Practical RAG: Building Specialized Chatbots Step-by-Step

Hocine Abdellatif Houari, @hahouari

LLMs?

LLMs (or Large Language Models) are artificial intelligence models that are trained on large amounts of text data to generate human-like text.

LLMs?

They are general purpose AIs, unlike specialized AI models that can perform specific tasks (e.g. Face Recognition, OCR, etc.).

LLMs and Their Limitations

While LLMs are powerful, they have limitations:

- **Static Knowledge**: They are only trained on data available up to a certain point.
- **Large Size**: They require significant resources for fine-tuning or training.
- **Context Limitation**: They struggle to retrieve specific information efficiently (or hallucinate).

Bridging the Gap: Why RAG?

To address these limitations, **Retrieval-Augmented Generation (RAG)** combines 2 steps:

1. **Retrieval Systems**: For fetching up-to-date, task-specific, or large-scale information on demand.
2. **LLMs**: For generating coherent and contextually relevant text.

This synergy enhances the effectiveness of AI systems in dynamic and specialized use cases.

Why not Fine-Tuning?

Fine-tuning is a process of taking a pre-trained model to train it and tweak its parameters to perform better on a specific task.

Pros & Cons over Fine-Tuning

RAG			Fine-Tuning
Up-to-Date	✅		Can still be outdated
No Training	✅		Training required
Easy to Switch Model	✅		Hard to Switch
Retrieval Quality	👌	👌	Training Quality
Extra Retrieval Step		✅	Real-time

What do we need to implement RAG System?

- Large Language Model (e.g. OpenAI GPT-3 or Anthropic Sonnet)
- Structured Data (e.g. in JSON format)
- Embedding Model for Semantic Search
- Database with vector storage & search capabilities

What do we need to implement RAG System?

- Large Language Model ✅
- Structured Data ✅
- Embedding Model for Semantic Search 🤔❔
- Database with vector storage & search capabilities ⏳

Embedding Models? 🤔

They are specialized ML models that convert data (like text, images, or audio) into vectors (embeddings). These vectors allow us to perform semantic search.

Database with vector storage & search capabilities

Examples:

- PostgreSQL (using pgVector)
- SurrealDB
- Pinecone
- Milvus