MODELS:

What are Models in LangChain? Models are components in LangChain that let you interact with different types of AI models, mainly Language Models and Embedding Models. LangChain provides a common interface so that you can work with various AI models easily without worrying about their individual differences.

Two Main Types of Models:

Language Models (LLMs): These take text input and output text. They are used for tasks like generating text, summarization, translation, and question-answering.

Embedding Models: These take text input and output numeric vectors (embeddings) that represent the meaning of the text. Embeddings are used for semantic search and similarity calculations.

LangChain supports connecting to different providers' models, including closed-source paid models like OpenAI's GPT and Anthropic's models, as well as open-source models from Hugging Face.

Language Models can be further divided into:

LLMs: General-purpose models for single-turn text processing.

Chat Models: Specialized for conversation tasks, supporting multi-turn dialogue and memory of past interactions.

The modern preference for Chat Models over LLMs because Chat Models provide features like conversation history, role-awareness (assigning roles like "doctor" to the model), and better contextual understanding.

Open-Source Language Models (LLMs)

Open-source models are AI models trained by researchers and shared freely on the internet. You can download them to your own computer, have full control over how you use them, and even change or update them as you like. These models:

- **Give you full control:** You can train them further, modify how they work, and deploy them wherever you want (your server or cloud).
- **Cost nothing:** They are free to use, unlike paid models.
- **Offer strong privacy:** Since you’re in control, your data stays private you don’t have to share information with outside companies.
- **Can be deployed anywhere:** You can set them up on your own computers or on cloud systems.

Some **popular open-source language models** include:

1. Llama
2. Mistral
3. Falcon
4. Bloom

Hugging Face is a popular online platform that hosts many of these open-source models.

Ways you can use open-source models:

- **Running locally:** Download and run them directly on your device.
- **Hugging Face Inference API:** Use the models online through APIs if you don't want to set up hardware.

Disadvantages:

- Need powerful hardware (like GPUs) to run well.
- Setup can be tricky or complicated.
- They may not be as well-refined since they often lack "Reinforcement Learning from Human Feedback" (RLHF). You can improve this by training them with better data.
- Many aren't as good at handling images or audio together with text (limited multimodal features).

Vector Embeddings

When working with models like OpenAI, you often use **vector embeddings** these are just lists of numbers that capture the meaning of text.

- For OpenAI, **small models** create vectors of length **1536**; **large models** create vectors of length **3072**.
- **Longer vectors** can give more detail and handle more complex context, but they can be more expensive to store and process.
- **Shorter vectors** are cheaper but carry less information.
- With OpenAI, the code used to make embeddings is **not open-source**, while many other models have open-source code.

How Document Similarity Search Works (Simple Steps)

- **Turn your texts into numbers:** Use an embedding model (like those from OpenAI or Hugging Face) to convert all your texts into vectors, which are just lists of numbers that represent each document's meaning.

- **Store these vectors:** Save all these vectors in a special database called a vector database (like FAISS, Chroma, or Pinecone).
- **Handle user questions:** When someone asks a question, you also turn that question into a vector using the same model as before.
- **Find the best matches:** The system then compares this question vector to your stored vectors using simple math (cosine similarity or dot product) to find the documents that are most similar to what the user asked.

  This way, you can search through lots of text and find what we need based on meaning, not just keyword matching