Lang Chain :-

ACHIEVER
Date : / /

LangChain is an opensource orchestration framework used for application development using large language Models (LLM's)

* LangChain's tools and APIs simplify the process of building LLM-driven applications like chatbots and AI-agents.

* The LangChain modular environment allows for program that use multiple LLM's:
For ex:- an application that uses one LLM to interpret user queries and another LLM author a response.

Families of LLM's

① OpenAI → GPT-3.5, 4, 4 Turbo, 4o (omni)

② Anthropic → Claude 1, 2, 3, 3.5

③ Google DeepMind → Gemini 1, 1.5, 2.0...

O.S ④ Meta → LLama 2 (os), Llama3 etc

O.S ⑤ Qwen → Qwen 1.5, 2.0, 2.5, Qwen 3

O.S ⑥ Mistral → Mistral 7B, Mistral 8x7B etc.

ACHIEVER
Date: / /

Integration with LLM's :-

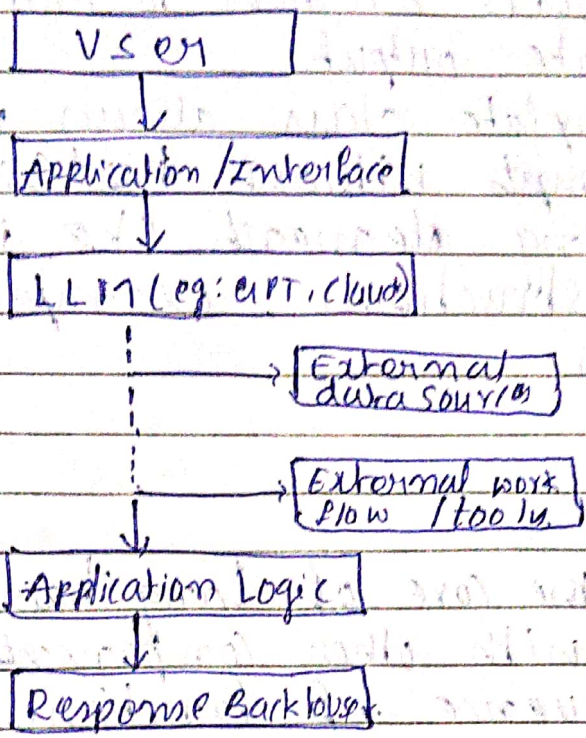
* LLM's are the brain. Applications are body. Data & API's are the sense and tools. orchestration frameworks are the nervous system that connects it all.

① LLM's are not applications :-
* they're statistical models trained on massive datasets.
* they must be paired with an application to serve real-world usercases.

② Need for external data sources :-
* LLM's are not omniscient.
* For real-time awareness or domain specific tasks, LLM's must connect to :
  ① internal docs, databases or API's
  ② News / current sources
  ③ Company specific knowledge.

③ Need for external workflow :-
* LLM's often need to call API's or tools.
* Ex :- Slack Integration → LLM must connect with slack API

Diagram style work flow :-



Explaination :-

① User → Sends a query (eg: summerize this PDF)
② App/Interface → Collects the query (chatGPT UI) etc
③ LLM → interprets input, generate potential respons
④ External Data → if info is outside model's memory (like company docs), it retrives via RAG (Retrieval - Aragumented Generation).
⑤ External tools / work flow → if tasks require action (Searching web, calling slack API) LLM calls tools.
⑥ Orchestration Layer → connects everything, manages work flow, decides when to call what [LangChain, IBM watsonx].
⑦ Response → Sent Back to user in natural language format.

Prompt Template :-

* prompt templates are structured instructions that guides LLM's on how to interpret input & generate output.
* The PromptTemplate class allows you to formalize prompts without hard coding them, including elements like input variables, instructions, examples and output formats.

Chains :-

* chains are the core of LangChain workflow linking LLM's with other components to execute a sequence of functions.

Ex:-

```
chain_ex = LLMChain(llm=flan-ts, prompt=
    ExamplePrompt)
chain_ex.run("input")
```

* Simple sequential chain uses the output of one function as input for the next
* Each step can use different prompts, tools, parameters or models.

Indexes :-

To achive certain tasks, LLm will need access to specific external data sources not included in its training dataset, such as internal documents, emails or datasets. LangChain collectively refers to such external documentation as "indexes".

① Document loaders :-

* LangChain offers a wide variety of document loaders for third party applications.
* This allows for easy importation of data from sources like file storage services (like Dropbox, google Drive and Microsoft OneDrive), web content (like youtube, pubMed or specific URLs), databases (like Pandas, MongoDB and Microsoft), among many others.

② Vector Database :-

* Represent data as vector embeddings (numerical vectors).
* Enable low-latency queries even for large datasets.
* LangChain supports 25+ embedding methods and 50+ vector stores.

③ Text Splitters :-

* Divide large texts into smaller, semantically meaningful chunks to speed up processing and reduce computational load.

④ Retrieval & RAEi (Rebrieval - Arguamented Generation) :-
* Models retrieve relvent info from connected data sources using rehever modules.
* Agentic RAEi syskems can also perform tasks beyond retrieval (eg: calculations, emails, data analysis).

LangChain use case :-

① AI Applications :- LangChain enables a wide range of AI applications, from simple Q & A and text generation to complex reasoning tasks.

② chatbots :- provides context-specific responses integrate into existing workflows and communication channels using APIs.

③ Summarization :- summerize complex articles, transcripts or incoming emails.

④ Question Answering :- Retrieve information from documents or specialized knowledge bases (eg: Wolfram, arXiv, PubMed).

⑤ Data Augmentation :- generate Synthetic data to expand training datasets for ML.

Wrapper classes for different Providers :-

① Google Gemini :
- -> ChatGoogleGenerativeAI -> chat based models
- -> GoogleGenerativeAIEmbeddings -> for Embedding

② Open AI :
- -> ChatOpenAI -> chat based GPT Models
- -> OpenAIEmbeddings -> embedding with OA models

③ Hugging face :
- -> HuggingFaceHub -> run models hosted on HF Hub
- -> HuggingFacePipeline -> run local Hugging Face models with transformers.

④ Ollama (local models) :
- -> ollama -> run models like Llama, Mistral, Gemma