### **1. Introduction to Retrieval-Augmented Generation (RAG)**

* **What is RAG?** RAG enhances LLMs by integrating external data retrieval, allowing models to generate responses grounded in **real-time, private, or domain-specific knowledge.**
* **Why Use RAG?** LLMs have a **knowledge cutoff** and can **hallucinate**. RAG mitigates this by retrieving **relevant, up-to-date information** before generating responses.

### **2. Core Components of RAG**

* **Indexing:** Prepares and structures external data into a searchable format for efficient retrieval.
* **Retrieval:** Find relevant documents from an indexed data store based on user queries.
* **Generation:** Uses retrieved documents as context to generate factually grounded responses.

### **3. Query Translation: Improving Retrieval Accuracy**

* **Why Translate Queries?** User queries are often ambiguous or incomplete, leading to poor retrieval results. Query translation **modifies the query** to enhance relevance.
* **Multi-Query Generation:** Rewrites a query from different perspectives to increase retrieval robustness.
* **Query Decomposition:** Breaks down complex questions into smaller sub-queries, solving them step-by-step.
* **Step-Back Prompting:** Reframes a specific question into a **broader, high-level question** to improve retrieval coverage.
* **RAG Fusion:** Retrieves from multiple sources and ranks (RRF - Reciprocal Rank Fusion) and provides the context for generation
* **HyDE (Hypothetical Document Embedding):** Creates a hypothetical document (say summary) using the question, performs embeddings, and uses that for retrieval of documents.
* **Reciprocal Rank Fusion:** A way to calculate combined rank in RAG fusion. Rank is in the denominator to give higher weight to better (lower rank) documents. Rank 1 and 2 has higher difference than rank 100 and 101. RRF(d) = Σ(r ∈ R) 1 / (k + r(d))

### **4. Routing: Directing Queries to the Right Data Source**

* **Logical Routing:** Uses an LLM to **classify the question** and direct it to the appropriate database (e.g., SQL, vector store, graph DB).
* **Semantic Routing:** Embeds both the query and available sources, then **computes similarity** to determine the best retrieval source.
* **Hybrid Routing:** Combines **logical and semantic routing** for improved efficiency.

### **5. Indexing: Structuring Data for Efficient Retrieval**

* **Why Index?** Raw documents are difficult to search directly. Indexing **organizes data** into a structure that allows fast retrieval.
* **Vector Indexing:** Converts documents into numerical embeddings, enabling **semantic similarity searches.**
* **Metadata Indexing:** Adds structured fields (e.g., **dates, categories, authors**) for efficient filtering.
* **Multi-Representation Indexing:** Stores **both raw documents and compressed summaries** to improve retrieval efficiency. Comparison made with compressed summaries for faster retrieval
* **RAPTOR (Recursive Abstractive Processing for Tree-Organized Retrieval):** Creates groups of documents, embeds and indexes them. Based on how generic or specific the queries are, the documents or clusters(summaries) are provided as context for generation.
* **ColBERT (Contextualized Late Interaction):** Creates contextual **token** embeddings for each document, compares it with query **token** embeddings using Maximum Similarity Sum, and retrieves documents using FAISS (Facebook AI Similarity Search).  
  **Disadvantage:** Requires a lot more computing and storage but optimizes using Quantization techniques.  
  **Advantage:** Improves retrieval accuracy by considering **local word meanings** rather than relying on a single document-level embedding.

### **6. Retrieval: Fetching Relevant Data from External Sources**

* **K-Nearest Neighbors (KNN) Search:** Finds the top-k most similar documents using **vector embeddings.**
* **Hybrid Retrieval:** Combines **semantic search (vector search)** with **structured metadata filtering** for more precise results.
* **Re-Ranking:** Uses secondary scoring models to reorder retrieved documents based on **relevance and quality.**

### **7. Adaptive RAG: Dynamic Improvement of Retrieval and Generation**

* **Why Adaptive RAG?** Standard RAG relies on static retrieval. Adaptive RAG **evaluates responses in real time** and adjusts retrieval or generation accordingly.
* **Hallucination Detection:** Checks if generated answers align with retrieved data. If discrepancies are found, retrieval is **re-triggered.**
* **Rewriting & Re-Retrieval:** If retrieval quality is low, the query is **reworded** and retrieval is repeated.

### **8. Generation: Creating Responses Grounded in Retrieved Data**

* **Why Ground Responses?** Without retrieved data, LLMs rely on pre-trained knowledge, leading to hallucinations.
* **RAG with Reasoning & Tool Use:** LLMs can **call APIs, perform calculations, or query databases** before finalizing an answer.

### **9. CRAG (Corrective RAG)**

* An enhanced form of RAG that iteratively refines retrieval and generation by detecting hallucinations, retrieval errors, or incomplete responses and correcting them in real-time