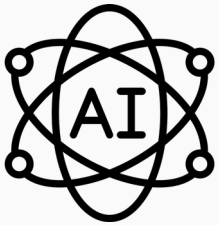



4 Days Workshop on

# Artificial Intellegence

# 01. Itroduction to RAG

Basic introduction RAG and it’s components.

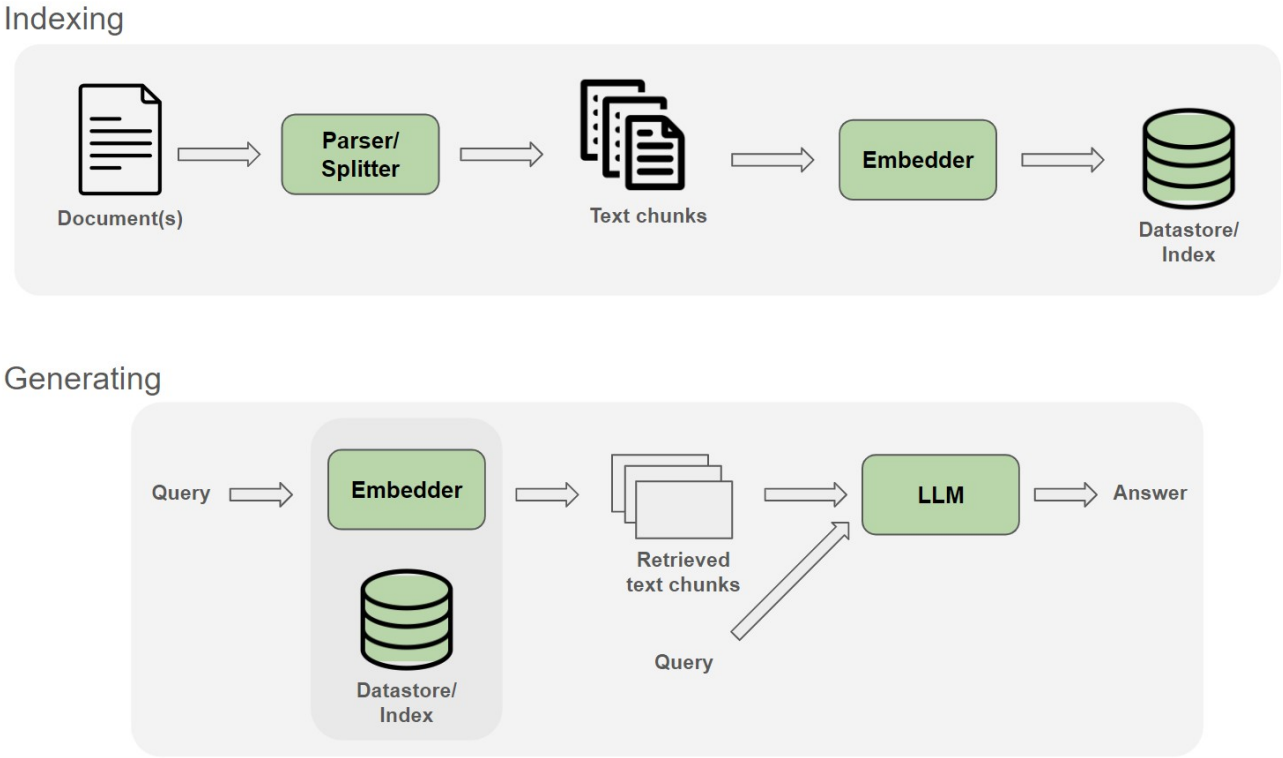
# Problems with LLM

- LLMs can’t access your private, recent, or external data.
- They often make up facts or hallucinate confidently.
- Updating their knowledge requires costly retraining or fine-tuning.
- They struggle with long documents or large context windows.
- Answers are not grounded — you don’t know where the info came from.

# RAG

- RAG is a method that combines LLMs with external data sources.
- It retrieves relevant info from documents before generating an answer.
- Uses embeddings to semantically match your question with stored chunks.
- The LLM then uses both the prompt and retrieved data to respond.
- Great for building “Chat with your data” apps using PDFs, wikis, etc

# Components of RAG



# Parsing

- Extracts text, metadata, and structure from PDFs
- Use libraries like PyMuPDF, pdfplumber, PyPDF2, pdfminer.six
- For scanned/image PDFs, use OCR like pytesseract or easyocr
- Use docling or unstructured to preserve layout, headers, and tables
- Paid option: LlamaParse (LlamaIndex) offers high-quality, structure-aware parsing
- Some LLMs can parse PDFs directly, but offer less control.

# Chunking

- LLMs have token limits → split large text into smaller parts
- Common methods: fixed-size, sliding window, header-based
- Overlapping chunks help preserve context
- Too large = context cutoff; too small = lacks meaning
- Good chunking boosts retrieval accuracy

# Embedding

- Converts text chunks into numeric vectors that capture meaning
- Similar meanings = closer vectors in space
- Use sentence-transformers or OpenAI embeddings
- These vectors are used to compare queries with documents

# Vector Database

- Stores and indexes embeddings for fast semantic search
- Popular tools: FAISS, Chroma, Pinecone, Weaviate
- Supports storing metadata (page, filename, etc.)
- Enables top-k similarity search
- Forms the memory base for your chatbot

# Retrieval

- At runtime, user query is embedded into a vector
- Vector DB finds top-k similar chunks
- Filtering based on score threshold improves relevance
- Retrieved chunks are passed as context to the LLM
- Ensures answers are grounded in real data

# LLM – Final Generation

- LLM gets: [retrieved context] + [user query] + [additional prompt]
- Generates an answer based only on relevant info
- No training or fine-tuning needed
- Works with almost any LLM
- Produces accurate, contextual, and customized responses