AGENTIC - AI

26-6-2024

Curriculum

- Foundations of Agentic AI
- Lang Graph fundamentals
- Advanced Lang Graph
- AI Agents
- Agentic RAG
- Productanization

Gen AI v/s Traditional AI

- Traditional AI is about finding patterns in data and giving predictions
- Gen. AI is about learning the distribution of data so that it can generate a new sample from it.
- The best part of Gen AI is that it can mimic humans.

Application Areas

- Creative and Business Writing
- Software development
- Customer support
- Education
- Designing

What is Agentic AI?

Agentic AI is a type of AI that can take up a task or goal from a user and then work towards completing it on its own, with minimal human guidance.

It plans, takes action, adapts to changes, and seek help only when necessary.

Key Characteristics

- Autonomous
- Goal Oriented
- Planning
- Reasoning
- Adaptability
- Content Awareness

NOTE: If any chatbot contains these 6 characteristics then it will known as AI Agent

Autonomy

Refers to the AI System's ability to make decisions and takes on its own to achieve a given goal, without needing step-by-step human instructions.

→ It's proactive

→ Autonomy in multiple facets
- Execution
- Decision Making
- Tool Usage

→ Autonomy can be controlled
- Permission Scope
- Human in the loop
- Override
- Guardrails/Policies

→ Autonomy can be dangerous

Goal Oriented

Being goal-oriented means that the AI system operates with a persistent objective in mind and continuously directs its actions to achieve that objective, rather than just responding to isolated prompts.

→ Goals acts as a compass for Autonomy

→ Goals can come with constraints

eg) "Hire an engineer from india"
↑ constraint

→ Goals are stored in core memory -
--- like json format ---

→ Goals can be altered

Planning

Planning is the agent's ability to break down a high-level goal into a structured sequence of actions or subgoals and decide the best path to achieve the desired outcome.

Step 1: Generating multiple candidate plans

- Plan A: Use internet service i.e. LinkedIn, Indeed, etc
- Plan B: use Internal refferal

Step 2: Evaluate Each Plan

- Efficiency : (which faster?)
- Tool Availability : (which tool available)
- Cost : (Does it require Premium tools?)
- Risk : (Will it fail if we get no applicants?)
- Alignment : (remote only?, budget? -constraints)

Step 3: Select the best Plan with the help of:

1) Human in the loop 2) A pre-programmed policy

Reasoning

It is the cognitive process through which an agentic AI system interprets information, draws conclusions, and makes decisions - both while planning ahead and while executing actions in the real time.

Reasoning During Planning :

- Goal Decomposition
- Tool selection
- Resource estimation

Reasoning During Execution :

- Decision-Making
- HITL Handling
- Error Handling

Adaptability

It's a agent's ability to modify its plans, strategies, or actions in response to unexpected conditions - all while staying aligned with the goal.

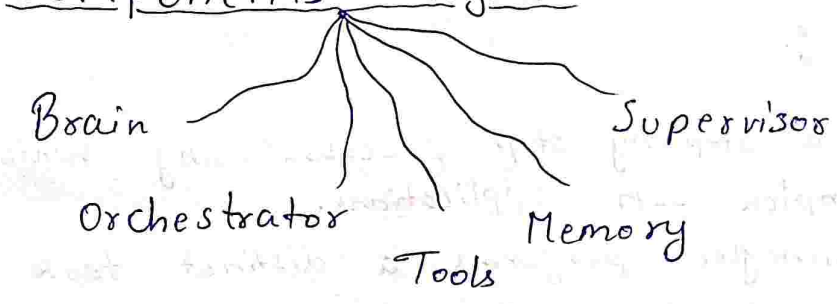
- Failures (Any API*)
- External Feedback (less no. of Application) - in our case - example
- Changing Goals (Hire a freelancer)

Content Awareness

It is the agent's ability to understand, retain & utilize relevant information from the ongoing task, past interactions, user preferences, and environmental cues to make better decisions throughout a multi-step process.

- → Content awareness is implemented through memory
- → Short term memory
- → long " "

Components of Agent



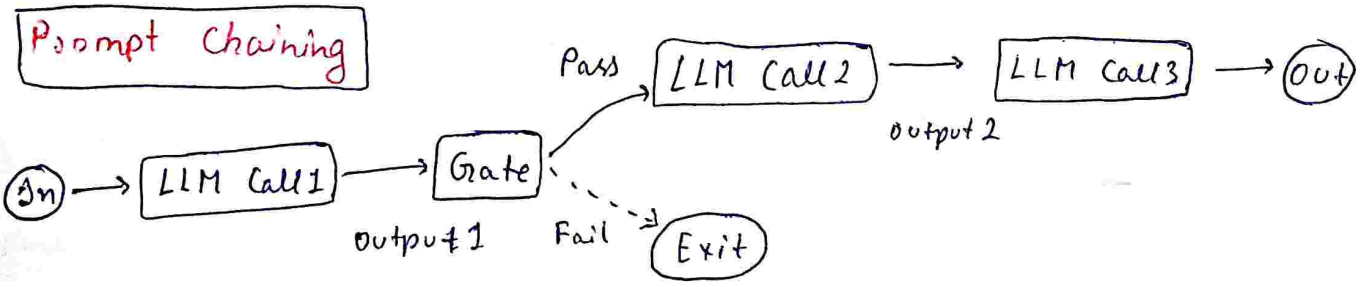
Challenges while developing Workflow Using LangCh

→ Control flow complexity
→ Handling State → L.Chain में कोई method नहीं है।
→ Event driven Execution
→ Fault Tolerence → किसी system में कुछ गड़बड़ होने के बाद भी वो work कर रहा है या नहीं
→ Human In the Loop
→ Nested Workflow ← not a challenge
It's a feature
→ observability → हर एक step को track करता है L.Graph

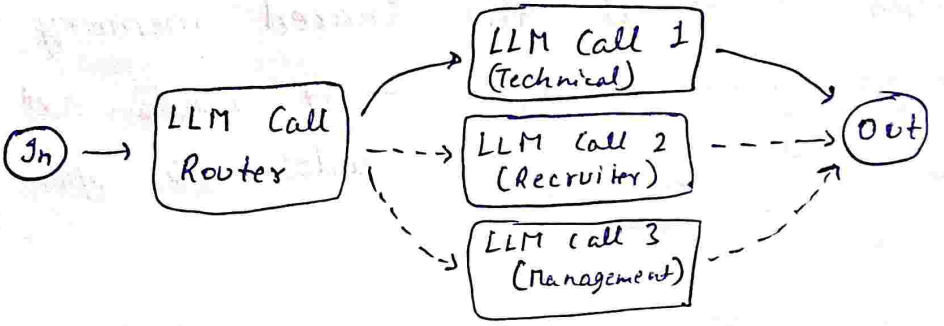
#४ ---------- LangGraph Core Concepts ----------

LLM Workflows :-

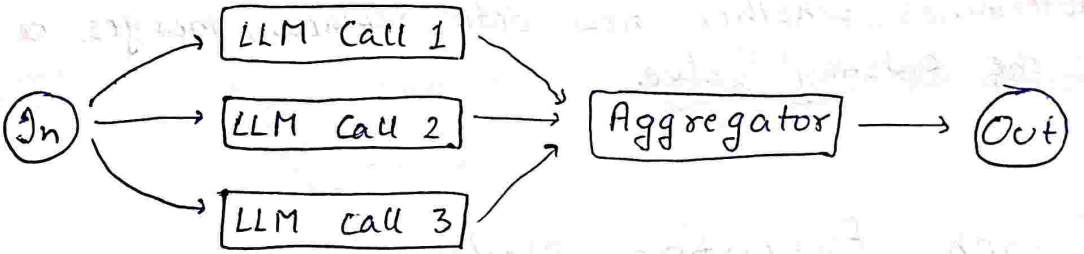
→ LLM Workflows are a step by step process using which we can build complex LLM applications.
→ Each step in a workflow performs a distinct task such as : prompting, reasoning, tool calling, memory access or decision-making.
→ Workflows can be linear, parallel, branched, or looped, allowing for complex behaviours. like retries, multi-agent communication, or tool augmented reasoning.
→ Common Workflows :



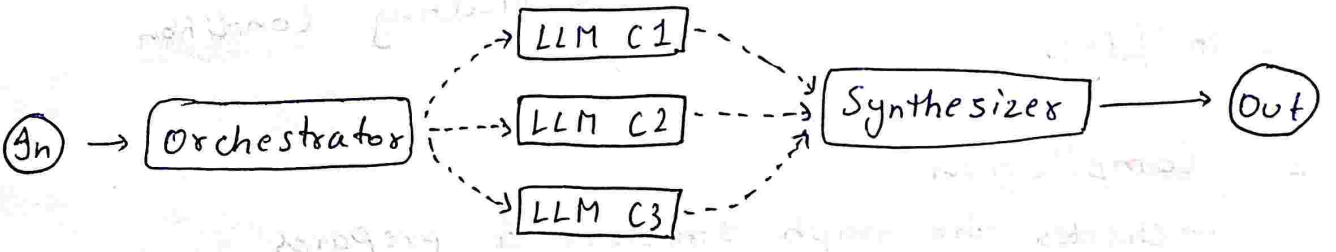
Routing



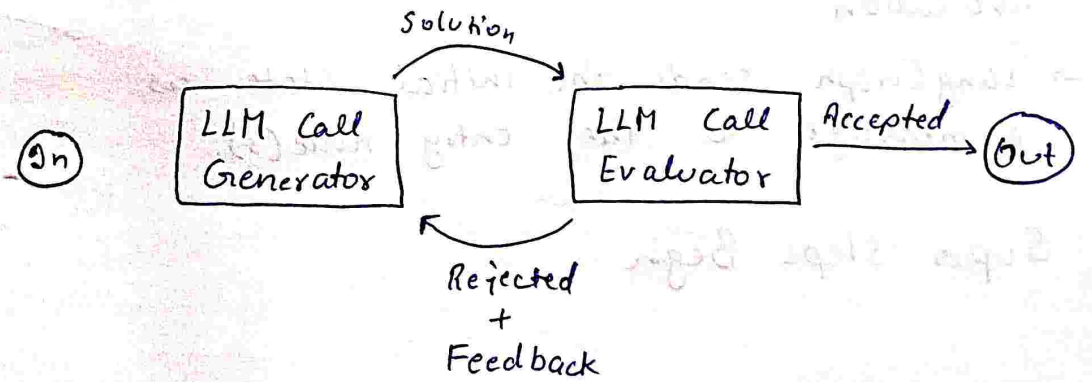
Parallelization



Orchestrator Worker



Evaluator Optimizer



State

In langGraph, state is the shared memory that flows through your workflow — it holds all the data being passed between nodes as your graph runs

- accessible
- mutable

Reducers

Reducers in langGraph define how updates from nodes are applied to the shared state.

Each key in the state can have its own reducer, which determines whether new data replaces, merges, or add to the existing value.

LangGraph Execution Model

1. Graph Definition
   - State schema
   - Nodes
   - Edges
2. Compilation
   → checks the graph structure & prepares it for execution.
3. Invocation
   → LangGraph sends the initial state as a message to the entry node(s).
4. Super steps Begin
5. Message Passing & Node Activation
6. Halting Condition

Sequential Workflow

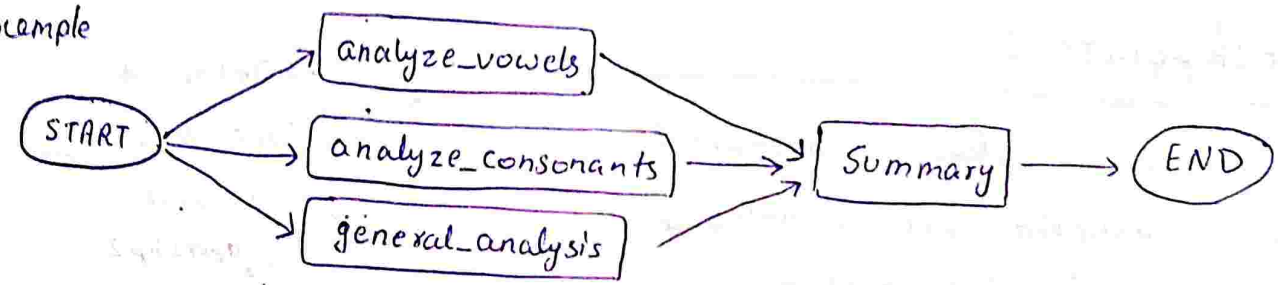
NOTE: In every workflows 'state' is passed as input to node & return also from node as output

- May be partial state or fully state updated, both are suitable.

Parallel workflow

- only partial workflow state updations are done on parallel nodes/edges

Example



vowel_count
consonants_count
words_count

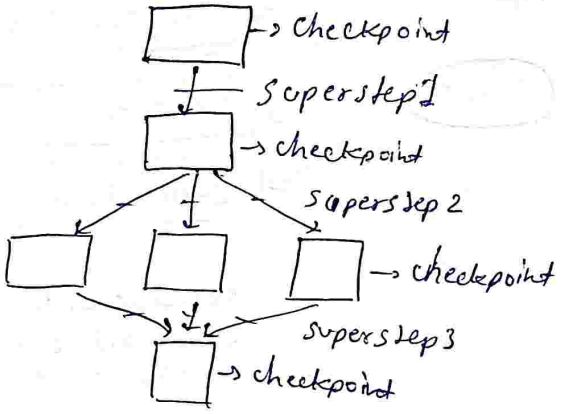
Persistence

- Persistence in LangGraph refers to the ability to save & restore the state of a workflow over time.
- At each execution of ~~the~~ graph state is erased when exec finished
- It store not only the final_values but also intermediate values.
- It provide fault tolerence : ex: if there are total 8 nodes & our workflow crash at node 3 (anyhow) then we will not need to restart it from starting, it will start from node 3

Checkpointers

- Each superstep → checkpointers
- each checkpoint save in database just after superstep execution
- for example :

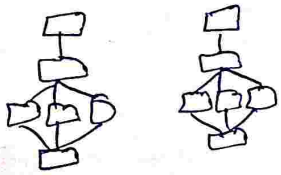
  state → numbers : [list[int], add]



START CP1 → numbers : [1] # Initial_state
CP2 → [2] → numbers : [1,2]
CP3 → [3] [4] [5] → numbers : [1,2,3,4,5]
END CP4 → numbers : [1,2,3,4,5]

Part of Persistence. that allow to pause, save progress & resume later.

Threads



thread_id = 1 = 2

- same workflow with different initial state
- These states stored in database against thread id
- It is used like in chatbots history

Benefits of Persistence

- Short term memory
- Fault tolerence
- HITL
- Time Travel