

**The Superior University, Lahore**

|  |  |  |  |
| --- | --- | --- | --- |
| Course Title: | PAI - LAB | | |
| Programme Name: | BSDS | | |
| Semester: | 4rd | Section: | BSDSM-4A |
| Student’s Name: | H.Hamza Dar | | |
| Instructor Name: | Prof. Rasikh Ali | | |

***Task 11***

**Describe the Difference between:**

**1. Lang-Chain**

**2. RAG**

**3. LLMs**

**4. FAISS**

**5. Vector**

**6. VectorDB**

**7. Generative AI**

**8. GANs**

**1. LangChain**

**LangChain** is a Python (and JavaScript) framework designed to build applications that use **Large Language Models (LLMs)** like ChatGPT in a more structured way.

**Key Features:**

* Connects LLMs to external **data sources** (like databases, PDFs, web pages).
* Allows **memory**, so the chatbot can "remember" past interactions.
* Supports **retrieval-augmented generation (RAG)**, where it fetches relevant data before answering.
* Helps chain **multiple prompts**, tools, or APIs together.

**Use Case:**

You want to build a **chatbot that answers questions from your company’s documents**—LangChain helps manage all the moving parts.

**2. RAG (Retrieval-Augmented Generation)**

**RAG** is a technique to improve the **accuracy of LLMs** by combining two steps:

1. **Retrieval**: Find relevant documents or text chunks from a database (usually a **Vector DB**).
2. **Generation**: Give those documents to an LLM to create a more accurate and grounded response.

**Why It's Useful:**

LLMs like ChatGPT don’t know anything beyond their training data. RAG lets them **access real-time or domain-specific info**.

**Use Case:**

A university chatbot that fetches **real-time admission info** from your website and answers queries.

**3. LLMs (Large Language Models)**

These are **AI models trained on massive text datasets** to understand and generate human-like language.

**Examples:**

* GPT-4 (OpenAI)
* PaLM (Google)
* LLaMA (Meta)

**Capabilities:**

* Chatting, summarizing, writing essays, coding, ec.

**Limitation:**

They don’t have real-time memory or awareness of current data **unless connected to external tools** (via LangChain, RAG, etc).

**4. FAISS (Facebook AI Similarity Search)**

FAISS is a **library from Facebook** for performing **fast similarity search on high-dimensional data**, especially **vectors**.

**What It Does:**

Given a vector (e.g., for a user query), FAISS finds **the most similar vectors** in your database (documents, images, etc).

**Use Case:**

Used in RAG to quickly find the **most relevant chunks of a document** that match a user's question.

**5. Vector**

In AI, a **vector** is a list of numbers (like [0.23, 0.97, ...]) that represents the **meaning** of text, image, or other data.

**How We Get It:**

LLMs or **embedding models** (like OpenAI’s text-embedding-ada) convert words/documents into vectors.

**Why It Matters:**

Vectors make it possible to **compare meanings** of texts using math. For example, two similar sentences will have similar vectors.

**6. VectorDB (Vector Database)**

A VectorDB stores and indexes **vectors** and allows you to perform **similarity search**.

**Examples:**

* **FAISS** (library)
* **Pinecone**, **Weaviate**, **Chroma**, **Qdrant** (full-featured hosted solutions)

**What It Does:**

* Store vector representations of data (e.g., paragraphs of a textbook)
* Search for vectors that are **close in meaning** to a user’s query

**Use Case:**

Used in RAG pipelines to fetch the best-matching content based on a query vector.

**7. Generative AI**

**Generative AI** refers to AI that can **create new content** such as text, images, music, or code.

**Examples:**

* GPT-4 (text)
* DALL·E (images)
* GitHub Copilot (code)
* ChatGPT (conversations)

**Why It’s Powerful:**

Instead of just analyzing or classifying, it can **generate human-like and creative outputs** from scratch.

**8. GANs (Generative Adversarial Networks)**

A GAN is a **type of Generative AI** used mostly for **image and video generation**.

**How It Works:**

* **Generator**: Tries to create fake but realistic data (e.g., a fake face)
* **Discriminator**: Tries to spot which data is real vs fake
* They **train together** until the generator becomes really good

**Use Case:**

* Deepfakes
* AI-generated art or people who don’t exist
* Style transfer (e.g., turning a photo into a painting)