**GEN-AI**

**ASSIGNMENT-3**

**PART-1**

**TASK-1**

**Q1. What is the motivation behind Retrieval-Augmented Generation (RAG)?**

Ans- The **motivation behind Retrieval-Augmented Generation (RAG)** is to improve the **factual accuracy**, **relevance**, and **scalability** of language models by **combining generation with information retrieval**.

**Q2. Explain the difference between RAG and standard LLM-based QA.**

### Ans- Difference Between RAG and Standard LLM-Based QA

| **Feature** | **Standard LLM-Based QA** | **RAG (Retrieval-Augmented Generation)** |
| --- | --- | --- |
| **Knowledge Source** | Fixed internal knowledge from pretraining | External documents retrieved at query time |
| **Factual Updates** | Requires retraining the model | Just update the retrieval corpus |
| **Response Accuracy** | May hallucinate or provide outdated info | More factual and grounded in real data |
| **Interpretability** | Hard to trace source of answer | Can point to retrieved sources |
| **Handling Niche Queries** | May fail due to missing knowledge | Retrieves relevant context on the fly |
| **Architecture** | Single-stage generation (prompt → output) | Two-stage: retrieval → generation |
| **Scalability of Knowledge** | Limited to training data | Easily scalable by adding to the corpus |

**Q3. What is the role of a vector store in a RAG pipeline?**

Ans-A **vector store** in a RAG pipeline **stores and retrieves documents as vector embeddings**, allowing the system to **find the most relevant information** to a user's query efficiently.

**Q4. Compare “stuff”, “map\_reduce”, and “refine” document chain types in LangChain.**

### Ans-Comparison Table:

| **Feature** | **Stuff** | **Map-Reduce** | **Refine** |
| --- | --- | --- | --- |
| **Strategy** | Concatenate all docs into one prompt | Process docs individually (map), then combine (reduce) | Process one doc, refine output with each next doc |
| **Efficiency** | Fastest, but limited by context length | Slower, but scalable | Slower, but detailed and iterative |
| **Best For** | Small number of short docs | Large sets of documents | Step-by-step refinement needs |
| **LLM Input** | All docs at once | One doc per map step, then all outputs for reduce | One doc initially, then refines with each |
| **Context Limit Issues** | Yes – limited by LLM’s input size | No – processes in chunks | Less sensitive – sequential processing |
| **Accuracy/Depth** | May be shallow or miss info | Balanced and scalable | More in-depth but may compound errors |

**Q5.What are the main components of a basic LangChain RAG pipeline?**

Ans-A basic **LangChain Retrieval-Augmented Generation (RAG)** pipeline has **two major stages**:

1. **Retrieval** – find relevant documents
2. **Generation** – use those documents to generate an answer

**TASK-2**

**Draw or describe the flow of a RAG system showing:**

**● User Query**

**● Retriever**

**● Vector Store**

**● LLM**

**● Final Answer Generation**

**┌────────────┐**

**│ User Query │**

**└─────┬──────┘**

**│**

**▼**

**┌─────────────────┐**

**│ Embed the Query │ ← (Convert query to vector)**

**└─────┬───────────┘**

**│**

**▼**

**┌────────────────────┐**

**│ Vector Store │ ← (Stores embeddings of documents)**

**│ (e.g., FAISS, etc.)│**

**└─────┬───────────────┘**

**│**

**▼**

**┌────────────────────┐**

**│ Retriever │ ← (Finds top-k similar documents)**

**└─────┬──────────────┘**

**│**

**▼**

**┌────────────────────────────┐**

**│ Retrieved Documents (Context)│**

**└────────────┬──────────────┘**

**│**

**▼**

**┌───────────────┐**

**│ Prompt + LLM │ ← (e.g., GPT-4, BERT, Claude)**

**└──────┬────────┘**

**│**

**▼**

**┌────────────────────┐**

**│ Final Answer Output │**

**└────────────────────┘**