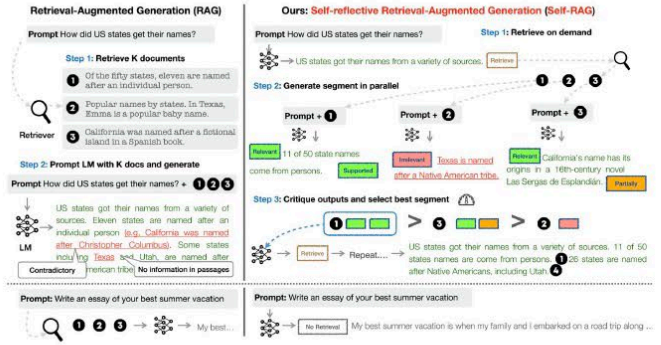
Adaptive

Dynamically Determined by the LLM, the Timing and Scope of Retrieva

FLARE [Jiang et al., 2023]



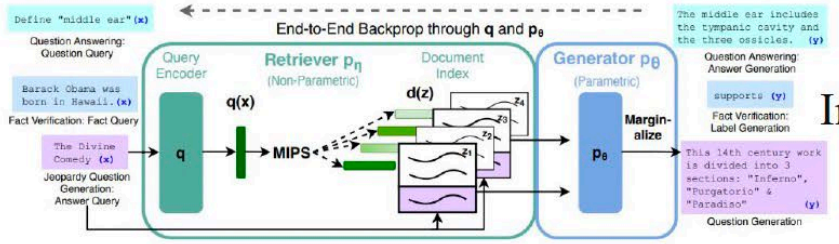
Self-RAG [Asai et al., 2023]



How to use?

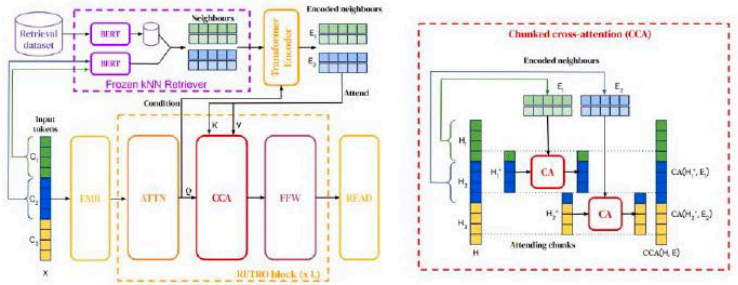
Integrating the retrieved information into different layers of the generation model,during inference process.

Integrate retrieval positions.



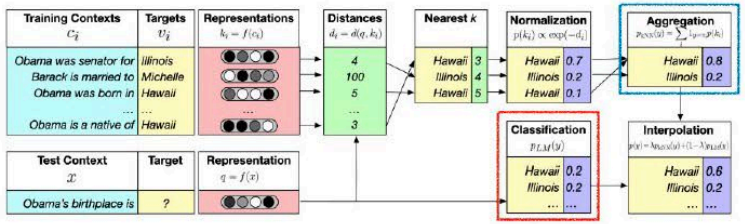
Input / Date layer

Using simple, but unable to support the retrieval of more knowledge blocks, and the optimization space is limited.



Model / Interlayer

Supports the retrieval of more knowledge blocks, but introduces additional complexity and must be trained.

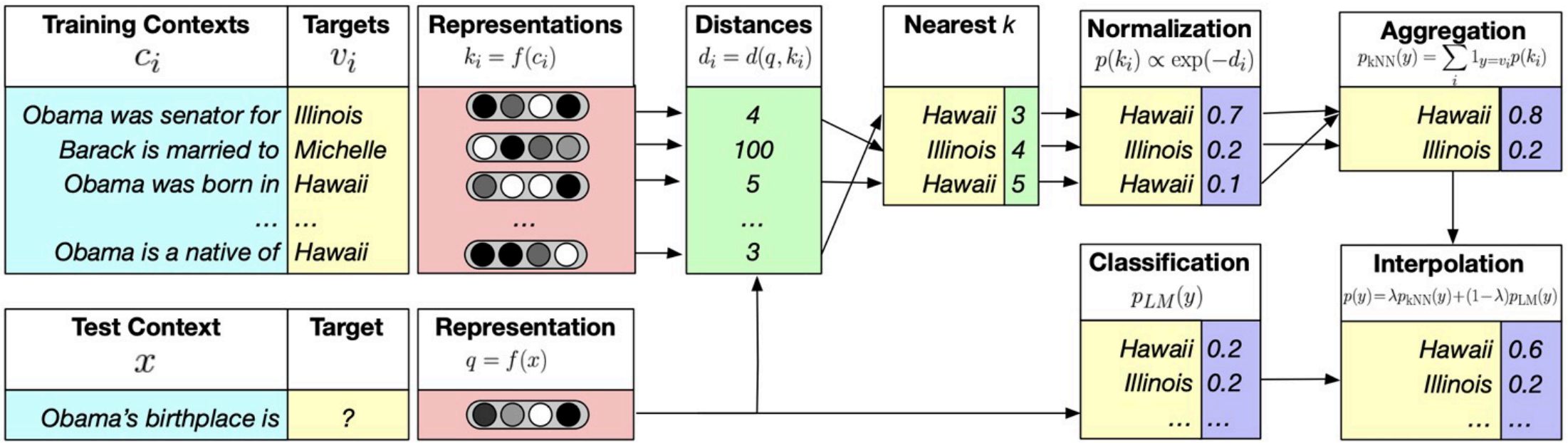


Output /Prediction layer

Ensuring the output results are highly relevant to the retrieval content, but the efficiency is low.

Example

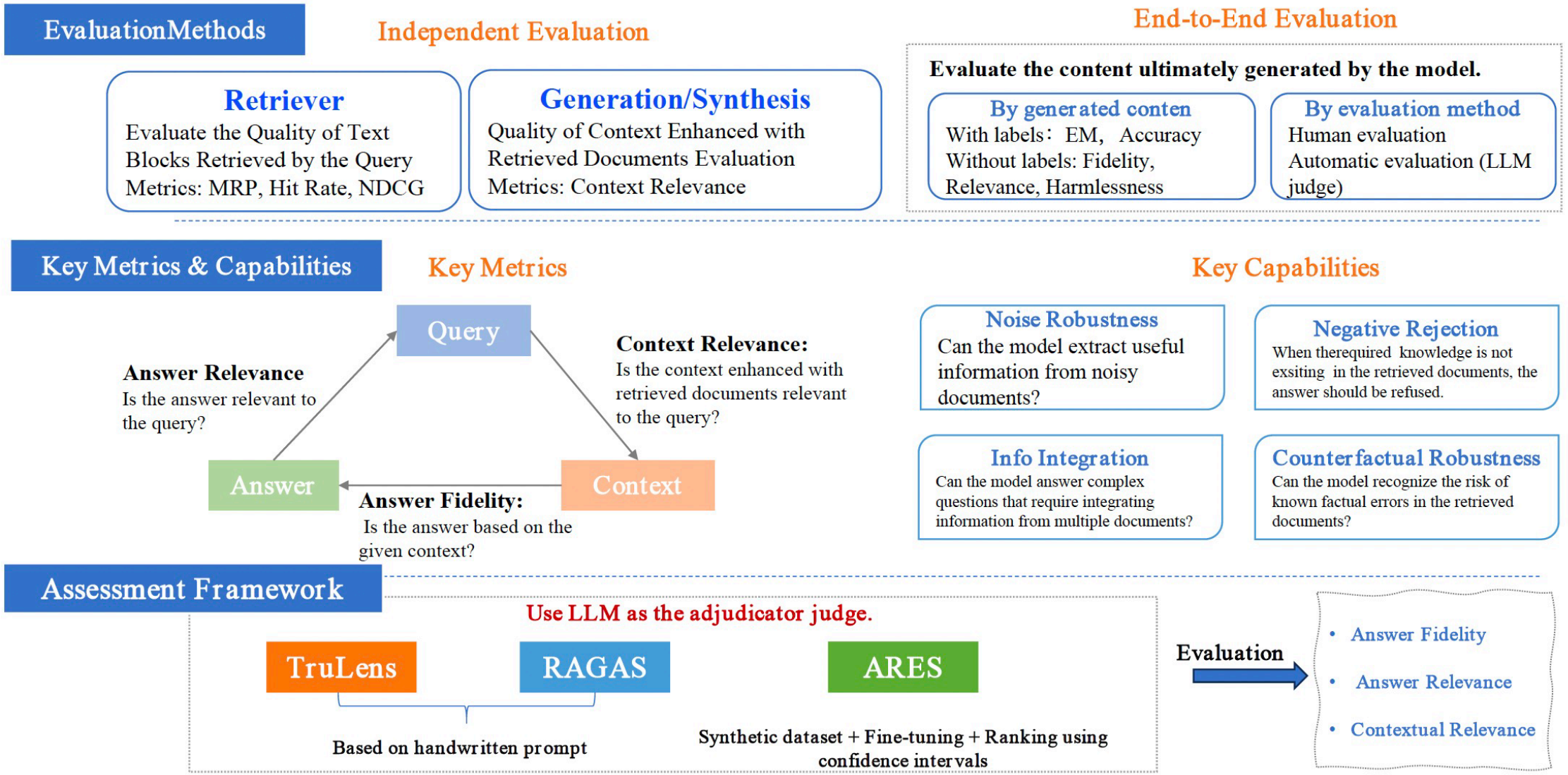
- kNN-LM (Khandelwal et al. 2019) retrieves similar examples, and uses the following token from them



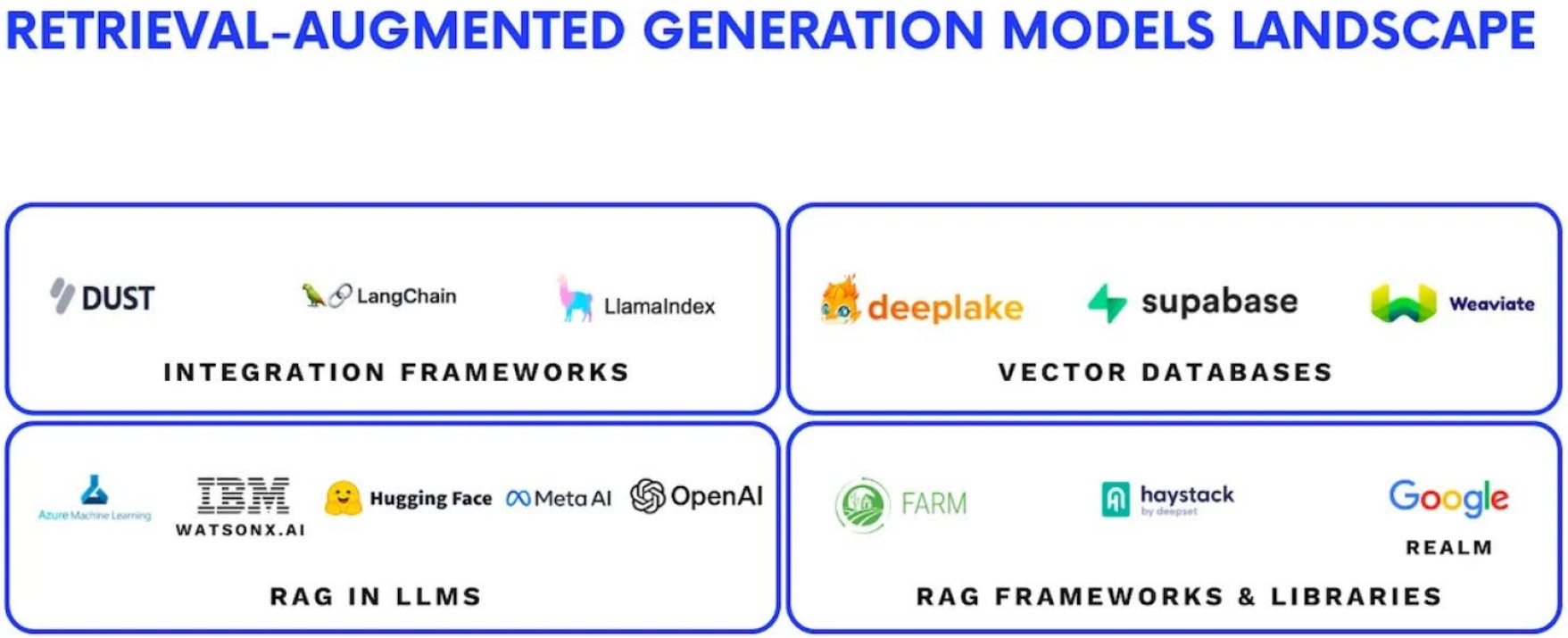
What to,When to and How to use Retrieval

	What do retrieve?	How to use retrieval?	When to retrieve?
REALM (Guu et al 2020)	Text chunks	Input layer	Once
Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)	Text chunks	Input layer	Every n tokens
RETRO (Borgeaud et al. 2021)	Text chunks	Intermediate layers	Every n tokens
kNN-LM (Khandelwal et al. 2020)	Tokens	Output layer	Every token
FLARE (Jiang et al. 2023)	Text chunks	Input layer	Every n tokens *(adaptive)*
Adaptive kNN-LM (He et al 2021, Alon et al 2022, etc)	Tokens	Output layer	Every n tokens *(adaptive)*
Entities as Experts (Fevry et al. 2020), Mention Memory (de Jong et al. 2022)	Entities or entity mentions	Intermediate layers	Every entity mentions
Wu et al. 2022, Bertsch et al. 2023, Rubin & Berant. 2023	Text chunks ***from the input***	Intermediate layers	Once or every n tokens

Evaluation


EvaluationMethods
Independent Evaluation
Retriever
Evaluate the Quality of Text Blocks Retrieved by the Query
Metrics: MRP, Hit Rate, NDCG
Generation/Synthesis
Quality of Context Enhanced with Retrieved Documents Evaluation
Metrics: Context Relevance
End-to-End Evaluation
Evaluate the content ultimately generated by the model.
By generated conten
With labels： EM， Accuracy
Without labels: Fidelity, Relevance, Harmlessness
By evaluation method
Human evaluation
Automatic evaluation (LLM judge)
Key Metrics & Capabilities
Key Metrics
Query
Answer Relevance
Is the answer relevant to the query?
Context Relevance:
Is the context enhanced with retrieved documents relevant to the query?
Answer
Context
Answer Fidelity:
Is the answer based on the given context?
Key Capabilities
Noise Robustness
Can the model extract useful information from noisy documents?
Negative Rejection
When therequired knowledge is not exsiting in the retrieved documents, the answer should be refused.
Info Integration
Can the model answer complex questions that require integrating information from multiple documents?
Counterfactual Robustness
Can the model recognize the risk of known factual errors in the retrieved documents?
Assessment Framework
Use LLM as the adjudicator judge.
TruLens
RAGAS
ARES
Based on handwritten prompt
Synthetic dataset + Fine-tuning + Ranking using confidence intervals
Evaluation
Answer Fidelity
Answer Relevance
Contextual Relevance

RAG Tools



AIMultiple

References

- [1] Guu, Kelvin, et al. "Retrieval augmented language model pre-training." International conference on machine learning. PMLR, 2020.
- [2] Ram, Ori, et al. "In-context retrieval-augmented language models." Transactions of the Association for Computational Linguistics 11 (2023): 1316-1331.
- [3] Shi, Weijia, et al. "Replug: Retrieval-augmented black-box language models." arXiv preprint arXiv:2301.12652 (2023).
- [4] Borgeaud, Sebastian, et al. "Improving language models by retrieving from trillions of tokens." International conference on machine learning. PMLR, 2022.
- [5] Khandelwal, Urvashi, et al. "Generalization through memorization: Nearest neighbor language models." arXiv preprint arXiv:1911.00172 (2019).
- [6] Jiang, Zhengbao, et al. "Active retrieval augmented generation." arXiv preprint arXiv:2305.06983 (2023).
- [7] Subramaniyaswamy, V., and R. Logesh. "Adaptive KNN based recommender system through mining of user preferences." Wireless Personal Communications 97 (2017): 2229-2247.
- [8] Févry, Thibault, et al. "Entities as experts: Sparse memory access with entity supervision." arXiv preprint arXiv:2004.07202 (2020).
- [9] De Jong, Michiel, et al. "Mention memory: incorporating textual knowledge into transformers through entity mention attention." arXiv preprint arXiv:2110.06176 (2021).
- [10] Gao, Yunfan, et al. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 (2023).
- https://qdrant.tech/articles/what-is-rag-in-ai/#
- https://www.llamaindex.ai/blog
- https://qdrant.tech/articles/what-are-embeddings/
- https://www.databricks.com/glossary/retrieval-augmented-generation-rag
- https://www.elastic.co/what-is/vector-database