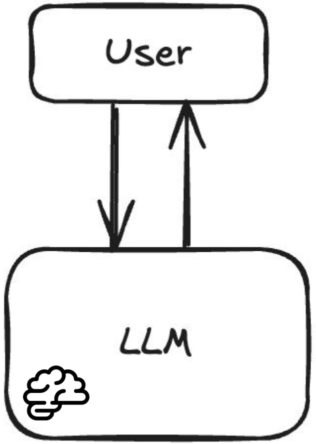
Exploring LLM-based Agents

Jerry Zikun Chen

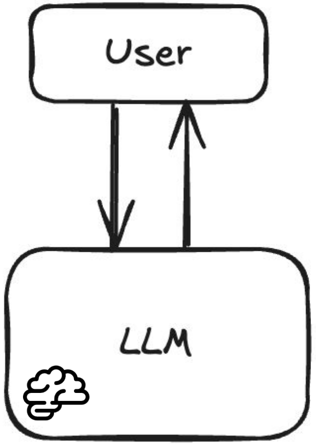
ChainML.net / Theoriq.ai

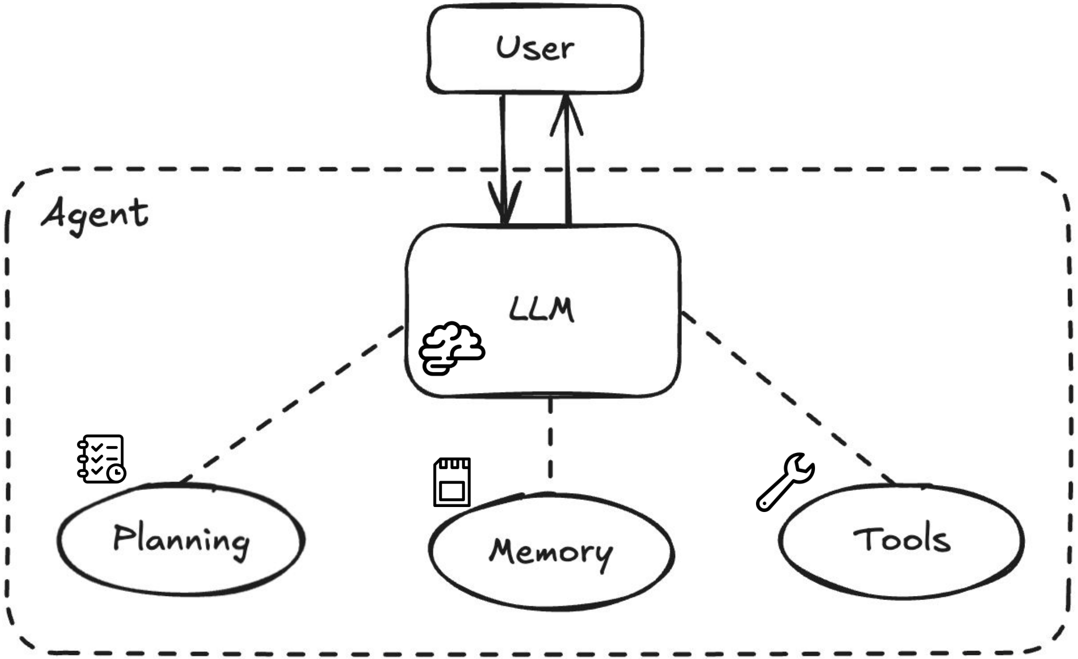
Dec. 9th, 2024

Traditional LLM Interaction



What is an Agent?



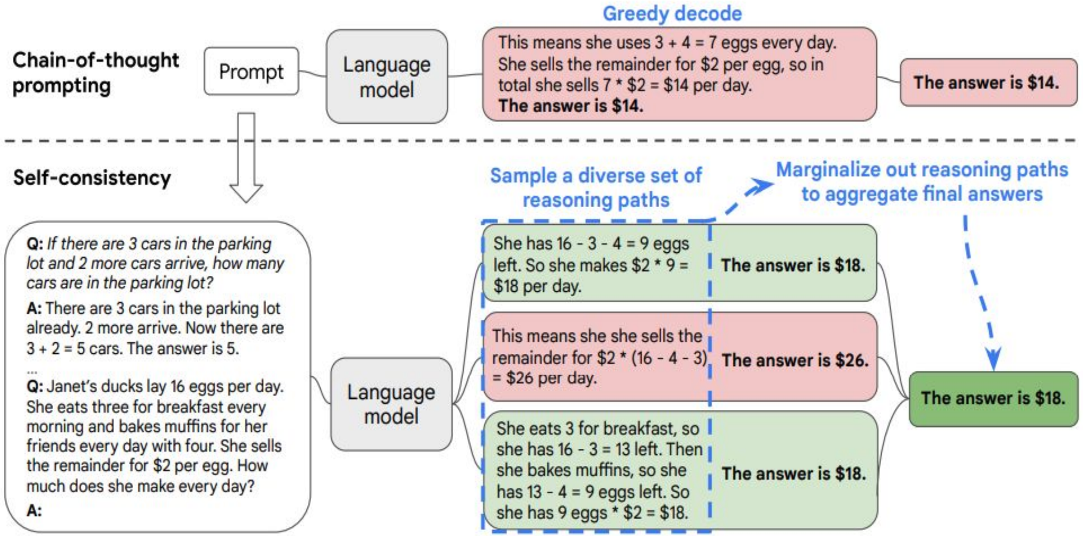


Planning

- Subgoals: breaks down large complex tasks into smaller, more manageable subtasks (i.e. planning without feedback)

- Reflections: self-criticizes and reflects over past actions, learn from mistakes and refine for future iterations (i.e. planning with feedbacks)

CoT and Self-Consistency


Greedy decode
Chain-of-thought prompting
Prompt
Language model
The answer is $14.
Self-consistency
Sample a diverse set of reasoning paths
Marginalize out reasoning paths to aggregate final answers
Language model
The answer is $18.

This means she uses 3 + 4 = 7 eggs every day. She sells the remainder for $2 per egg, so in total she sells 7 * $2 = $14 per day. **The answer is $14.**

Q: *If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?*
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.
...
Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder for $2 per egg. How much does she make every day?
A:

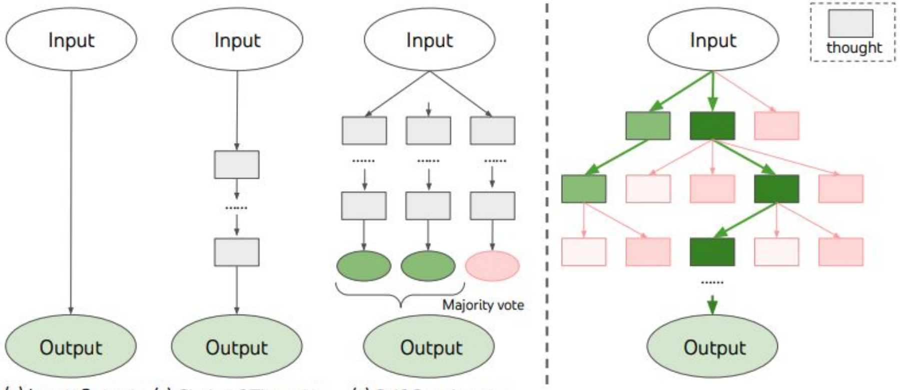
She has 16 - 3 - 4 = 9 eggs left. So she makes $2 * 9 = $18 per day. **The answer is $18.**

This means she she sells the remainder for $2 * (16 - 4 - 3) = $26 per day. **The answer is $26.**

She eats 3 for breakfast, so she has 16 - 3 = 13 left. Then she bakes muffins, so she has 13 - 4 = 9 eggs left. So she has 9 eggs * $2 = $18. **The answer is $18.**

- **Chain of Thought**
  - Instruct model to "Think step by step" to utilize more test-time compute to decompose complex tasks, making them more manageable
  - Improves interpretability of model's internal thinking process
- **Self-Consistency**
  - Replace the "greedy decoding" in CoT by sampling to generate a diverse set of reasoning paths
  - Choose the most consistent answer (majority vote)

ToT



(a) Input-Output Prompting (IO) (c) Chain of Thought Prompting (CoT) (c) Self Consistency with CoT (CoT-SC) **(d) Tree of Thoughts (ToT)**

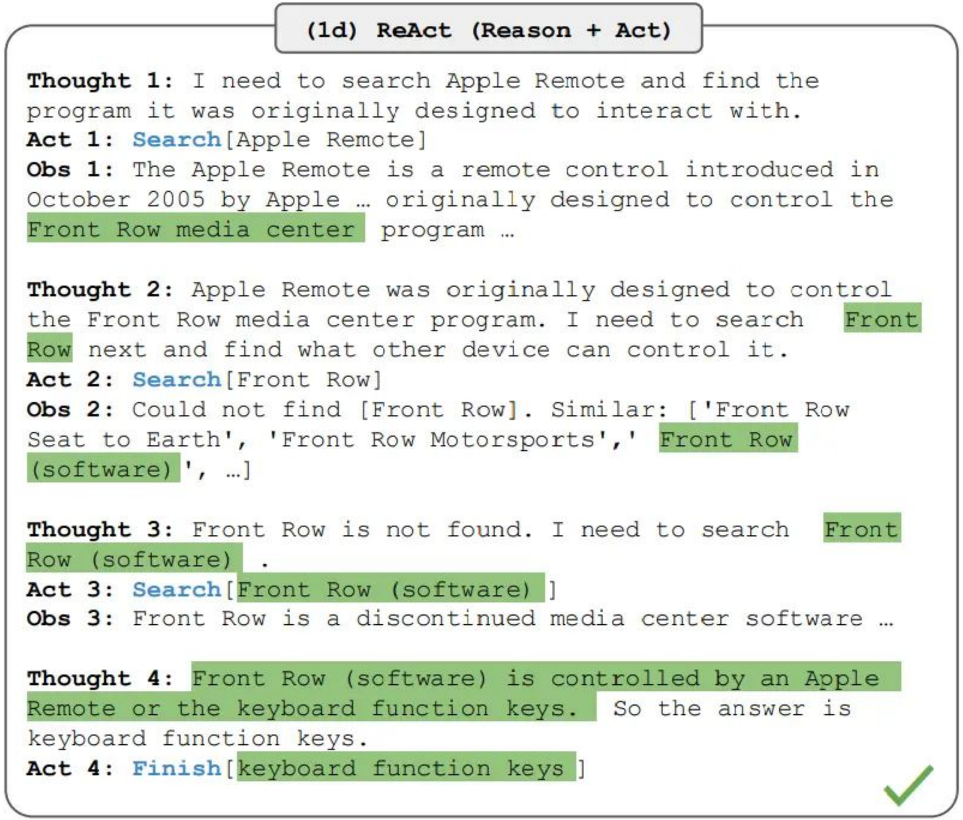
- **Tree of Thought**
  - Decomposes the problem into multiple thought steps
  - Generates multiple thoughts per step, creating a tree structure
  - Evaluate each state with a scalar value, classification / majority vote
  - Use BFS or DFS to navigate the solution space
  - Increase the capacity for complex reasoning by expanding and evaluating intermediate steps

Planning

- Subgoals: breaks down large complex tasks into smaller, more manageable subtasks (i.e. planning without feedback)

- **Reflections: self-criticizes and reflects over past actions, learn from mistakes and refine for future iterations (i.e. planning with feedbacks)**

ReAct: Synergizing Reasoning and Acting in Language Models



(1d) ReAct (Reason + Act)

```
Thought 1: I need to search Apple Remote and find the
program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in
October 2005 by Apple … originally designed to control the
Front Row media center program …

Thought 2: Apple Remote was originally designed to control
the Front Row media center program. I need to search Front
Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row
Seat to Earth', 'Front Row Motorsports',' Front Row
(software) ', …]

Thought 3: Front Row is not found. I need to search Front
Row (software) .
Act 3: Search[Front Row (software) ]
Obs 3: Front Row is a discontinued media center software …

Thought 4: Front Row (software) is controlled by an Apple
Remote or the keyboard function keys. So the answer is
keyboard function keys.
Act 4: Finish[keyboard function keys ]
```

✓

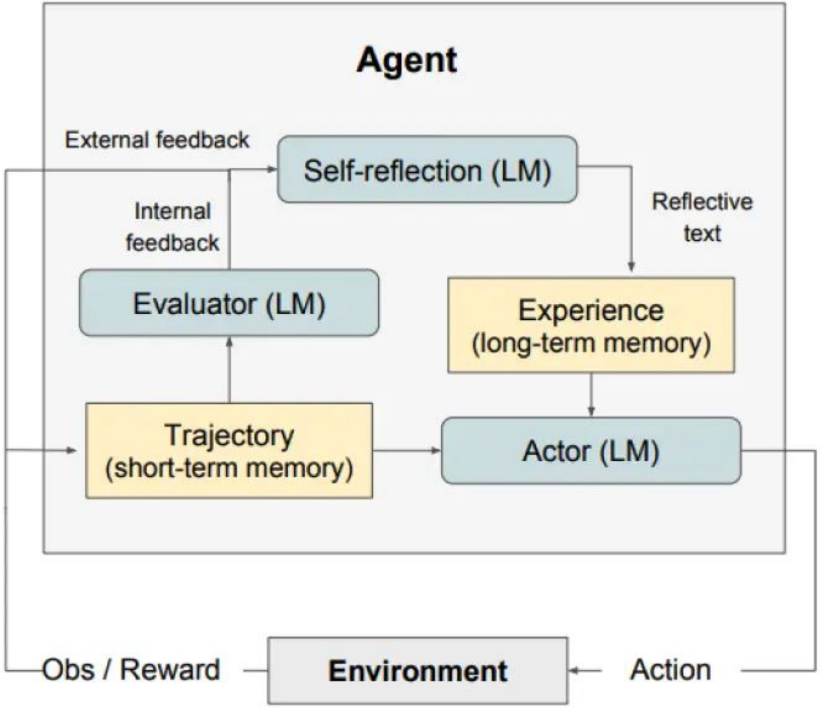
- Structured prompt with explicit step for LLM to think and act
  - Thought
  - Action
- Observations from environment serve as feedback to the agent
- Integrates reasoning and acting: Action space = task-specific discrete actions + language space
- Interleaving thoughts and actions
- Combines internal reasoning and external feedbacks

Memory

- Short-term / working memory: in-context learning / prompt engineering
    - finite due to context window constraints

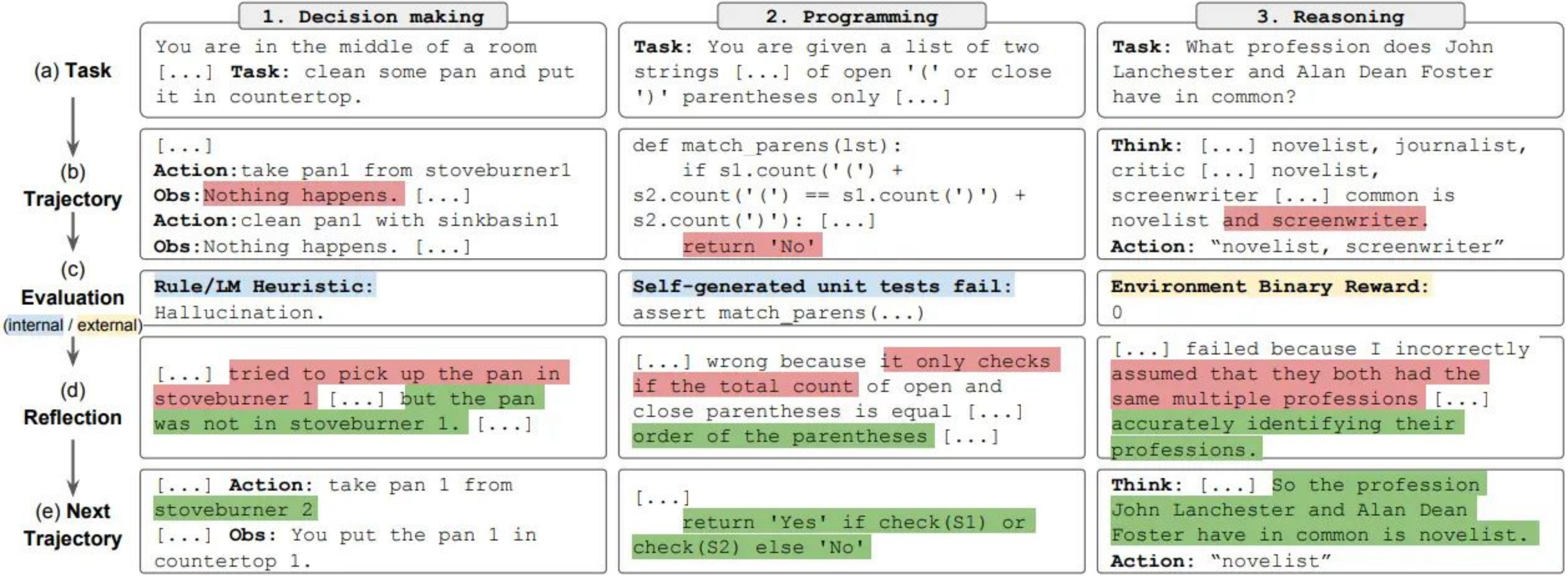
- Long-term memory: agent can retain and recall information over extended periods, leveraging external vector DB and fast retrieval

Reflexion



- Converts feedback (language or scalar) from the environment into linguistic feedback (self-reflection), provided as context for the agent in the next episode
- **Core components**
    - **Actor**: CoT, ReAct
    - **Evaluator**: trajectory -> reward score
    - **Self Reflection**: generate verbal reinforcement cues to improve the Actor
    - Memory:
        - short-term: current trajectory history
        - long-term: outputs from self-reflection (lessons learned over several trials)

Reflexion

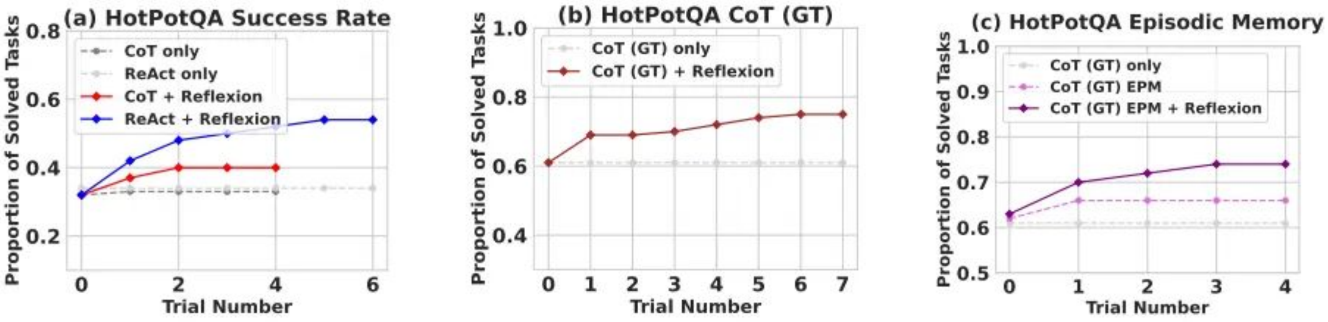


	1. Decision making	2. Programming	3. Reasoning
(a) **Task**	You are in the middle of a room [...] **Task:** clean some pan and put it in countertop.	**Task:** You are given a list of two strings [...] of open '(' or close ')' parentheses only [...]	**Task:** What profession does John Lanchester and Alan Dean Foster have in common?

(b) **Trajectory**	[...]
	Action:take pan1 from stoveburner1
	Obs: Nothing happens. [...]
	Action:clean pan1 with sinkbasin1
	Obs: Nothing happens. [...]
		def match_parens(lst):
		    if s1.count('(') + s2.count('(') == s1.count(')') + s2.count(')'): [...]
		    return 'No'
			Think: [...] novelist, journalist, critic [...] novelist, screenwriter [...] common is novelist and screenwriter.
			Action: "novelist, screenwriter"

(c) **Evaluation** (internal / external)	**Rule/LM Heuristic:** Hallucination.	**Self-generated unit tests fail:** assert match_parens(...)	**Environment Binary Reward:** 0
(d) **Reflection**	[...] tried to pick up the pan in stoveburner 1 [...] but the pan was not in stoveburner 1. [...]	[...] wrong because it only checks if the total count of open and close parentheses is equal [...] order of the parentheses [...]	[...] failed because I incorrectly assumed that they both had the same multiple professions [...] accurately identifying their professions.
(e) **Next Trajectory**	[...] **Action:** take pan 1 from stoveburner 2 [...] **Obs:** You put the pan 1 in countertop 1.	[...] return 'Yes' if check(S1) or check(S2) else 'No'	**Think:** [...] So the profession John Lanchester and Alan Dean Foster have in common is novelist. **Action:** "novelist"

Reflexion



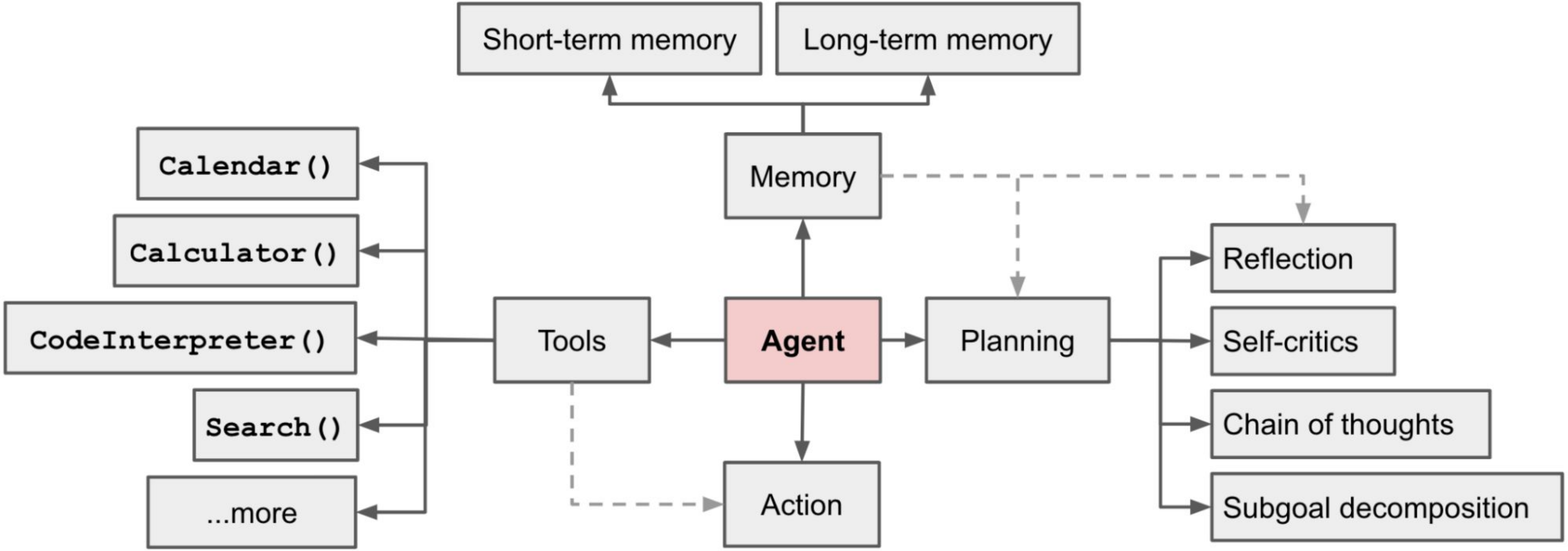
Advantages:

- The agent can learn from trial and error, reflecting on past mistakes for future decisions. e.g. decision-making, reasoning, programming etc.
- A lightweight alternative to traditional RL, which requires training data and expensive model parameter fine-tuning
- Verbal feedback is more nuanced and specific than a scalar reward in traditional RL, helping agent to better understand its mistakes
- Memory containing the reflections is more explicit and interpretable for understanding where the learning comes from

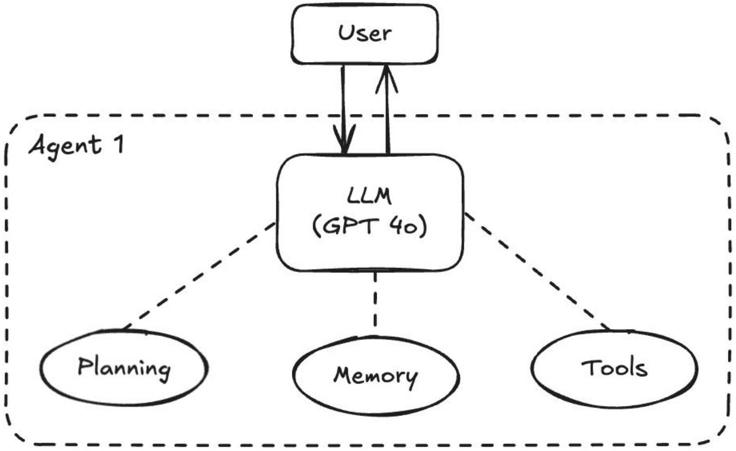
Tool Use 🔧

- a.k.a. function calling
- allows an agent to perform actions, interacting with the external environments such as using:
  - search API
  - code interpreter
  - math engine
  - knowledge bases
  - github access
  - ...

Agent in a Nutshell



Single Agent Challenges



- An agent can have **too many tools** to choose from and make poor decision about which tool to call next

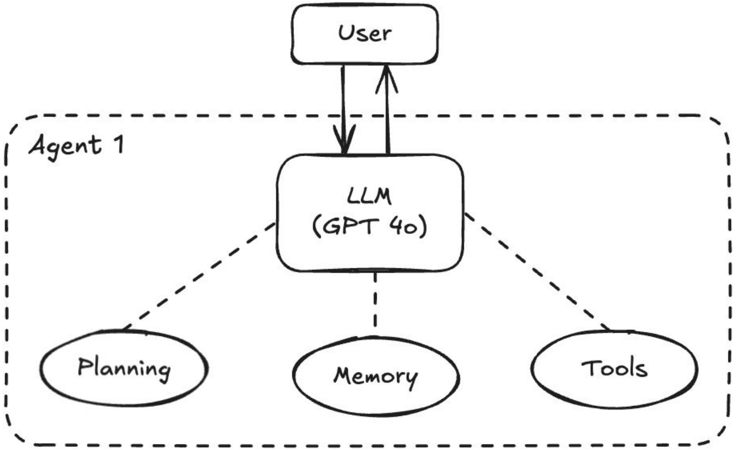
- **Context grows too complex** for single agent to keep track of (e.g. too many reflections of experiences with different tasks)

- There can be a need for multiple specialization areas in the system (e.g. planner, researcher, code expert etc.) to **make optimization of individual capabilities easier**

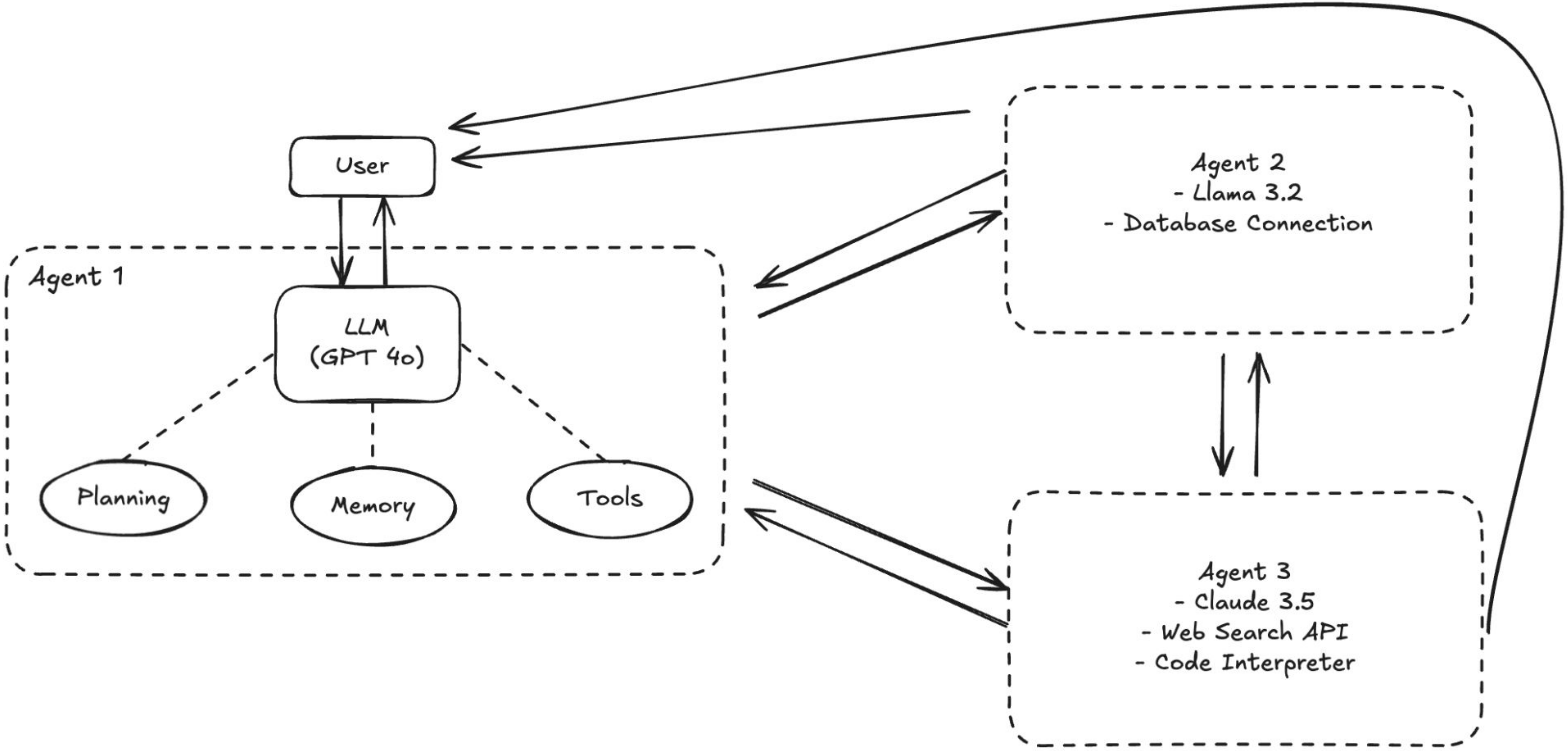
Why Multi-Agent Systems?

- Modularity
  - Separate agents are easier to develop, test, and maintain
- Specialization
  - Create experts focused on specific domains that helps the overall system performance
- Control
  - Explicitly control how agents communicate, compared to relying an agent's decision on function and tool calling

Multi-Agent Systems



Multi-Agent Systems



Multi-Agent Systems History

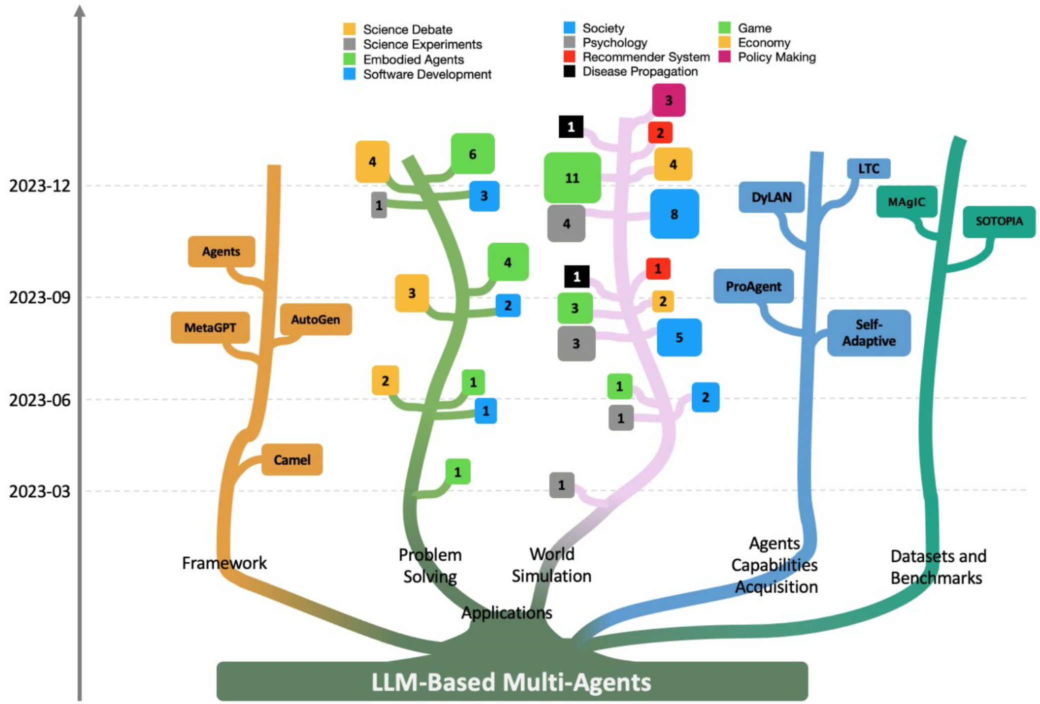


Figure 1: The rising trend in the research field of LLM-based Multi-Agents. For Problem Solving and World Simulation, we categorize current work into several categories and count the number of papers of different types at 3-month intervals. The number at each leaf node denotes the count of papers within that category.

Aspects of Multi-Agent Systems

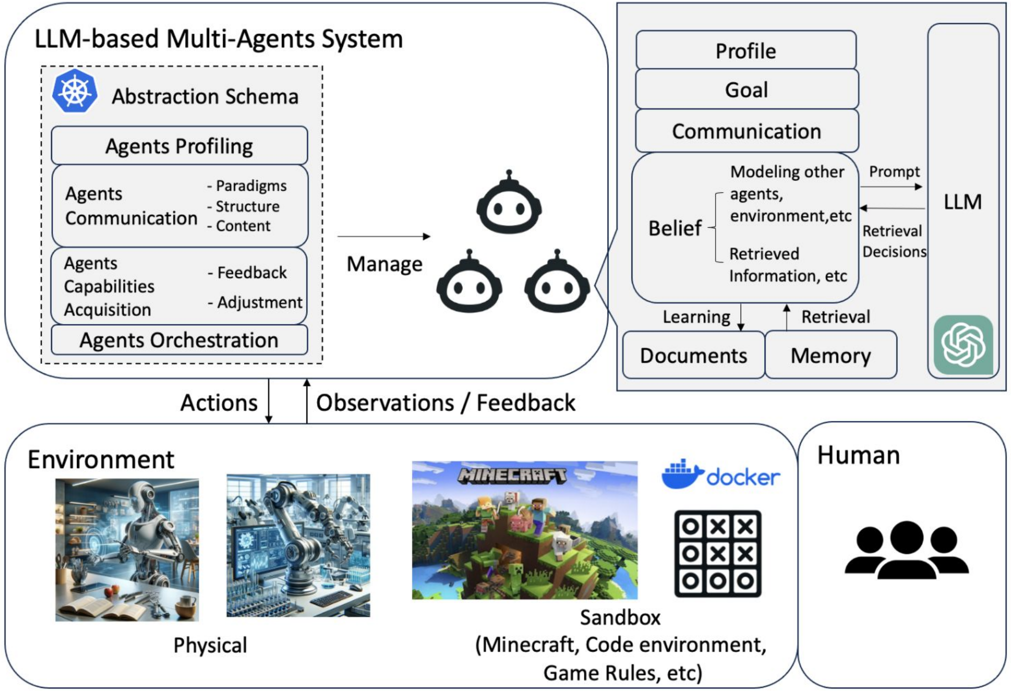


Figure 2: The Architecture of LLM-MA Systems.

Multi-Agent Communication

- **Cooperative**: agents work together towards a shared objective, and exchange information to enhance a collective solution

- **Debate**: agents engage in argumentative interactions, presenting and defending their own viewpoints or solutions, and critiquing others

- **Competitive**: agents working towards their own goals that might be in conflict with goals of other agents

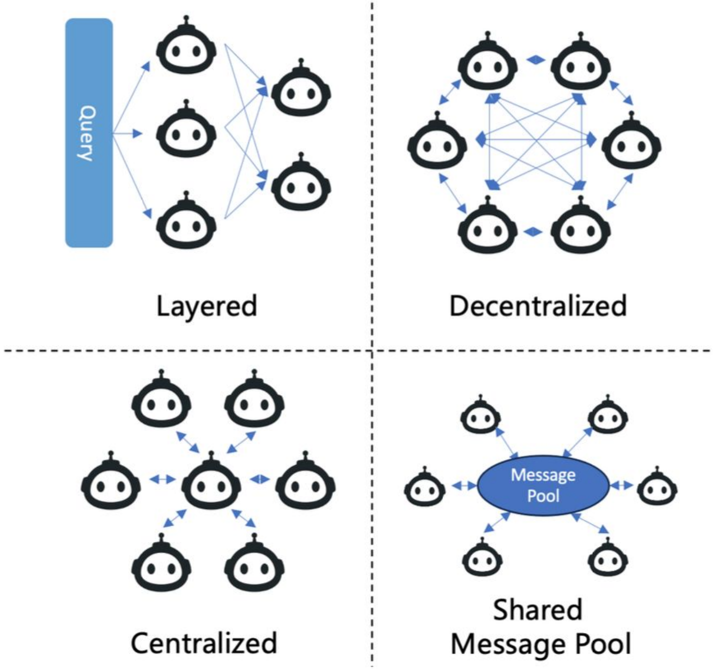


Figure 3: The Agent Communication Structure.

Agents as Graphs

GPTSwarm: Language Agents as Optimizable Graphs

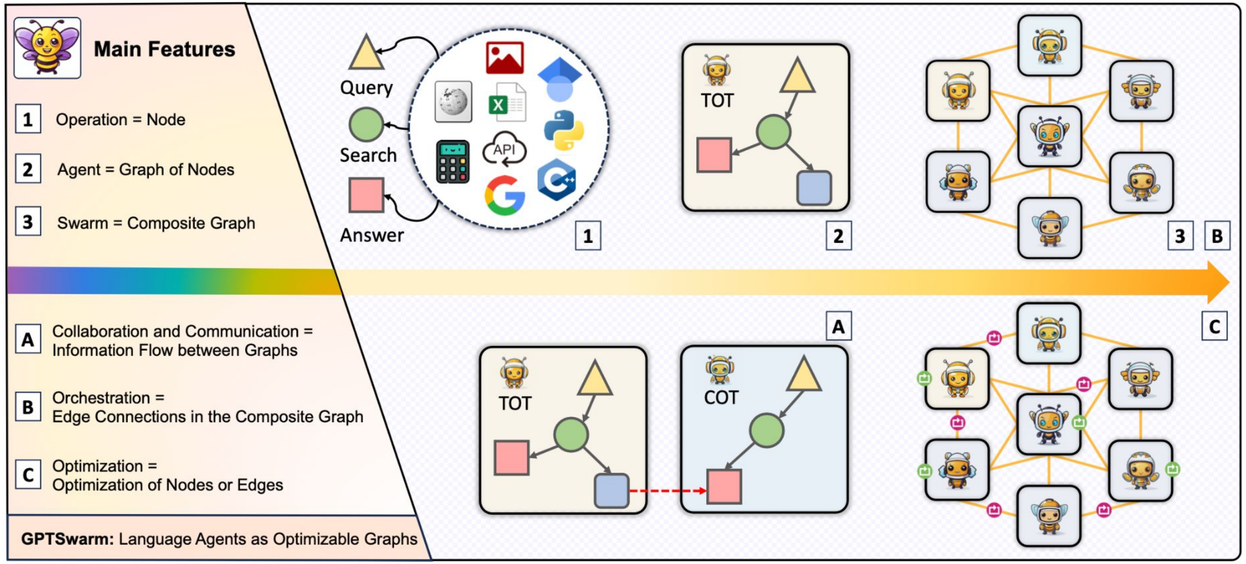

Main Features
1 Operation = Node
2 Agent = Graph of Nodes
3 Swarm = Composite Graph
A Collaboration and Communication = Information Flow between Graphs
B Orchestration = Edge Connections in the Composite Graph
C Optimization = Optimization of Nodes or Edges
GPTSwarm: Language Agents as Optimizable Graphs
Query
Search
Answer
API
1
TOT
2
3
B
A
TOT
COT
C

Figure 1. **GPTSwarm is a framework that represents agents as graphs.** In this framework, each node represents an operation (e.g., LLM inference or tool use). An agent is a graph composed of these nodes. An edge between two agent graphs characterizes a communication channel; each agent collaborates with others through different channels. When connected, multiple agents form a composite graph with a certain orchestration topology. This graph representation lends itself to optimization of nodes and edges via prompting and evolutionary or reinforcement learning techniques.

Agent Frameworks

- CrewAI
- LangGraph from LangChain
- Autogen from Microsoft
- OpenAI Swarm

CrewAI

Pros:

- Straightforward and beginner-friendly setup
- Built on top of LangChain, an established framework with lots of of community support, documentation, and useful tools
- Clear documentation with working examples and YouTube tutorials

Cons:

- Customizing things can get more complex
- Reliance on LangChain adds another dependency to your project

LangGraph

Pros:

- Low-level framework that gives you fine-grained control over each element, making it powerful for complex workflows
- Seamless integration with the LangChain ecosystem (in particular, the very large number of available built-in tools)
- Checkpoints can capture the past and present states of the agent for monitoring and error recovery purposes
- LangGraph Studio, a new specialized IDE, lets you visualize, interact with, and debug your agent workflows

Cons:

- Documentation can be hit or miss, sometimes outdated or missing key details
- Not the easiest framework to work with initially

Microsoft Autogen

Pros:

- Agents can generate, fix, and run code in Docker containers
- Autogen Studio, a no-code tool, making it easier to manage and debug
- Great option if you like a visual approach
- Offers a lot of flexibility and plenty of conversation patterns to work with
- Clear documentation with useful examples

Cons:

- Works best with the latest models; older ones may struggle with tasks
- May have a bit of a learning curve

OpenAI Swarm

Pros:

- Clean and easy-to-work-with structure, making it perfect for experimentation
- Simplest, cleanest, and most lightweight of the options

Cons:

- Still experimental and not meant for production use
- Does not work with the assistance API
- Limited features and capabilities compared to other frameworks
- Not a lot of community backing or support

References

Papers

Tree of Thoughts: https://arxiv.org/abs/2305.10601

ReAct: https://arxiv.org/abs/2210.03629

Reflexion: https://arxiv.org/abs/2303.11366

LLM-Based Agents Survey: https://arxiv.org/pdf/2402.01680

GPTSwarm: https://arxiv.org/pdf/2402.16823

YouTube

https://www.youtube.com/watch?v=pEMhPBQMNjg&ab_channel=SamWitteveen

https://www.youtube.com/watch?v=hvAPnpSfSGo&t=1s&ab_channel=LangChain

https://www.youtube.com/watch?v=4nZl32FwU-o&ab_channel=LangChain

Others

https://www.youtube.com/watch?v=q1XFm21I-VQ&t=369s&ab_channel=SnowflakeDevelopers

https://lilianweng.github.io/posts/2023-06-23-agent/

https://www.promptingguide.ai/research/llm-agents

https://developer.nvidia.com/blog/introduction-to-llm-agents/

https://blog.dataiku.com/open-source-frameworks-for-llm-powered-agents

LangGraph Demo