AI Agents

Week 1 → foundation

- Setting up Environment →
- IDE → Cursor → Cursor is an AI platform that's powered by LLMs that allows us to be so productive and its built on Vscode.
- uv → used in Virtual Environment. It is really fast and simple. It is a package manager
- steps →
  1. Installing Git
  2. clone the Repo
  3. Intall cursor
  4. Setting up uv.
- using dotenv module and its load_dotenv class for loading the api-key stored in .env file.
- load-dotenv is an utility function that looks for a file with a particular name in project root directory and it happens to be always .env or name.env.
- .env consist of the environment variables.
- and load_dotenv function load the variables for using in the project.
- load-dotenv utility looks at the file, interprets it, and sets the environment variables as a result.
- a parameter override = True tells that if you already have set the environment variable in the past then override that variable value.
- import os → Os module provides us a function getenv() that read the api-key from the .env file.
- import openai → since in this project we are using groq-api-key instead of openai_api-key some functions & classes and modules are different.

→ Planner agent, M [illegible] coming up with number of scenarios.

◉ OpenAI →
→ OpenAI is an lightweight library for connecting to OpenAI's endpoint on the cloud.
→ OpenAI format to represent messages →
messages = [{"role": "user", "content": "what is 2+2?"}]
→ It is represented as list of dictionaries where each dictionary has a role and a content

◉ Calling OpenAI →

```
response = openai.chat.completions.create(
    model = "",  // model to be used,
    messages = messages // the message in openAI format
)
print(response.client[0].message.content)
```

◉ Day 2 → Agents & Agentic Architecture
→ What is an Agent?
Ans → AI Agents are programs where LLM outputs control the workflow. → huggingface
that simply means that one output from LLM is able to decide what tasks are carried out in a sequence.
→ People tends to often use the term Agentic AI, if any or all of the haulmarks are met →
① multiple LLM calls
② LLMs with ability to use Tools
③ An environment where LLM's interact with each other, a kind of coordination orchestration environment.
④ A planner to coordinate activities.
⑤ Autonomy → this terms really defines the essence of agentic AI.

→ Planner agent, ~
coming up with number of scenarios.

Agentic Systems

Anthropic distinguishes two types →

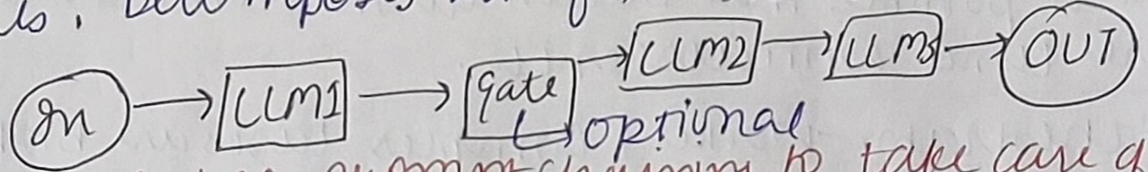
① **Workflow** → are systems where LLMs and tools are orchestrated through predefined code paths.

② **Agents** → are the systems where LLMs dynamically direct their own process and tool usage, maintaining control over how they accomplish tasks.

→ ① **Workflows** → anthropic identifies 5 different design patterns that you'll find when building agentic system that has workflow in them.

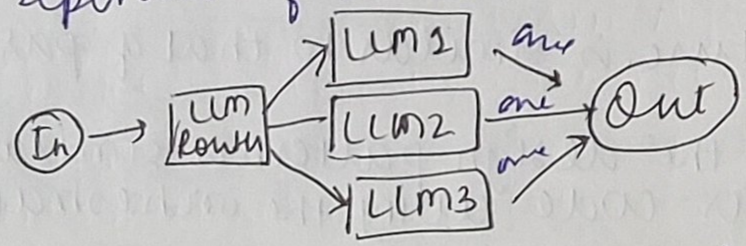
5 types →

① **Prompt chaining** → creating a chains of LLM calls. Decomposes into fixed sub-tasks.



and we use prompt chaining to take care of each LLM call very precisely, get the most effective LLM response.

② **Routing** → Directs an input into specialised sub-task, ensuring seperation of concern.

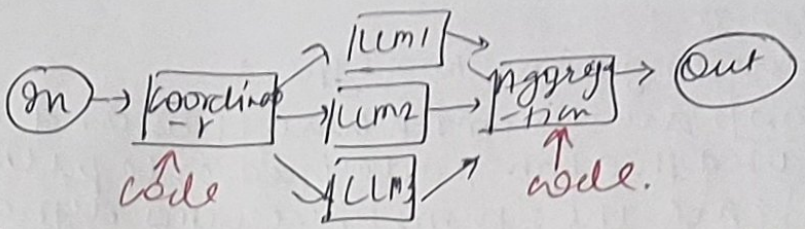


→ LLM1, LLM2, LLM3 are specilised to do some different tasks.

→ LLM router job is to clarify the tasks, which of the specialists will be best equipped to clarify the task

→ In the model, there is some artificial agents used.

③ **Parallelization** → Breaking down tasks and running multiple subtasks concurrently.

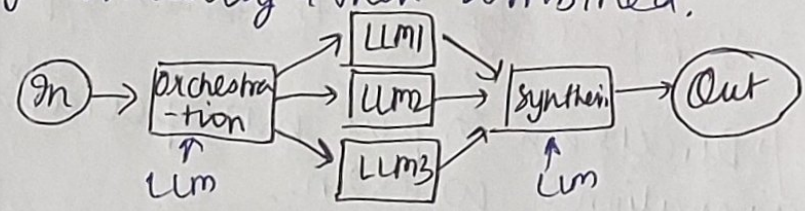


→ coordinator is a node that takes a tasks and break it down into multiple ~~dices~~ pieces that should all run in parallel.

→ so, in routing workflow one of the LLM model was executing the ~~task~~ task at one time but in parallelization we are granting to three ~~times~~ llms to carry all three different activities concurrently.

→ It is possible that we can do the same task for three times concurrently.

(4) **Orchestrator-worker** → Complex tasks are broken down dynamically & then combined.



→ this structure is similar to that of parallelization structure.

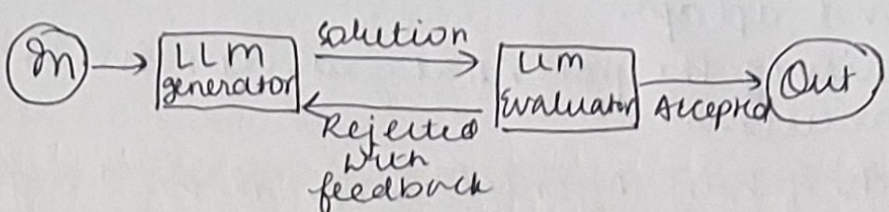
→ Although the design pattern is similar but it is not longer code doing the orchestration it is an an LLM

→ Now, instead of code, we are using an LLM model to break down the task into smaller steps and then again an model to combine the results.

→ We can debate on the topic that orchestrator-worker design is an agents rather than an workflow since, the model can choose the way to divide the task and it could choose how many llm's get assigned any activity.

→ here LLM break down the task → LLM carry out the task → LLM combines → Out

→ ⑤ Evaluator-optimizer → LLM output is evaluated by another

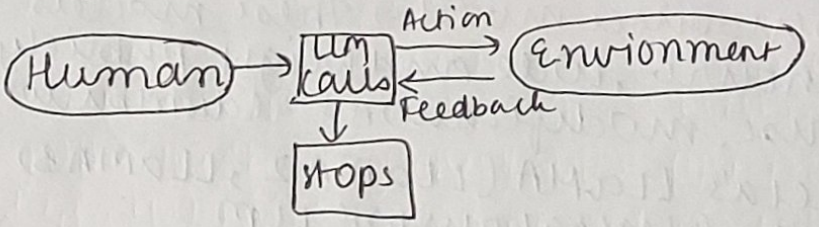


→ in this design two LLM's are used one for doing some specific task and the other one for evaluating the output of the first LLM's.
→ LLM evaluate only evaluate the work of the first LLM, and based on that it choose either it is accepted or it is rejected.
→ If ~~the~~ it accepts it, then it goes to the output
→ If it is rejected then rejection with the reasons goes back to the LLM generator.
→ It uses a powerful feedback loop.
→ It is an effective pattern

② Agents → By contrast with the workflow pattern agents are →

① ~~open-minded~~ open-ended
② it has feedback-loop
③ No fixed-path → It does not uses serious of steps

Generic Diagram →



→ human intiates a request or input to LLM
→ LLM process the humans input & process it to generate an "Action" that is executed or applied with the function.

⑤ Stable Diffusion (by Stability AI)
use → Text-to-image generation

Coc ⑥ Whisper → [illegible]

→ Environment provides feedback back to the LLM based on the action executed, allows the LLM to learn and adapt.

→ After achieving the potential goal, LLM can stop the proper execution

◉ Drawbacks of using Agents framework and agentAI design patterns →

① Unpredictable path
② Unpredictable Output.
③ Unpredictable cost → because we do not know how it will take
④ Monitoring → visibility & understanding of whats going on behind the scene.
⑤ Guardrails → it ensures your agent behave safely, consistently, and within your intended boundries.
- It makes sures that model is doing what it is supposed to be doing.

Day 3 → Orchestrating LLMs

◉ Calling multiple LLMs →
→ We ~~will be using~~ can use either Paid APIs and open source models in the cloud and locally throughout this course.

Notes → Opensource model → These models make their architecture, code and weight publicly available for free use, modification & redistribution.

Eg → ① Meta's LLaMA (LLAMA 2, LLAMA 3)
use → General purpose LLM
② Mistral & Mixtral
use → General purpose LLM, in chatbots & coding assistants.
③ Falcon
use → Language generation and reasoning
④ Bloom →
use → Multilingual Language generation.

... def do_some_processing() → hv

⑤ Stable Diffusion (by Stability AI)
use → Text-to-image generation
⑥ Whisper → (by OpenAI)
use → speech-to-text transcription

② Closed source Models →
These models are proprietary - source code, training data, and weights are not shared publicly. They are typically accessed via paid APIs

Eg ① GPT-4 / GPT-4o (By OpenAI)
- use → chatbots, text generation, reasoning, vision, coding
② Claude (by Anthropic)
use → Ethical AI assistant & Enterprise Applications
③ Gemini (formerly Bard, Google Deepmind)
use → Web assistant, coding & research
④ Cohere Command R+
use → Retrieval-augmented generation (RAG), NLP
⑤ Amazon Titan (by AWS)
use → Enterprise NLP tasks.

→ We can use 'gemini-2.0-flash' with Gemini API key
→ Deepseek is a upstart startup that has powerful model.
→ Groq → open-source LLMs including Llama3.3
→ Ollama → open source LLMs on the system locally
very similar to openAI

Lab 2 →
① In this Lab2, they are experimenting with open-source & closed source of LLM for seeing the difference in the way of answering the question.
② And the second objective is to do orchestration between models.

→ openAI : gpt-4-o-mini

↳ we use ~~client.messages.create~~

we use openai.chat.completions.create(

model = " ",

messages = messages;

)

answer = responses.choices[0].message.content

→ anthropic claude-3-7-sonnet-latest

claude = Anthropic() we use claude.messages.create(

model = " "

message = " "

max_tokens = " ")

answer = response.content[0].text

→ gemini ("gemini-2.0-flash".

gemini = OpenAI (api_key = —, base_url = " ")

→ same as method you did in openai.

OpenAI is used with google gemini, aside. OpenAI function you will mention the base_url for building an endpoint.

and by giving the base_url we are switching to google's endpoint instead of openAI.

→ Deepseek is executed as that of gemini google. like base_url.

→ groq is again similar to gemini & deepseek.

→ "!" → shortcut to execute terminal commands directly from python cell.

→ (""" """) triple ~~sto~~ quotes in python is used for writing multi-line strings and docstrings

and integrate

coming up with number of features.