Advanced RAG Techniques

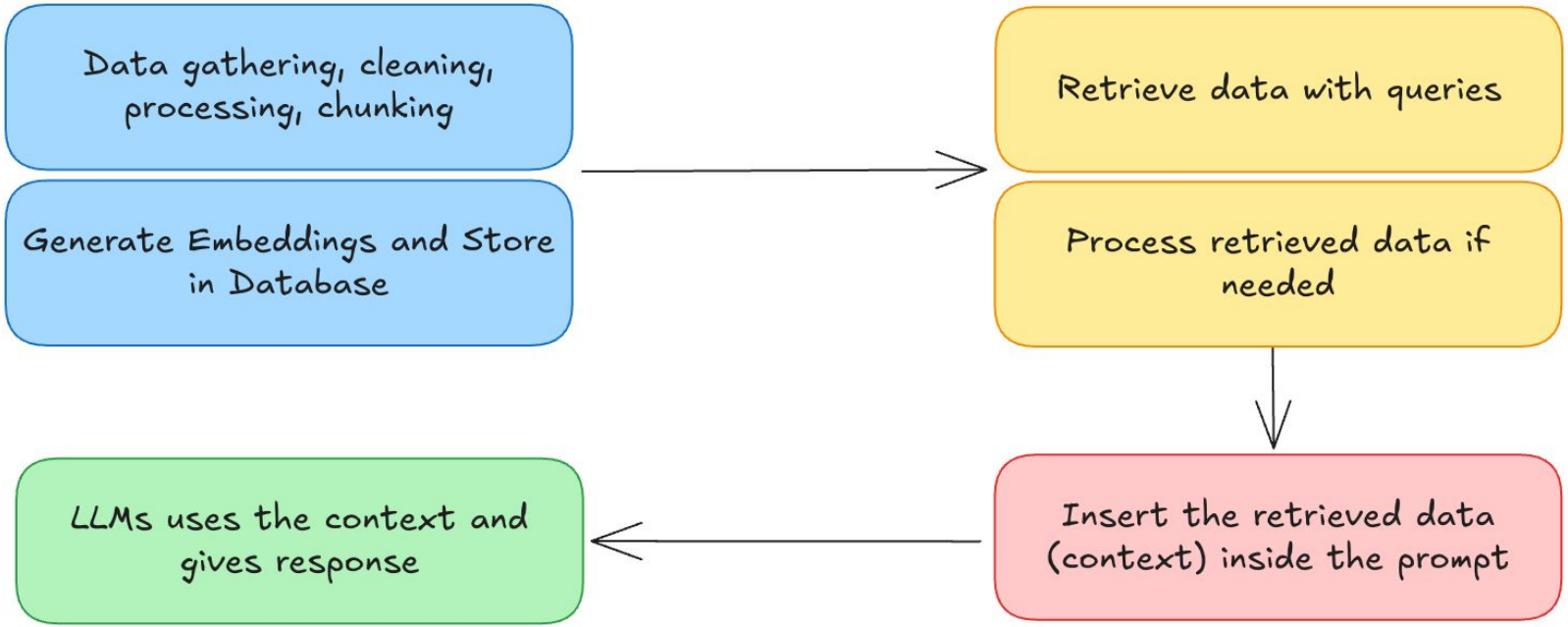
Md. Shahad Mahmud Chowdhury
AI Team Lead
Delineate (YC W25)

What is RAG?

The term comes from **R**etrieval **A**ugmented **G**eneration.

1. We retrieve some data based on query
2. We augment the prompt/query with the retrieve data
3. The LLMs generate response using the augmented data

Typical RAG life cycle

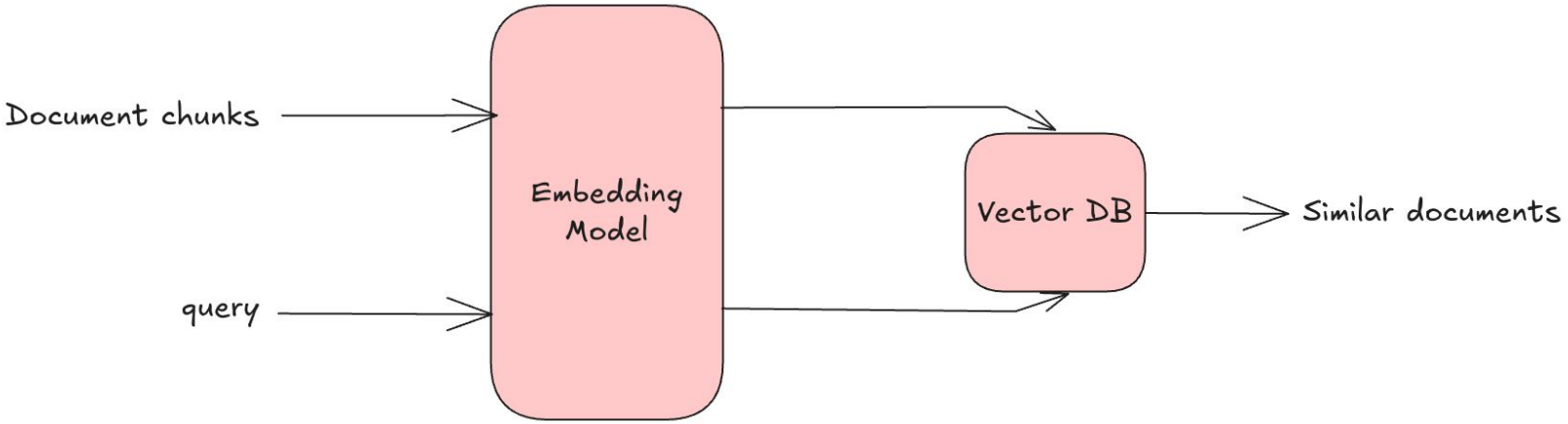


Basic RAG Limitations

Advanced RAG Techniques

- Query Transformation
- Hybrid Search
- Reranking
- Context distillation
- Applying Filters
- Entity-Aware Retrieval

Zoom Into Retrieval



- Document chunks and query are embedded via same model
- Semantic similarity is calculated based on embedding vectors
- Quality of retrieved documents depends on the query

Query Transformation

A technique to modify, rewrite, or expand a user's original query **before** it's sent to the retrieval system.

The core idea is that **the way a user asks a question is often not the best way to find the answer** in a vector database.

Query Transformation: Why we need it?

The retrieval step is the foundation of RAG. If we retrieve irrelevant documents (*garbage in*), the LLM will generate an irrelevant or wrong answer (*garbage out*).

Problem Type	User Query	Problem Description	Rephrased Query Example
Vagueness & Ambiguity	"Tell me about it."	The retriever has no idea what "it" is.	"Tell me about the 'Project Phoenix' deployment." (Using chat history)
Jargon Mismatch	"How do I fix the 'blue screen' error?"	The technical knowledge base might not use the term "blue screen."	"How to troubleshoot a 'BSOD' or 'Windows stop error'."
Complex Questions	"What's the difference between RAG and fine-tuning, and which is better for my chatbot?"	A single search query will struggle to find a document that perfectly answers all parts.	*(Text implies splitting into multiple, simpler queries)*

Query Transformation: Common Techniques

Contextualization

This is the most commonly used in chatbots. The original query is combined with previous messages to create a standalone, context-rich query.

- User (Turn 1): "What is 'Project Alpha'?"
- Bot (Turn 1): "Project Alpha is our new data analytics platform."
- User (Turn 2): "Why did it fail?"
- *Rephrased Query for Retrieval:* "Why did the 'Project Alpha' data analytics platform fail?"

Query Transformation: Common Techniques

Query Expansion

This involves broadening the query to cover more semantic ground.

- **Adding Synonyms/Related Terms:**
  - Original: "remote work policy"
  - Rephrased: "remote work policy OR work from home guidelines OR telecommuting rules"
- **Spelling & Typos:** The LLM can correct spelling mistakes before retrieval."

Query Transformation: Common Techniques

Sub-Query Generation or Decomposition

This is for complex questions. The LLM breaks the user's query into several smaller, independent queries. The RAG system then retrieves documents for all sub-queries and combines the context.

- Original: Compare the security of JWTs vs. session cookies.
- Rephrased Sub-Queries:
  - How does JWT security work?
  - How does session cookie security work?
  - Security vulnerabilities of JWTs
  - Security vulnerabilities of session cookies

Hybrid Search

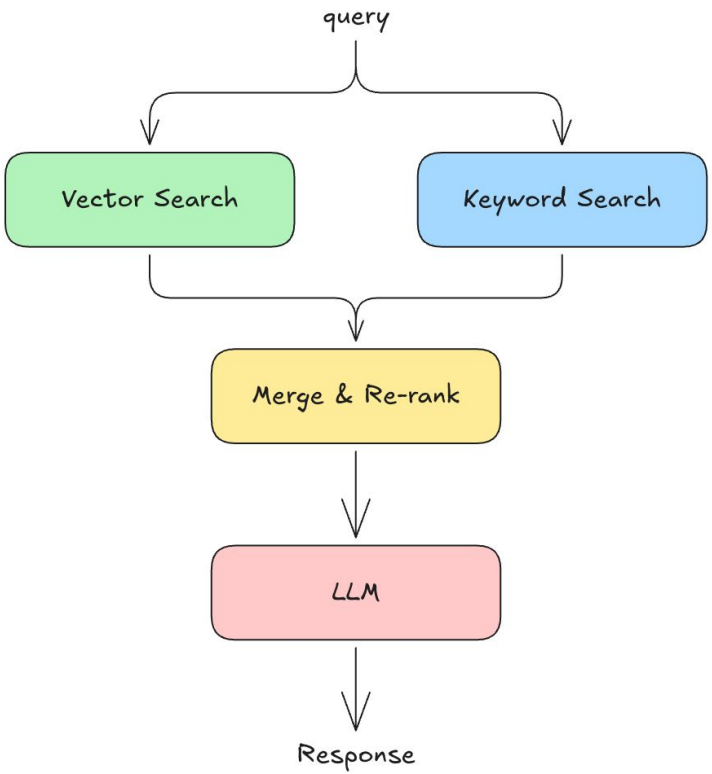
Hybrid search is a technique that combines two different types of search — keyword (*lexical*) search and vector (*semantic*) search — to retrieve the most relevant documents.

The core idea is that neither method is perfect on its own. By blending them, you get the "best of both worlds," leading to a much more accurate and robust retrieval step for our RAG pipeline.

Hybrid Search: Components

1. Vector or Semantic Search
    a. Searches the similar document by calculating distance of the document vectors against query vector.
    b. The vectors contains semantic meaning, hence its called semantic search.
    c. However, it struggles to find specific or exact match terms like names, error codes, etc.
2. Keyword or Lexical Search
    a. Searches for exact keywords in DB. Most commonly used algorithm is **BM25**.
    b. It is extremely precise and perfect for finding specific keywords.
    c. However, it has zero understanding of meaning.
3. Hybrid search brings the best of two parts. Semantic meaning from vector search and exact keyword from lexical search.

Hybrid Search: How does it work?



Re-ranking

Re-ranking is a technique used to significantly improve the quality of a Retrieval-Augmented Generation (RAG) system by adding a second, more sophisticated filtering step after the initial retrieval.

The standard RAG pipeline involves retrieving information and then feeding it to an LLM. But re-ranking helps to select or reorder the retrieved information before feeding to the LLM.

Re-ranking

With re-ranking, the retrieval becomes a two step process.

1. **Initial Retrieval:** First, we use a fast, traditional method (like semantic similarity) to grab a large bunch of potentially relevant documents. This step is broad and designed to capture anything that might be relevant.
2. **Re-ranking:** This large set of documents is passed to a "more sophisticated model" (a reranker) that takes a much closer look. This model analyzes the documents more deeply, considering:
    a. Contex
    b. User Intent
    c. Nuanced relationships between concept

The reranker then reorders the entire list, pushing the most relevant documents to the very top.

Re-ranking: How It’s Done

Two types of models are used for re-ranking:

1. **Cross-Encoders:** These are specialized models designed specifically for relevance ranking. They are generally faster and more efficient than LLMs for this task but may need fine-tuning for specific domains.
2. **LLM-based Re-ranking:** This uses a powerful LLM (like GPT-4 or Claude) to assess the relevance of each document. This is incredibly flexible and can handle very complex reasoning, but it can also be slower and more resource-intensive.

Context distillation

In RAG, Context Distillation is a technique for compressing the retrieved documents into a shorter, more relevant, and less "noisy" piece of context before sending them to the final LLM.

The core idea is to filter out the extra from your retrieved documents and pass only the most potent, relevant facts to the generator model.

This technique is also commonly called **Context Compression**.

Context distillation: Why is this necessary?

Context Compression is crucial for solving several major problems in RAG pipelines:

- **The "Lost in the Middle" Problem:** LLMs have a U-shaped performance curve. They pay high attention to information at the very beginning and very end of a prompt, but "lose" or ignore information buried in the middle. If we just put 10 full documents into the context, the most critical fact might be lost in the middle.
- **Context Window Limits:** We can't fit 100 documents into an 8K or 32K context window. You need a way to shrink them.
- **Noise and Distraction:** If 3 of our 10 retrieved documents are only vaguely related, they act as noise that can distract the LLM, reducing the quality of the answer or even causing hallucinations.
- **Cost and Latency:** Sending a massive prompt (e.g., 20,000 tokens) to a powerful LLM is slow and expensive. Sending a compressed 2,000 token prompt is much faster and cheaper.

Context distillation: How Does It Work?

1. **Retrieve:** First, the RAG system does a vector search to fetch potentially relevant documents (e.g., top 20).
2. **Distill/Compress:** This is the key step. We take these 20 documents and the original query and pass them to a Document Compressor. This compressor can be:
    a. **An Extractive Model:** A small LLM is explicitly prompted to "iterate over the documents and extract only the sentences or facts that are relevant to the query. This is the most common method.
    b. **An Abstractive Model:** A model that summarizes the documents in relation to the query, creating new, concise sentences.
    c. **Filtering:** A simpler method that just filters out entire documents that don't meet a certain relevance threshold
3. **Generate:** The final, LLM receives the original query plus this new, short, and highly relevant distilled context to generate the final answer.

Entity-Aware Retrieval

Entity-Aware Retrieval is a RAG technique that explicitly identifies specific, named "things" — or entities — in a user's query and then uses that information to retrieve more precise and relevant documents.

Instead of just searching for the general meaning of a query, it anchors the search to the specific nouns that matter most.

Entity-Aware Retrieval: Why Is This Needed?

Standard vector search is fuzzy. It's good at finding documents *with similar meaning*, but it can fail when precision is critical.

- **User Query:** "What is the status of Project Titan?"
- **Standard Vector Search:** Might retrieve documents about "titanium," "large-scale projects," or other projects with similar descriptions but misses the specific anchor.
- **Entity-Aware Search:** Knows "Project Titan" is a specific thing. It searches for documents that are semantically about "status" but are also explicitly linked to the "Project Titan" entity.

This method helps resolve ambiguity (e.g., "Tesla" the scientist vs. "Tesla" the car company) and dramatically increases the relevance of retrieved facts.

Entity-Aware Retrieval: How Does It Work?

This technique adds two crucial steps *before the main retrieval* phase:

1. **Entity Extraction:** First, the user's query is processed to find all the entities. This can be done either a NER model or an LLM.
2. **Entity Linking:** This is an optional but powerful step. The extracted entity string is linked to a unique ID in a Knowledge Graph or entity database. This resolves ambiguity and allows the system to pull in other known aliases.
3. **Filtered Retrieval:** The identified entities are used to filter the search results. This is most commonly implemented as metadata filtering.

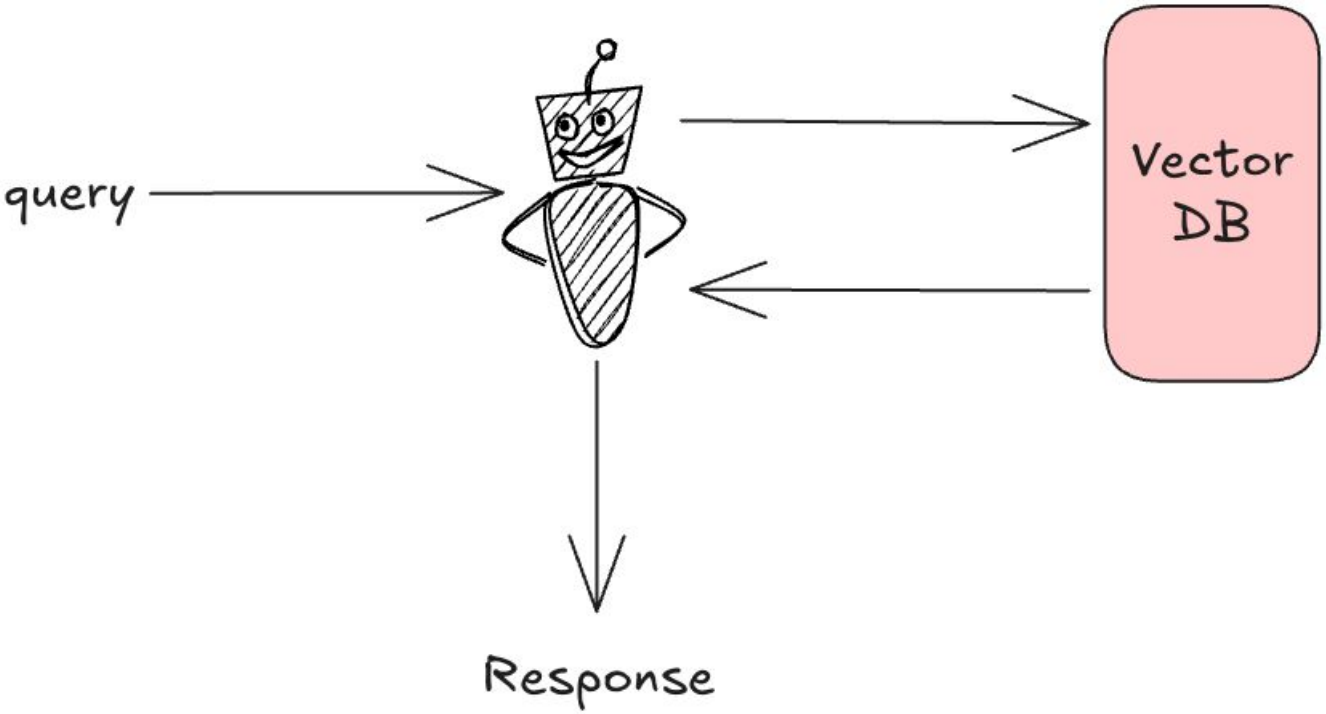
Applying Filters

This is the process of using information about our documents (their metadata) to narrow down the search *before* or *after* the retrieval step.

Generally we filter the documents with some kind of metadata like project name, date, etc.

It helps to reduce the search space and improves the retrieval performance.

Agentic RAG



Thank you!