جامعة الحسين التقنية
Al Hussein Technical University

Center of Professional Development and Community Outreach

Generative AI Training Program - Professional Development (Reskilling) Training Program

Generative AI - RAG

Module 6-B:

RAG

Dr. Abeer Al-Hyari
Abeer.alhyariups@htu.edu.jo

Module-6-B

Module	Module Name and Focus	Key Topics to Cover	PLOs Addressed
Module 6-B	**Retrieval Augmented Generation (RAG)**	1. The Power of Retrieval Augmented Generation (RAG) for enterprise use cases.	Model Development, Prompt Engineering

Objectives

1. Understand RAG fundamentals
2. Learn vector databases & retrieval technologies
3. Evaluate and test retrieval quality
4. Optimize RAG systems for enterprise use cases
5. Explore RAG applications & alternatives

Topics Covered

What is Retrieval-Augmented Generation (RAG)?

RAG (Retrieval-Augmented Generation) is an AI framework that combines the strengths of traditional information retrieval systems (such as search and databases) with the capabilities of generative large language models (LLMs).

The RAG Problem & Solution

The Problem: LLMs Only Know Their Training Data

LLMs are trained on massive datasets, but they have **two fundamental limitations**:

a) Fixed Knowledge (Knowledge Cutoff)

- They cannot know anything that happened after their last training update.
- They cannot access real-time information (unless explicitly connected to external tools).

The RAG Problem & Solution

The Problem: LLMs Only Know Their Training Data

LLMs are trained on massive datasets, but they have **two fundamental limitations**:

b) No Access to Private or Proprietary Data

LLMs **cannot** see your:

- internal documents
- PDFs, emails, reports
- database records
- company policies

- Unless this data is **explicitly provided**, the model cannot use it.

Result:
LLMs may hallucinate, give outdated answers, or miss important enterprise context.

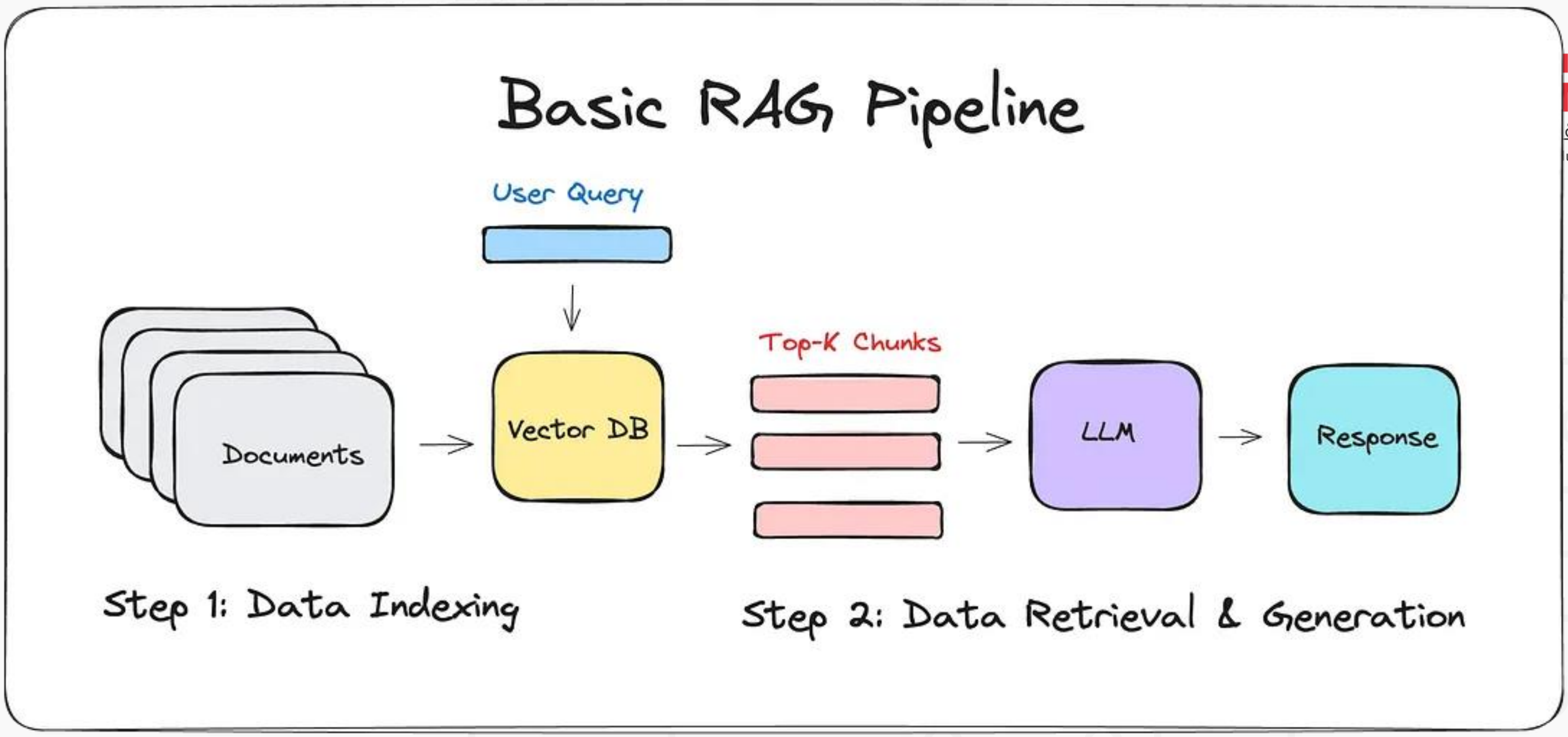
The RAG Problem & Solution

The Solution: Retrieval Augmented Generation (RAG)

What RAG Does

- RAG enhances an LLM by **feeding it relevant, external, up-to-date information** retrieved from your own data sources.

Instead of trying to make the model “know more,” we give it the knowledge it needs at query time.



https://miro.medium.com/v2/resize:fit:1100/format:webp/1*J7vyY3EjY46AlduMvr9FbQ.png

How RAG Works

1. Index your data

- Split documents into chunks
- Generate embeddings
- Store in a vector database (e.g., Pinecone, Chroma, Weaviate)

2. Retrieve relevant chunks

- User asks a question
- System finds the most similar document sections
- This provides context the model never had during training

How RAG Works

3. Augment the prompt

Retrieved text is added to the LLM prompt as grounding context

4. Generate answer

The LLM produces an output that is:

- grounded in real data
- Factual
- up-to-date
- aligned with your internal knowledge sources

RAG for Enterprise Use Cases

Up-to-date information

- LLMs can respond with information from “right now,” not last year’s training data.

Uses proprietary data securely

- Your private data stays inside your systems (vector DB, storage).

Reduces hallucinations

- The model answers based on **real documents**, not guesses.

Cheaper than fine-tuning

- You do not need to retrain the model every time data changes.

جامعة الحسين التقنية
Al Hussein Technical University

Vector Databases in RAG

Vector Databases Are the Base of RAG Retrieval

A Vector DB lets us quickly find the most similar documents to a query by comparing embeddings (vectors).”

The RAG system can rapidly retrieve relevant knowledge entries within the vector database by transforming user queries into vector embeddings.

This approach enables LLMs to access the most up-to-date information stored in the database when responding to user queries, effectively addressing issues such as delays in knowledge updates and the occasional inaccuracies in generated content, often called "hallucinations.“

In RAG, you typically store embeddings in a vector database (vector store).

Information Retrieval Technologies

Category	Search Engines	Relational Databases	Document Databases	Vector Databases
Primary Products	Elasticsearch	MySQL	MongoDB	Milvus
Implementation Principle	Utilizes inverted indexing for fast text searches with limited vector search capabilities	Employs standardized data models and SQL for optimal transaction processing, struggles with handling unstructured data	Stores data in JSON format, offering flexible data models and basic full-text search but limited semantic search capabilities	Designed specifically for high-dimensional vectors, uses Approximate Nearest Neighbor (ANN) algorithms for effective semantic similarity searches

Information Retrieval Technologies

Category	Search Engines	Relational Databases	Document Databases	Vector Databases
Primary Products	Elasticsearch	MySQL	MongoDB	Milvus
Use Cases	Ideal for full-text searches and straightforward data analysis	Best for applications demanding high consistency and intricate transaction management	Well-suited for rapid development and environments with frequent data model changes	Optimal for unstructured data-based retrieval like image and semantic text searches
Large Vector Dataset Retrieval Efficiency (The higher the better)	Medium	Low	Low	High

Information Retrieval Technologies

Al Hussein Technical University

Category	Search Engines	Relational Databases	Document Databases	Vector Databases
Primary Products	Elasticsearch	MySQL	MongoDB	Milvus
Multimodal Generalization Capability (The higher the better)	Medium	Low	Low	High
Suitability for RAG Retrieval(The higher the better)	Medium	Low	Low	High
Total Costs(The lower the better)	High	High	Medium	Low

Vector DB Classification

Vector DB Classification

1. By location

- **Local / embedded** (FAISS, Chroma in a notebook).
- **Cloud-hosted / managed** (Pinecone, Weaviate Cloud, managed Postgres with pgvector).

Vector DB Classification

2. By search algorithm

- **Exact search** (brute-force, small datasets).
- **Approximate Nearest Neighbor (ANN)** for large datasets:
  - HNSW (Hierarchical Navigable Small World graphs),
  - IVF, PQ (in FAISS), etc.

Vector DB Classification

3. By data model

- **Vector-only stores** – focus on embeddings + simple metadata.
- **Hybrid stores** – combine:
  - Full-text search (BM25),
  - Vector search,
  - Structured filters (SQL-style).

Faiss

- Faiss is a library for efficient similarity search and clustering of dense vectors.
- It contains algorithms that search in sets of vectors of any size, up to ones that possibly do not fit in RAM. It also contains supporting code for evaluation and parameter tuning.
- Faiss is written in C++ with complete wrappers for Python/numpy. Some of the most useful algorithms are implemented on the GPU.
- It is developed primarily at Meta's Fundamental AI Research group.
- https://github.com/facebookresearch/faiss

Chroma

- Open-source search and retrieval database for AI applications.
- https://www.trychroma.com/
- Chroma is an open-source embedding database.
- The fastest way to build Python or JavaScript LLM apps with memory
- Chroma makes it easy to build LLM apps by making knowledge, facts, and skills pluggable for LLMs.
- https://github.com/chroma-core/chroma

Pinecone

- The vector database for machine learning applications.
- Build vector-based personalization, ranking, and search systems that are accurate, fast, and scalable.
- Pinecone use cases : Natural language processing, Computer vision, Recommendation systems
- https://github.com/pinecone-io

جامعة الحسين التقنية
Al Hussein Technical University

What Problems does RAG solve?

What Problems does RAG solve?

1. **Hallucinations**: Traditional generative models can produce incorrect information. RAG reduces this risk by retrieving verified, external data to ground responses in factual knowledge.
2. **Outdated Information**: Static models rely on training data that may become outdated. It dynamically retrieves latest information ensuring relevance and accuracy in real time.
3. **Contextual Relevance**: Generative models often struggle with maintaining context in complex or multi turn conversations. RAG retrieves relevant documents to enrich the context improving coherence and relevance.

What Problems does RAG solve?

4. Domain Specific Knowledge: Generic models may lack expertise in specialized fields. It integrates domain specific external knowledge for tailored and precise responses.

5. Cost and Efficiency: Fine tuning large models for specific tasks is expensive. It eliminates the need for retraining by dynamically retrieving relevant data reducing costs and computational load.

6. Scalability Across Domains: It is adaptable to diverse industries from healthcare to finance without extensive retraining making it highly scalable.

RAG Applications

RAG Applications

- **Question-Answering Systems**: It enables chatbots or virtual assistants to pull information from a knowledge base or documents and generate accurate, context aware answers.

https://github.com/VivekChauhan05/RAG_Document_Question_Answering

- **Content Creation and Summarization:** It can gather information from multiple sources and generate concise, simplified summaries or articles.

https://github.com/aryanmangal769/Legal-Doc-Summarization-LLM-LoRA-RAG

RAG Applications

- **Conversational Agents and Chatbots:** It enhances chatbots by grounding their responses in reliable data making interactions more informative and personalized.

https://github.com/NirDiamant/GenAI_Agents

- **Information Retrieval:** Goes beyond traditional search by retrieving documents and generating meaningful summaries of their content.

https://github.com/reichenbch/RAG-examples

- **Educational Tools and Resources:** Provides students with explanations, diagrams or multimedia references tailored to their queries.

https://github.com/lukablaskovic/edu_bot

جامعة الحسين التقنية
Al Hussein Technical University

RAG Alternatives

RAG Alternatives

Method	Description	When to Use
Prompt Engineering	Adjusts the input prompt to guide model behavior without changing its training.	When you need a quick and simple solution for specific tasks or queries.
Retrieval-Augmented Generation (RAG)	Combines retrieval and generation to use external data for more factual and context-aware responses.	When you want the model's responses to include real-time, relevant information from external sources.

RAG Alternatives

Method	Description	When to Use
Fine-Tuning	Retrains the model on a smaller, domain-specific dataset.	When you need better performance on a particular topic or industry data.
Retrieval-Augmented Generation (RAG)	Combines retrieval and generation to use external data for more factual and context-aware responses.	When you want the model's responses to include real-time, relevant information from external sources.

RAG Alternatives

Method	Description	When to Use
Pre-Tuning	Trains the model from scratch using a large and diverse dataset.	When you want to build a strong foundation for later customization and adaptation.
Retrieval-Augmented Generation (RAG)	Combines retrieval and generation to use external data for more factual and context-aware responses.	When you want the model's responses to include real-time, relevant information from external sources.

جامعة الحسين التقنية
Al Hussein Technical University

Optimizing & Testing Retrieval in RAG Systems

Summary of Google Cloud Best Practices

Why Retrieval Quality Matters

- RAG performance depends heavily on the relevance and accuracy of retrieved documents.
- Even a strong LLM will hallucinate if retrieval is weak.
- Testing retrieval = testing the foundation of the entire RAG pipeline.

Core Metrics for Retrieval Testing

1. Precision@k

- Measures how many of the top-k retrieved documents are relevant.
- Good for tasks where irrelevant documents harm quality.

2. Recall@k

- Measures whether the system can retrieve all relevant documents.
- Important when missing information leads to incorrect answers.

3. MRR (Mean Reciprocal Rank)

- Evaluates how early the first relevant document appears in results.
- Good for systems where the top 1–2 results matter most.

Advanced Metrics

- **nDCG (Normalized Discounted Cumulative Gain)**
  - Scores results by ranking quality: higher-ranked relevant docs = better.
- **Hit Rate / Top-k Accuracy**
  - Checks whether any relevant document is retrieved.
  - Useful for classification-style retrieval tasks.

Testing Methods

A. Golden Dataset (Labeled Evaluation Set)

- Prepare a set of query–relevant document pairs.
- Use this fixed dataset to compute retrieval metrics.
- Ensures consistent, repeatable testing.

B. Human Judgement

- Human reviewers label:
  - Relevant / partially relevant / irrelevant
- Helps catch subtle relevance issues model metrics miss.

Stress & Robustness Testing

Test retrieval under challenging conditions:

- Long queries
- Ambiguous queries
- Noisy or informal queries
- Domain-specific jargon
- Multilingual or code-mixed input

Ensures the retriever performs well beyond “clean” examples.

Content Quality Testing

Evaluate the *documents* being retrieved:

- Are chunks too small or too large?
- Does chunking split important information?
- Is metadata complete (titles, timestamps, IDs)?
- Are documents up-to-date?

- Improving document quality directly boosts retrieval metrics.

Evaluating Hybrid Retrieval

Combine keyword search + vector search in tests:

- Compare:
  - Vector-only retrieval
  - Keyword/BM25 retrieval
  - Hybrid scores
- In many enterprise tasks, hybrid retrieval gives the highest recall & precision.

Runtime & Efficiency Testing

Even if retrieval quality is high, test:

- Latency of embedding + search
- Scalability with large corpora
- Cost-performance trade-offs
- Index update behavior (batch vs real-time)

High-quality retrieval must also be **fast and cheap enough** to deploy.