Chat with your own data: A hands-on introduction to Retrieval Augmented Generation(RAG) by IEEE AUTH SB

ARISTOTLE UNIVERSITY OF THESSALONIKI IEEE STUDENT BRANCH

Michail Dadopoulos
25/05/2025

What are Large Language Models

- Deep learning models trained on a huge amount of data to understand and generate human-like text.

- Tuned into useful assistants (ChatGPT, Gemini, Claude)

- These assistants can then help with a broad range of tasks (text generation, summarization, translation, question answering,e.t.c..)

Limitations of LLMs

- Prone to hallucinations
- Stale Knowledge
- Attribution Problem
- Lack of Domain-Specific / Private Knowledge

Approaches to add new knowledge in LLMs

Fine-tuning:

- How it works: Updating the weights of a pre-trained LLM by training it on a new, specific dataset. This is referred to as a parametric approach.

- Purpose: To inject new, relatively stable knowledge or adapt the model's style/behavior.

- Limitations: Expensive & Time-Consuming, Still Stale (eventually), Poor for Specific Point Queries from large data.

Approaches to add new knowledge in LLMs

In-Context Learning (Prompting):

- How it works: Putting relevant information directly into the prompt alongside the user's query. The model learns from this context without changing its underlying weights. This is referred to as a semi parametric approach.

- Purpose: To provide immediate context for a specific query or task.

- Major Limitation: Strictly bounded by Context Window: Cannot handle large documents or entire knowledge bases.

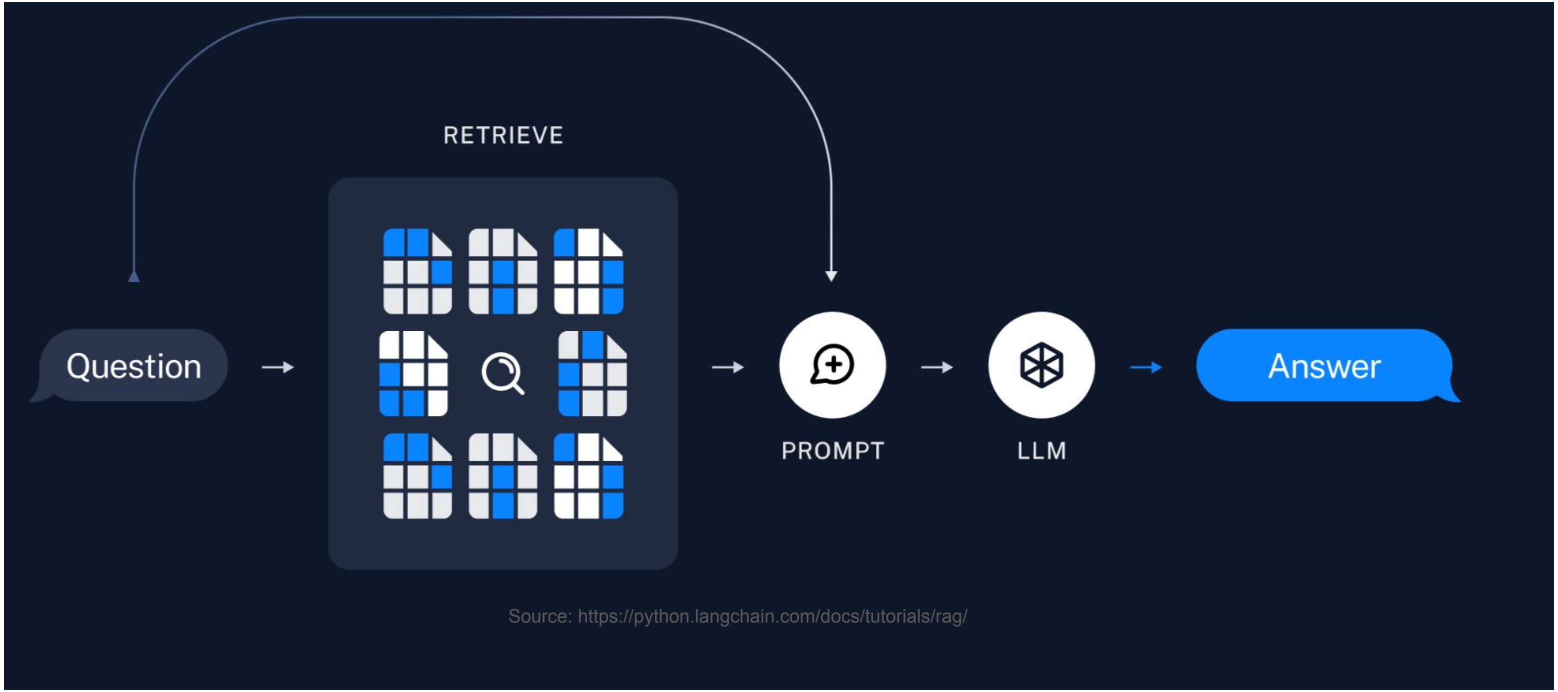
THE SOLUTION?

Retrieval-Augmented Generation (RAG)

RAG is a technique that enhances large language models by retrieving relevant external or domain-specific data to provide context, improving the quality and accuracy of generated responses.

Allows LLMs to "chat with your own data," providing grounded, accurate, and context-aware answers, overcoming the limitations discussed.

Simple RAG pipeline



Core Components of a RAG System

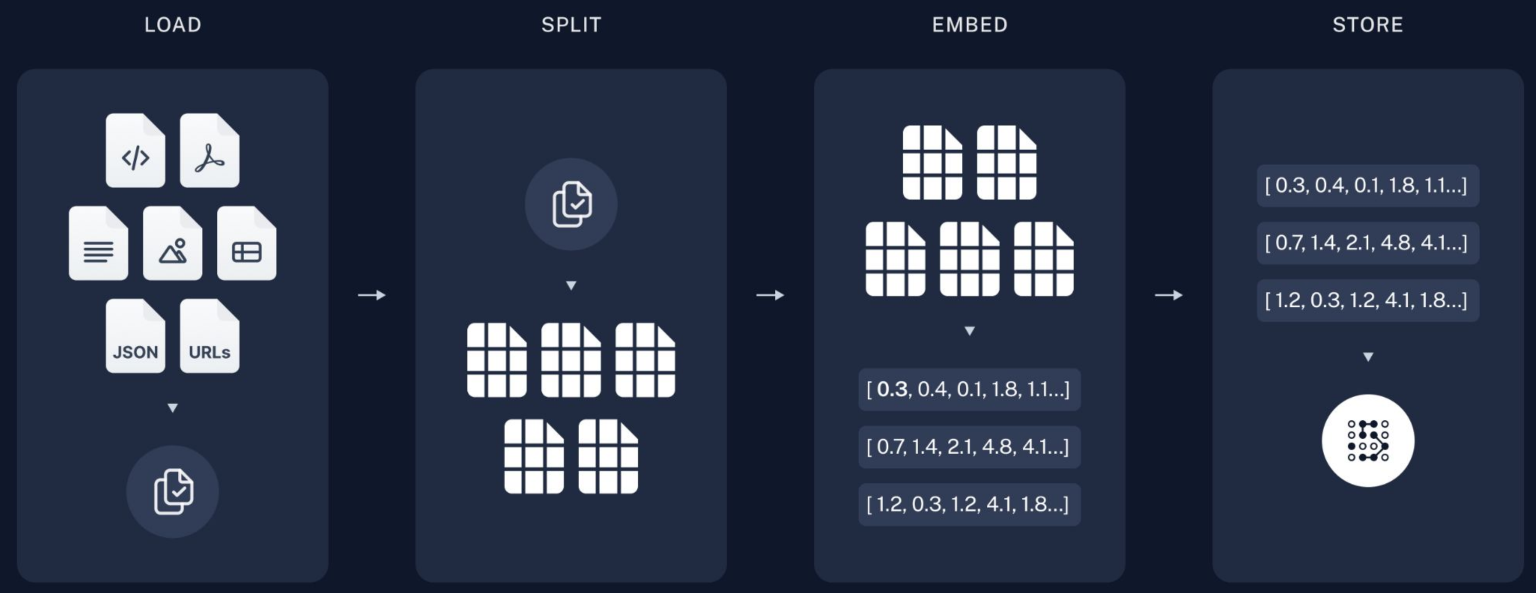
1. Indexing

2. Retrieval

3. Generation

Indexing steps

1. Loading: Reading data from sources.
2. Splitting/Chunking: Breaking down large documents into smaller pieces (chunks) to fit within the LLM's context window.
3. Embedding: Converting text chunks into numerical vector representations (embeddings) that capture semantics and relationships between words, using an embedding model.
4. Storing: Saving the chunks and their embeddings in a Vector Database, so that they can be searched over later.



Source : https://python.langchain.com/docs/tutorials/rag/

Retrieval

The Retriever:

1. Takes the user's query.

2. Converts the query into an embedding using the same embedding model.

3. Searches the Vector Database to find the top K most semantically similar chunks.

Generation

The Generator(LLM):

1. Receives the original user query and the retrieved relevant chunks.

2. Uses a carefully crafted prompt that includes the query and the retrieved context.

3. Generates the final answer based on the provided context.

Coding time

https://github.com/MDadopoulos/IEET_CON_2025_RAG_workshop

Advanced RAG Techniques

Query Processing / Understanding:

- Query Expansion
- Query Rephrasing
- Intent Detection / Router

Advanced RAG Techniques

Indexing Techniques:

- Advanced parsing
- Advanced Chunking Strategies
- Adding Metadata and using it in filtering
- Building Hierarchical or Graph-based Indexes

Advanced RAG Techniques

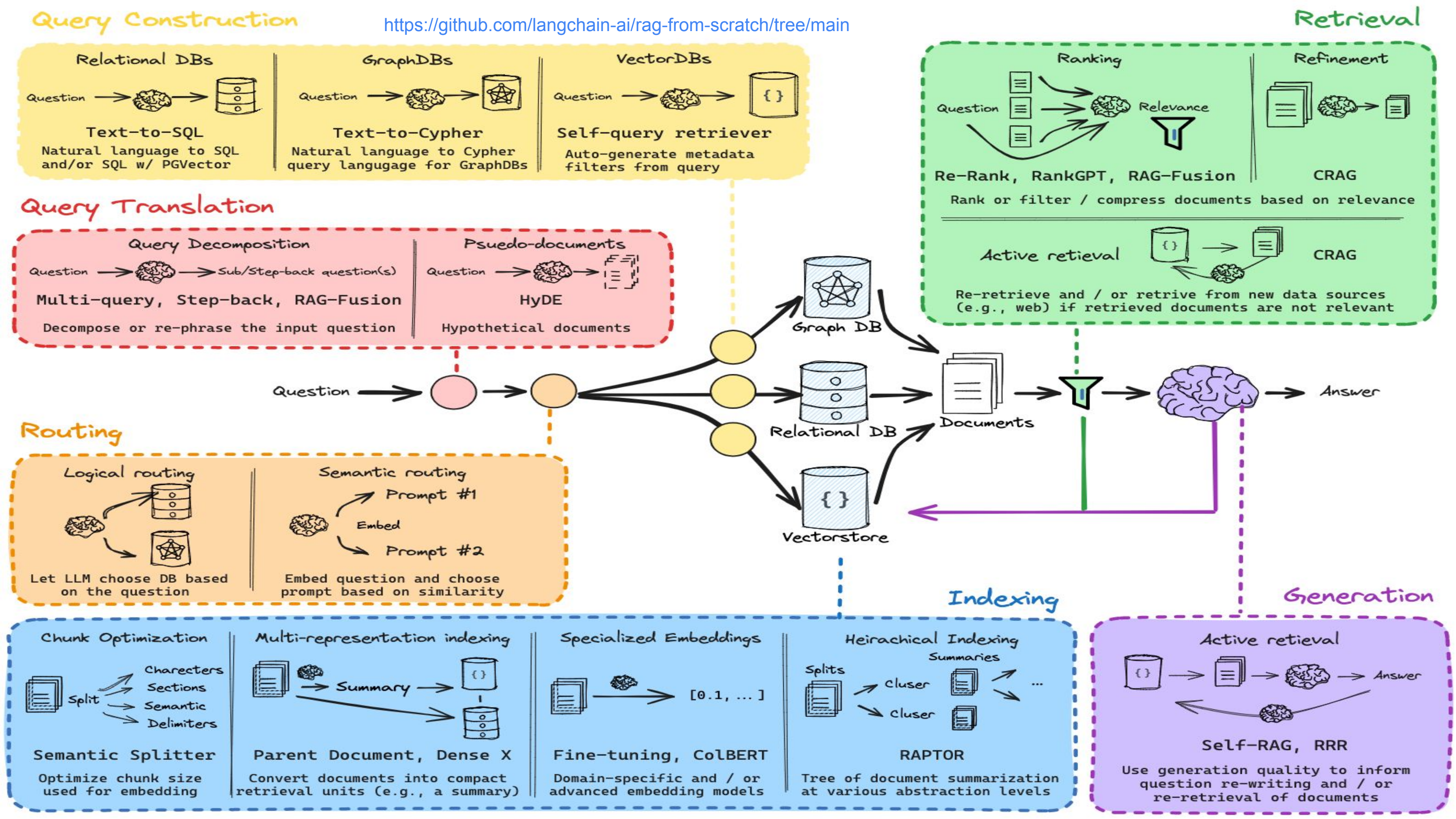
Retrieval Techniques:

- Hybrid Search
- Reranking
- Multi-hop Retrieval
- Contextual Compression/filtering

Advanced RAG Techniques

Generation Techniques:

- Better Prompt Engineering
- Structured Output
- Self-Correction/Refinement


https://github.com/langchain-ai/rag-from-scratch/tree/main
Query Construction
Relational DBs
Question
Text-to-SQL
Natural language to SQL and/or SQL w/ PGVector
GraphDBs
Question
Text-to-Cypher
Natural language to Cypher query langugage for GraphDBs
VectorDBs
Question
{ }
Self-query retriever
Auto-generate metadata filters from query
Query Translation
Query Decomposition
Question
Sub/Step-back question(s)
Multi-query, Step-back, RAG-Fusion
Decompose or re-phrase the input question
Psuedo-documents
Question
HyDE
Hypothetical documents
Routing
Logical routing
Let LLM choose DB based on the question
Semantic routing
Prompt #1
Embed
Prompt #2
Embed question and choose prompt based on similarity
Question
Graph DB
Relational DB
{ }
Vectorstore
Documents
Answer
Retrieval
Ranking
Question
Relevance
Re-Rank, RankGPT, RAG-Fusion
Refinement
CRAG
Rank or filter / compress documents based on relevance
Active retrieval
{ }
CRAG
Re-retrieve and / or retrive from new data sources (e.g., web) if retrieved documents are not relevant
Indexing
Chunk Optimization
Split
Charecters
Sections
Semantic
Delimiters
Semantic Splitter
Optimize chunk size used for embedding
Multi-representation indexing
Summary
{ }
Parent Document, Dense X
Convert documents into compact retrieval units (e.g., a summary)
Specialized Embeddings
[0.1, ...]
Fine-tuning, ColBERT
Domain-specific and / or advanced embedding models
Heirachical Indexing
Summaries
Splits
Cluser
Cluser
...
RAPTOR
Tree of document summarization at various abstraction levels
Generation
Active retrieval
{ }
Answer
Self-RAG, RRR
Use generation quality to inform question re-writing and / or re-retrieval of documents

Agentic RAG

The LLM acts as an agent which:

- Is capable of step-by-step planning and decision-making
- Can use tools (retrieval, calling other APIs)
- Can iterate, refine its answer or perform further actions based on tool results
- Have memory

Evaluating RAG Systems

- Context relevance
- Context recall
- Faithfulness
- Answer relevancy

Productionizing RAG: Key Considerations

- Latency, cost, scalability
- Data freshness and updates
- Privacy, security, and compliance
- Monitoring and feedback loops

Real-World Applications

- Enterprise Search & Internal Knowledge Bases
- Customer Support Chatbots
- Research & Analysis Assistants
- Legal Document Analysis …

Useful Resources

Rag basic components and advanced techniques:

- https://github.com/labdmitriy/llm-rag?tab=readme-ov-file
- https://github.com/langchain-ai/rag-from-scratch/tree/main
- https://github.com/NirDiamant/RAG_Techniques
- https://python.langchain.com/docs/tutorials/rag/
- https://github.com/CornelliusYW/RAG-To-Know?utm_source=substack&utm_medium=email

Evaluation:

- https://arxiv.org/abs/2309.15217
- https://github.com/explodinggradients/ragas?tab=readme-ov-file
- https://www.nb-data.com/p/rag-evaluation-monitoring-and-logging

LLMOps:

- https://decodingml.substack.com/

Thank You!

Michail Dadopoulos

AI Research Engineer | Research Interests: LLMs, Agentic AI & Rei...

Feel free to connect!