

95-891:

Introduction to Artificial Intelligence

Session 16: AI Agents

Yubo Li

yubol@andrew.cmu.edu

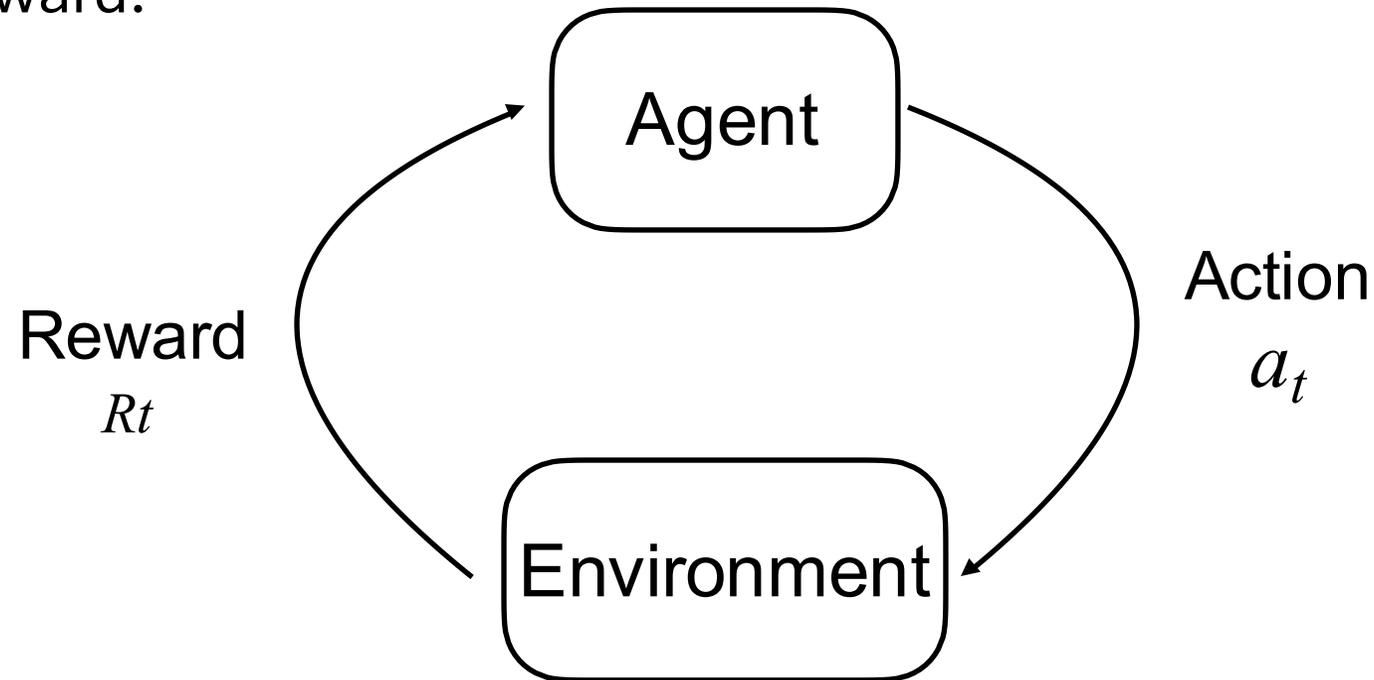
Oct. 28, 2025

Agenda

- LLM vs LLM-based AI Agent
- AI Agent Applications
- LLM Reasoning
- AI Agent Main Components
- System Architectures (single agent/multi agents)
- Challenges

Agent in RL

The reinforcement learning framework is an interaction between an **agent** and an **environment**, where the agent observes state(feedbacks), takes actions according to a policy(what has been learned), and aims to maximize cumulative reward.



Limitations of the Agent in RL

- One Task specific, lack of general capabilities.
- Credit assignment (temporal assignment/value issue)



LLM Recap



No interaction with environments!

Coding with LLMs

 Help me build a calculator in Java.

(java code)



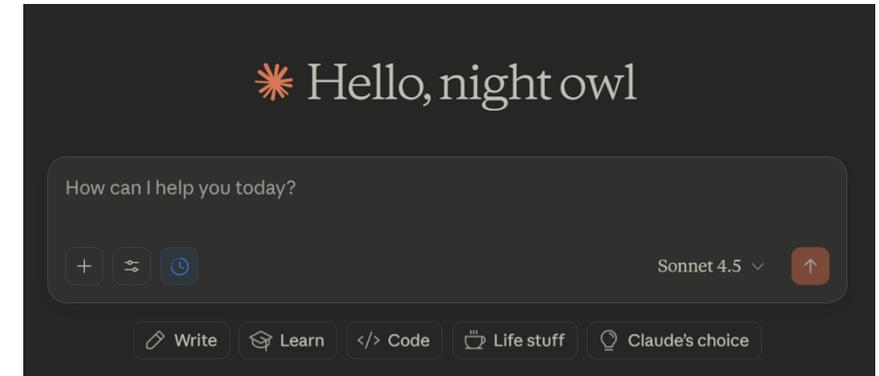
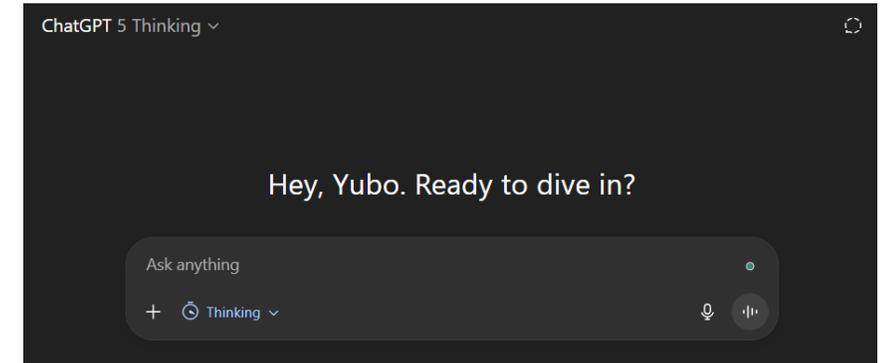
 Find a bug in the provided code.

(Paste the error to the chat window).

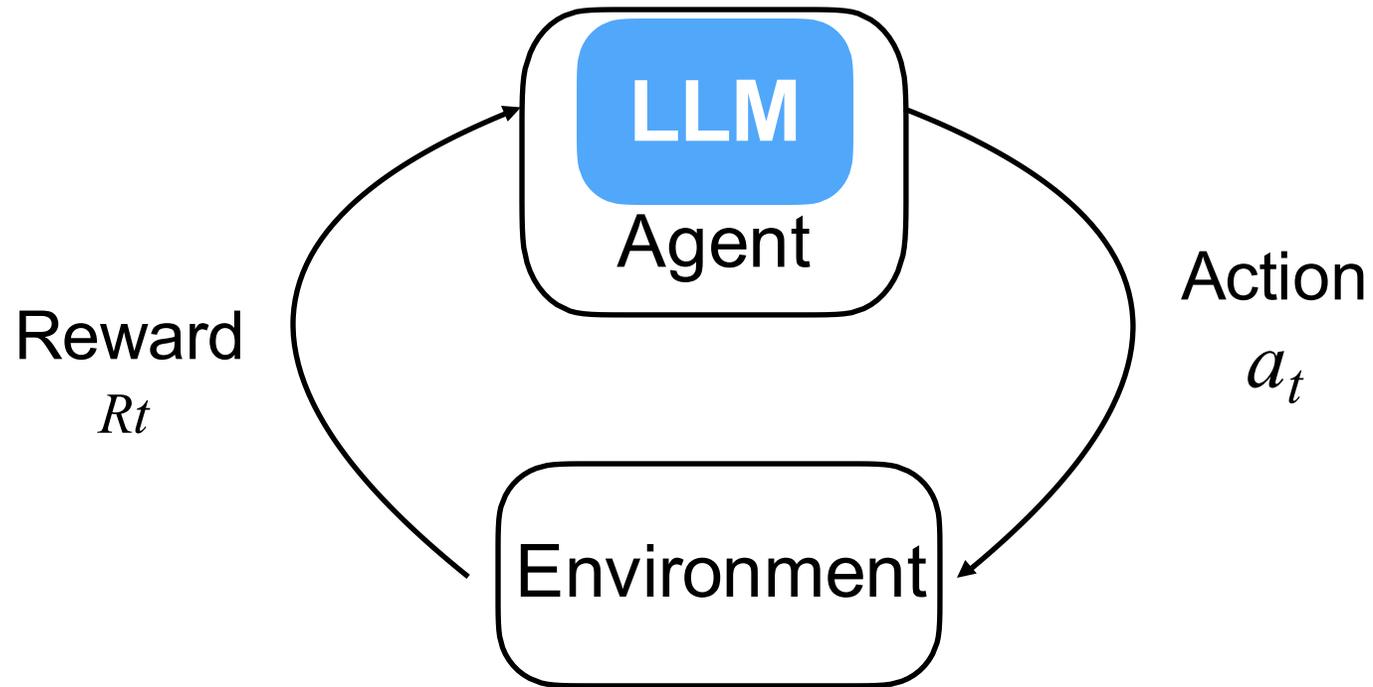
read the error and detect the issue,
then provide the updated java code



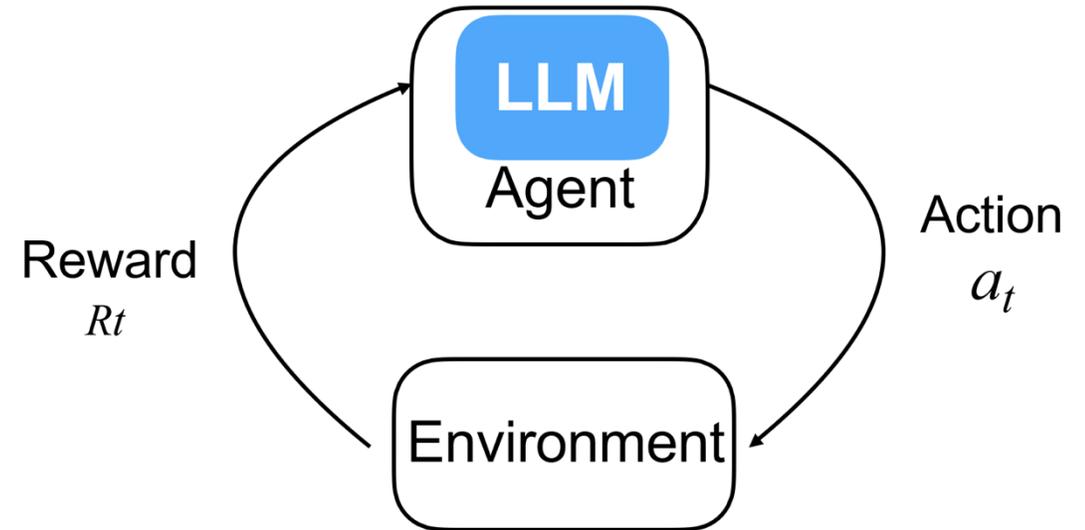
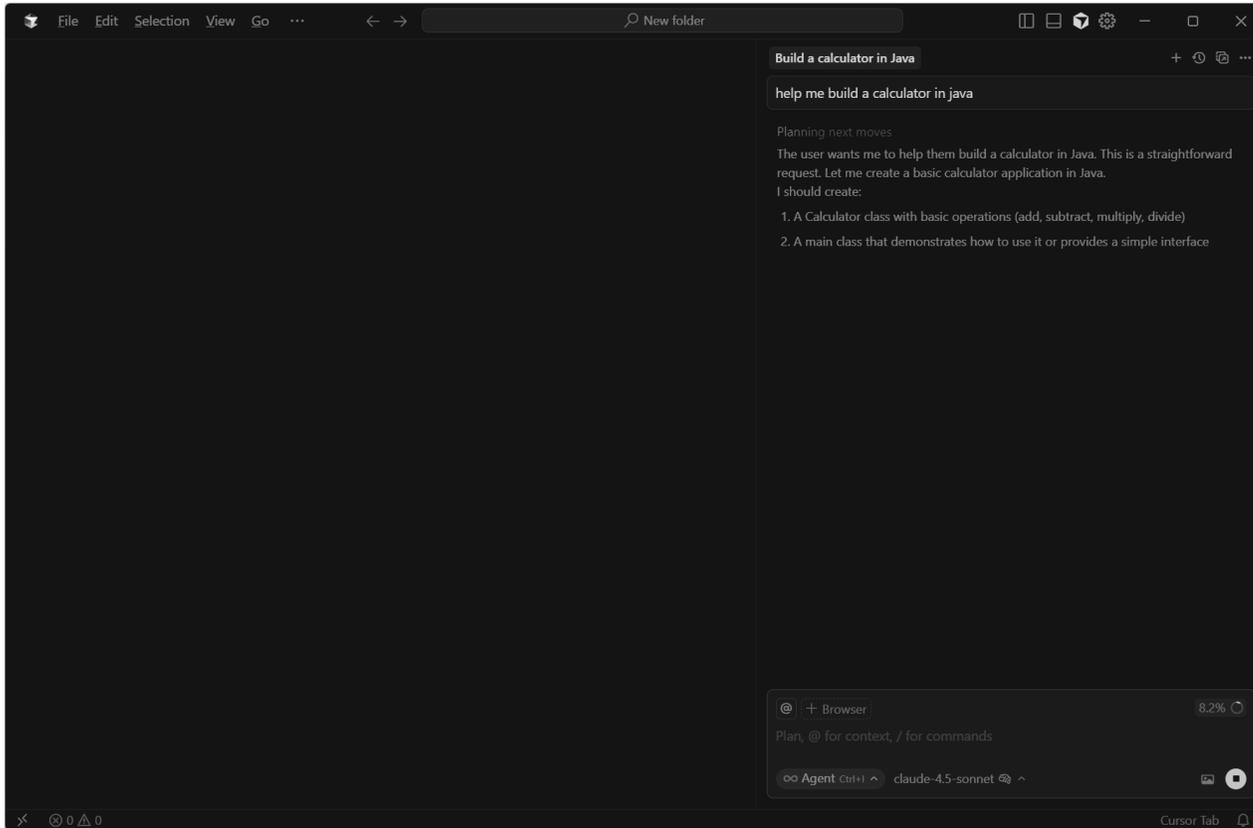
... (after many turns) ...



AI Agent = LLM + Agent

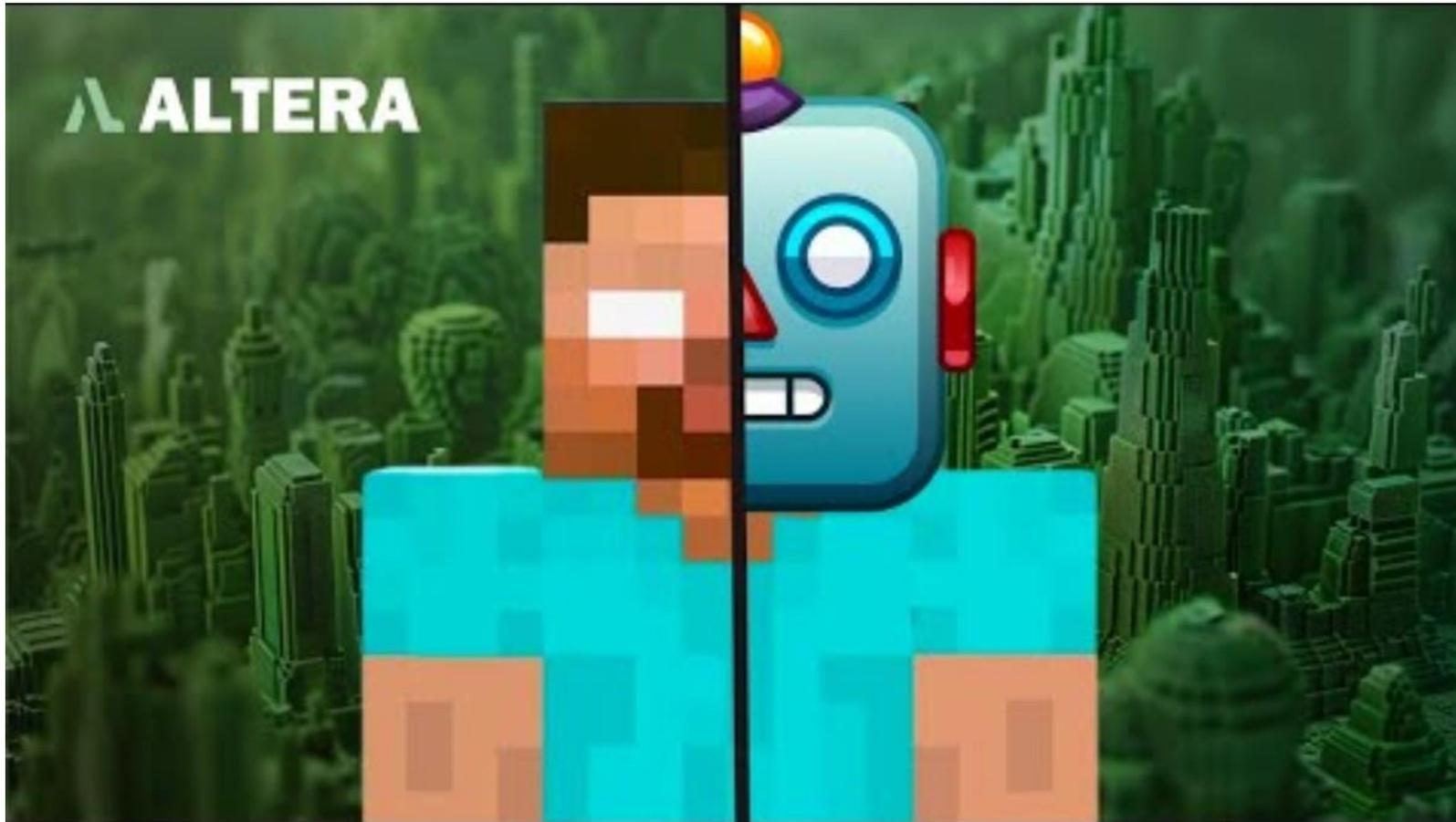


Coding with AI Agent - Cursor

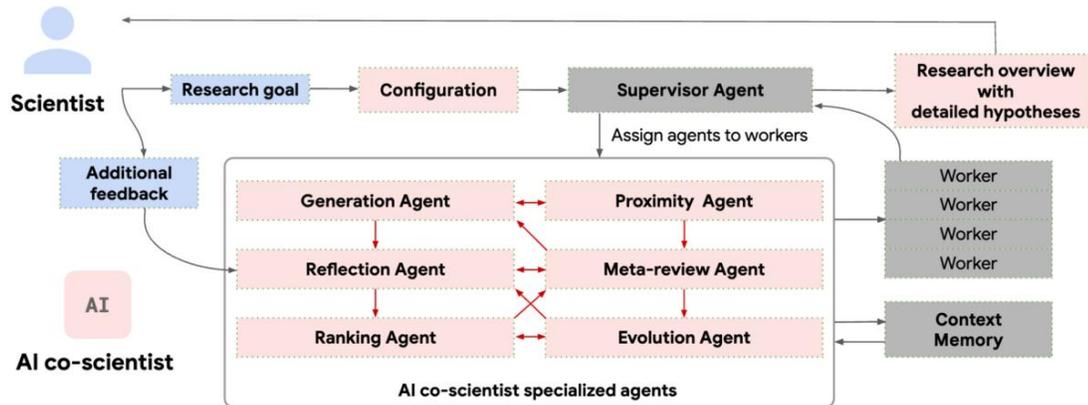


Gaming with AI Agent– Minecraft Civilization

The first simulations of 1000+ truly autonomous agents collaborating in a virtual world, w/ emergent economy, culture, religion, and government.



AI Agent Applications - Researcher

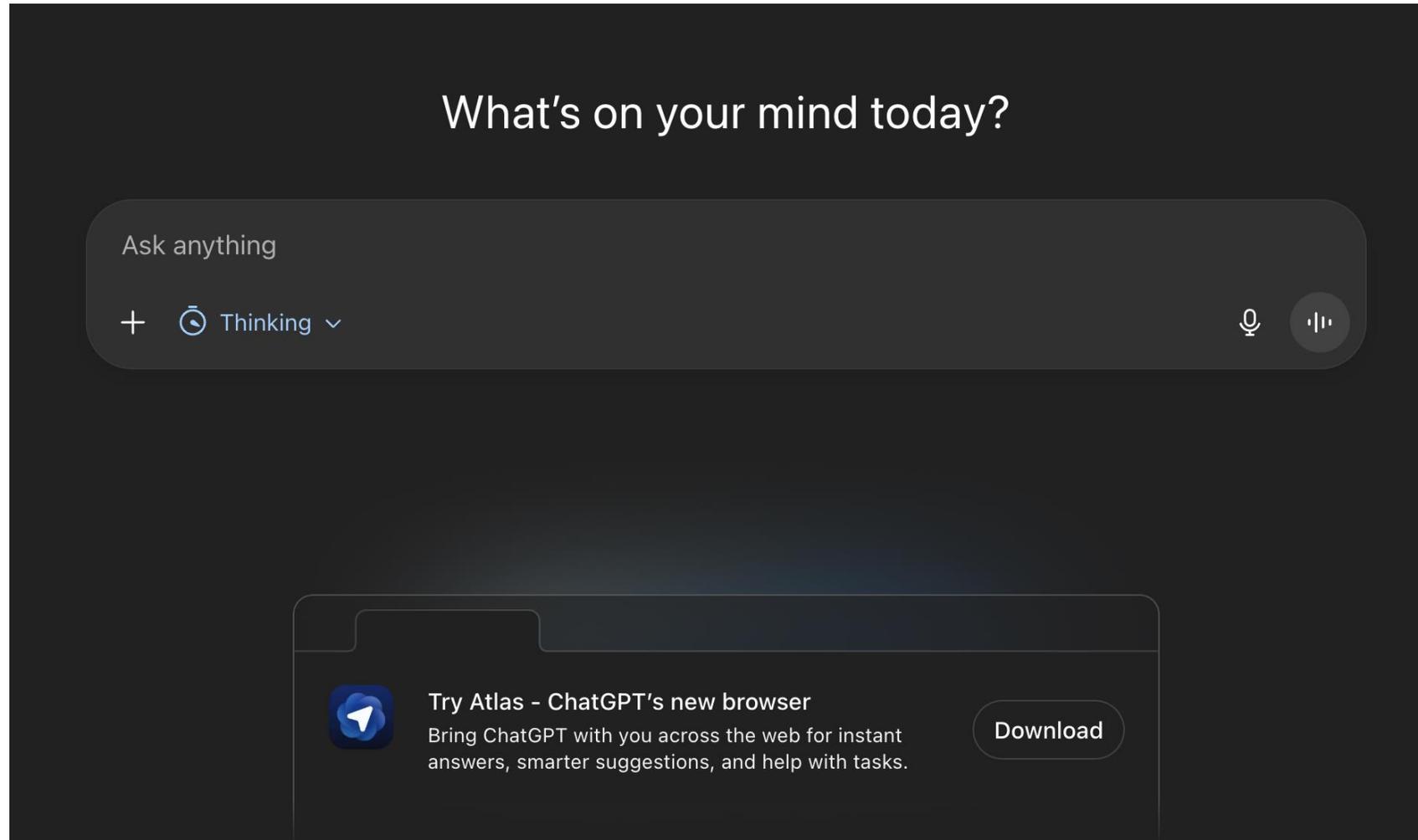


Google AI co-scientist



Agents4Science Conference by Stanford

General AI Agent – OpenAI Atlas/Manus



General AI Agent – OpenAI Atlas



ChatGPT Atlas

AI Agent Applications

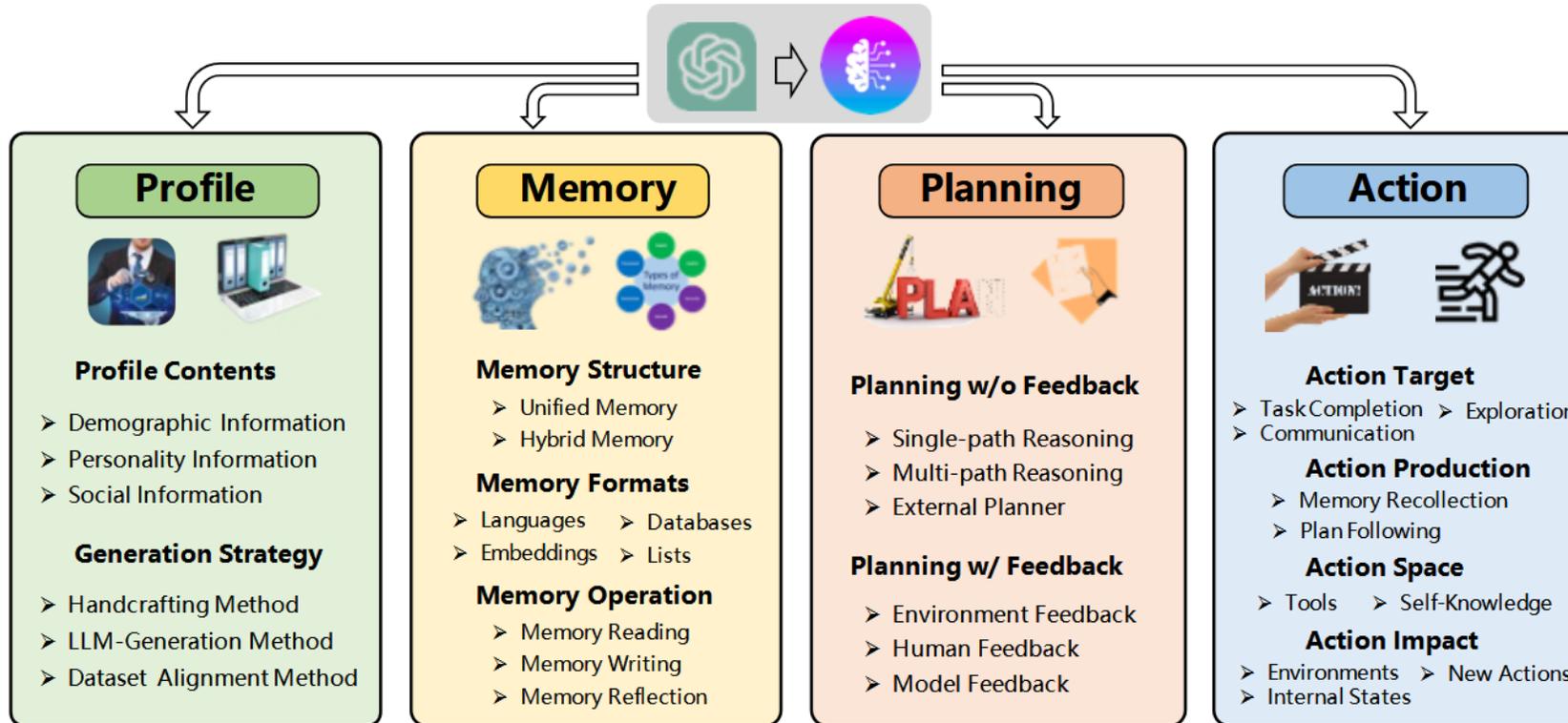
AI agents are rapidly adapting to diverse domains while demonstrating the ability to handle progressively more complex workflows.

What makes all these magic happen?

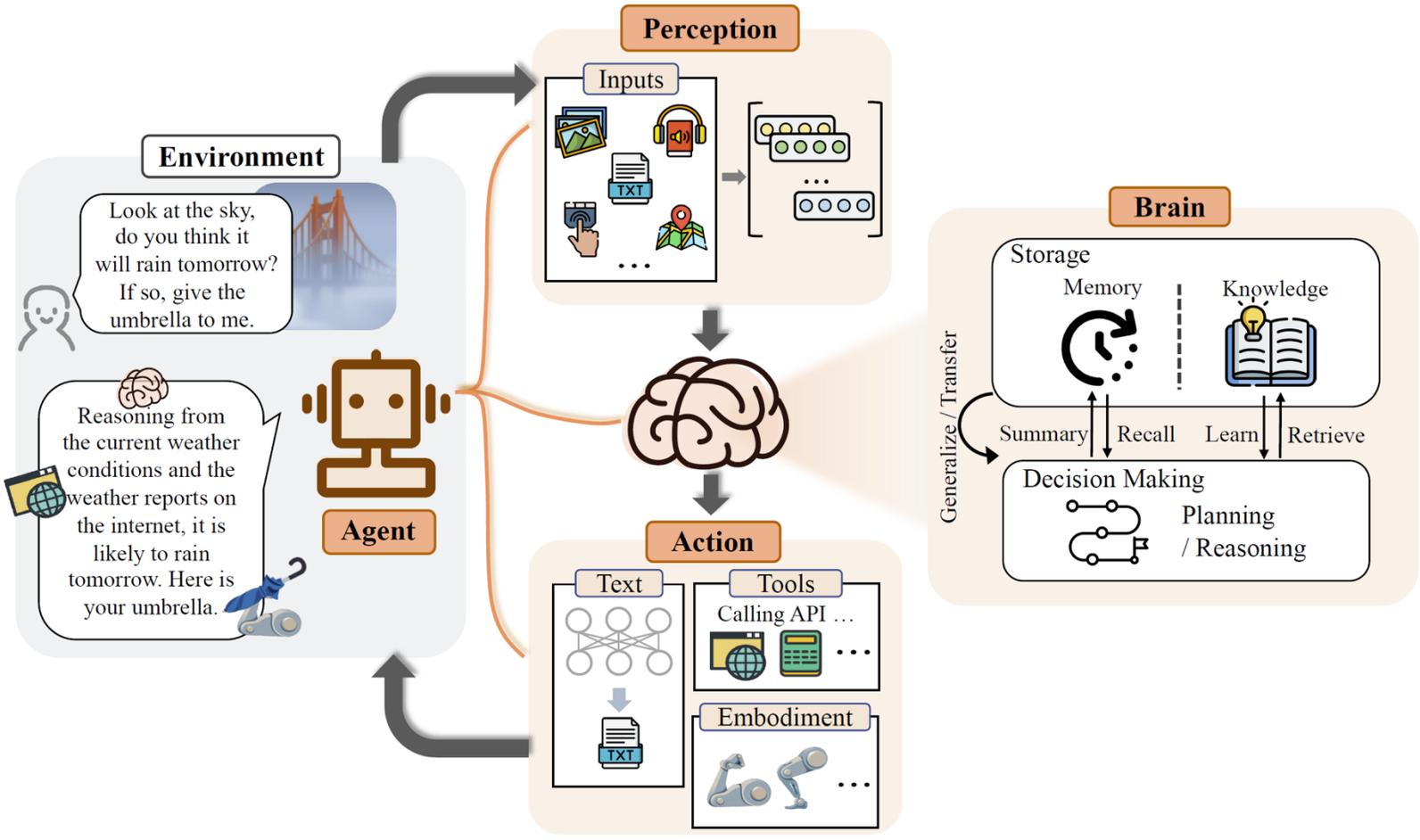


Main Components of AI Agents

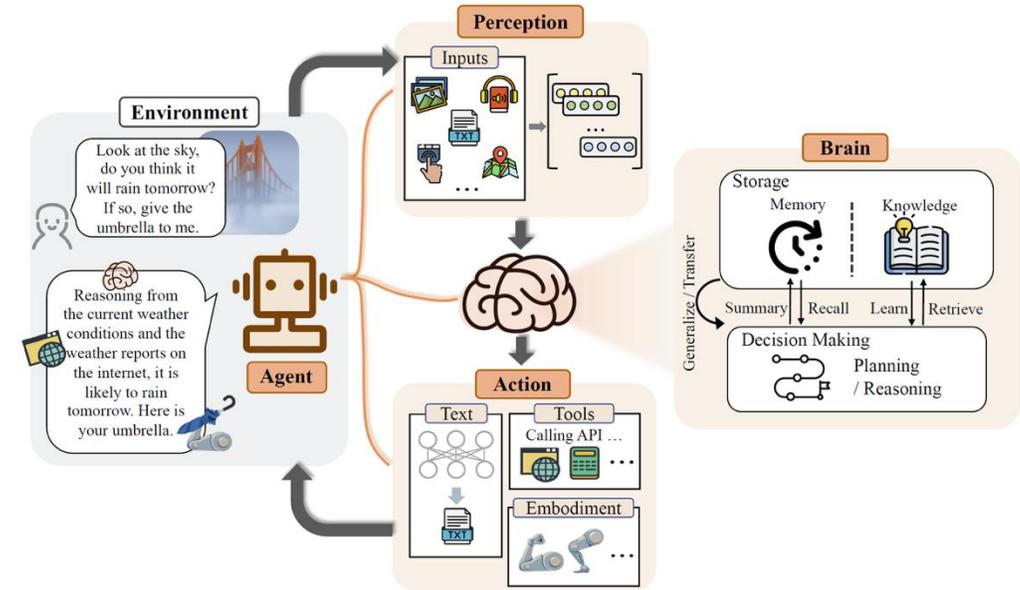
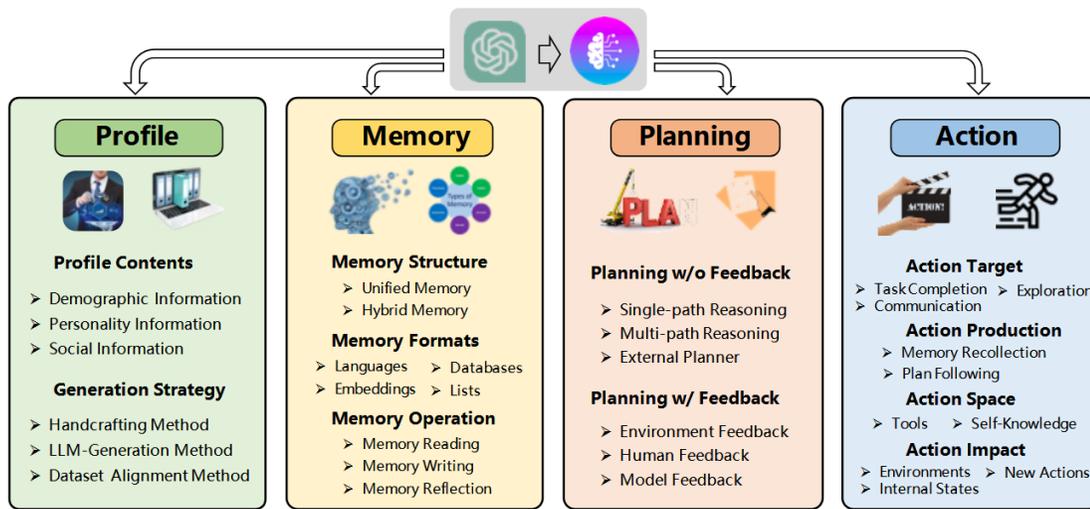
AI Agent Internals



AI Agent Internals



AI Agent Internals Comparison

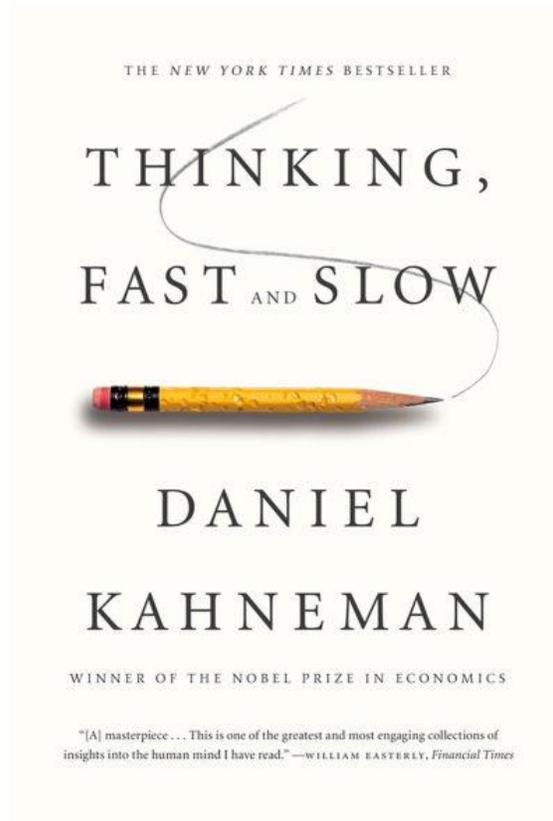


Q: Which component of AI Agent is missing? Potential reason?

The origin: LLM Reasoning



Thinking Fast & Slow



- **System 1:**
Fast, automatic, pattern-based, intuitive. It just “spits out an answer.”
- **System 2:**
Slow, deliberate, step-by-step, logical. It thinks, checks, plans, corrects.

LLM Reasoning

- Chain-of-Thought (CoT): A coherent series of intermediate reasoning steps that lead to the final answer for a problem.

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

LLM Reasoning – Let’s think step by step

LLMs are decent **zero-shot reasoners** by simply adding “Let’s think step by step” before each answer.

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) *The answer is 8.* ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) *The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4.* ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) *8* ✗

(d) Zero-shot-CoT (Ours)

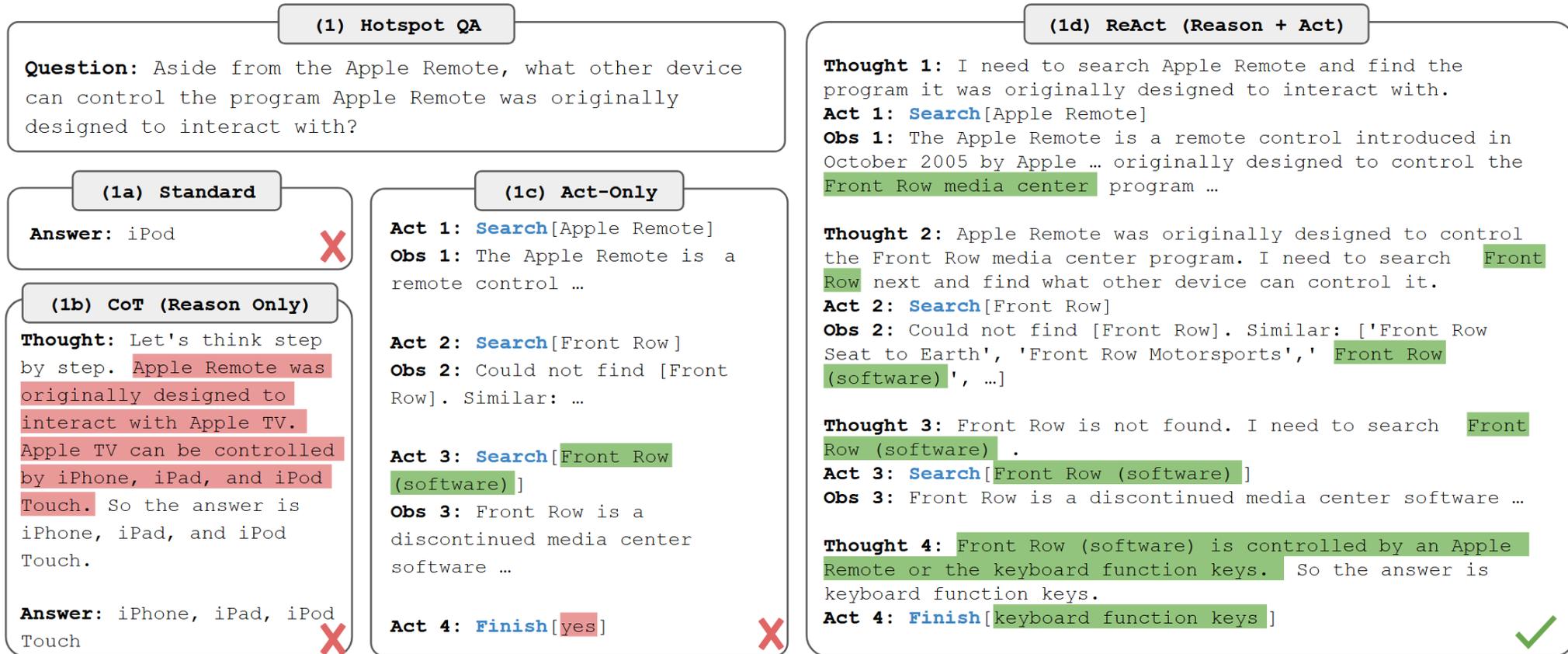
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let’s think step by step.**

(Output) *There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls.* ✓

Interaction Mode - ReAct

- ReAct: Synergizing Reasoning + Acting.



Interaction Mode – Plan&Act

arXiv > cs > arXiv:2502.01390

Computer Science > Human-Computer Interaction

[Submitted on 3 Feb 2025]

Plan-Then-Execute: An Empirical Study of User Trust and Team Performance When Using LLM Agents As A Daily Assistant

Gaole He, Gianluca Demartini, Ujwal Gadiraju

Since the explosion in popularity of ChatGPT, large language models (LLMs) have continued to impact our everyday lives. Equipped with external tools that are designed for a specific purpose (e.g., for flight booking or an alarm), LLMs provide a limited understanding of how they can provide daily assistance based on planning and sequential decision making capabilities. We draw inspiration from recent work that has highlighted the value of 'LLM-modulo' setups in different levels of risk typically associated with them (e.g., flight ticket booking and credit card payments). To ensure user agency and control over the LLM agent, we adopted LLM agents in a plan-then-execute manner, where collaborative team performance. Our findings demonstrate that LLM agents can be a double-edged sword -- (1) they can work well when a high-quality plan and necessary user involvement in execution are available, and (2) they can achieve better overall task outcomes. Our work has important implications for the future design of daily assistants and human-AI collaboration with LLM agents.

arXiv > cs > arXiv:2503.09572

Computer Science > Computation and Language

[Submitted on 12 Mar 2025 (v1), last revised 22 Apr 2025 (this version, v3)]

Plan-and-Act: Improving Planning of Agents for Long-Horizon Tasks

Lufti Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, Amir Gholami

Large language models (LLMs) have shown remarkable advancements in enabling language agents to tackle simple tasks. However, applying them for complex, long-horizon tasks remains difficult since LLMs are not inherently trained for this task. To address this, we propose Plan-and-Act which generates structured, high-level plans to achieve user goals, and an Executor model that translates these plans into environment-specific actions. To train the using web navigation as a representative long-horizon planning environment, demonstrating a state-of-the-art 57.58% success rate on the WebArena-Lite benchmark.

LangChain

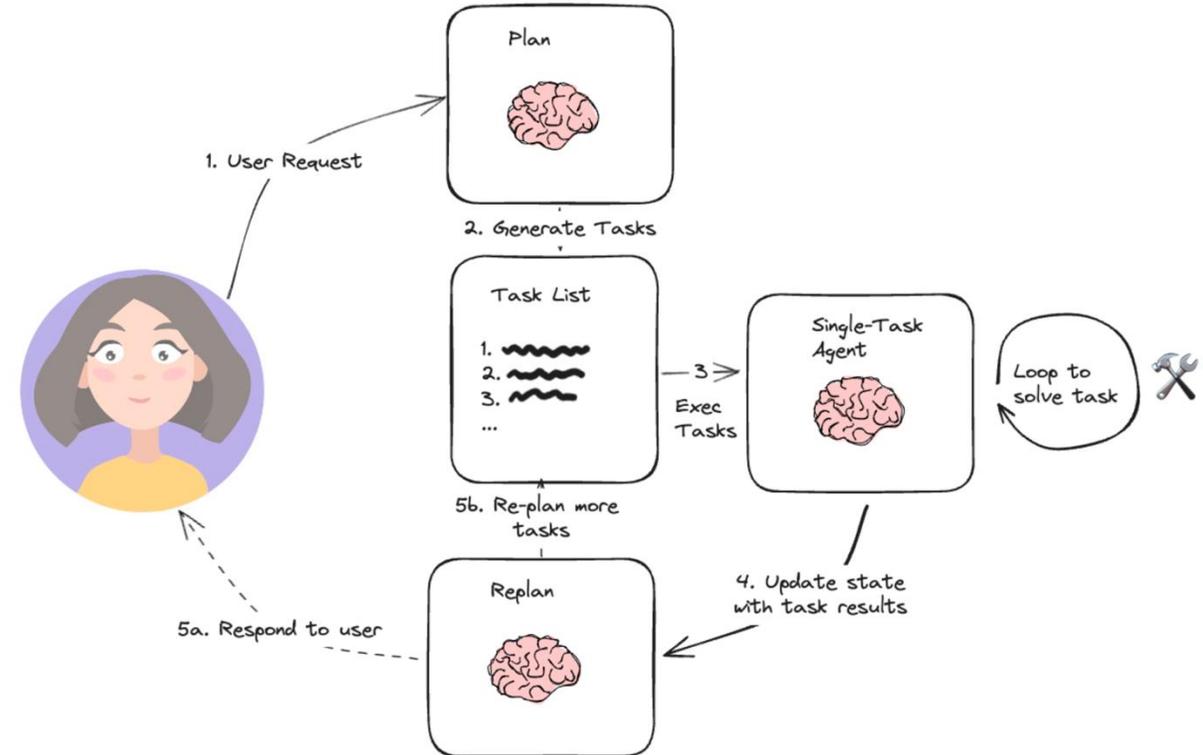
Case Studies In the Loop LangChain Docs Changelog Sign in Subscribe



Plan-and-Execute Agents

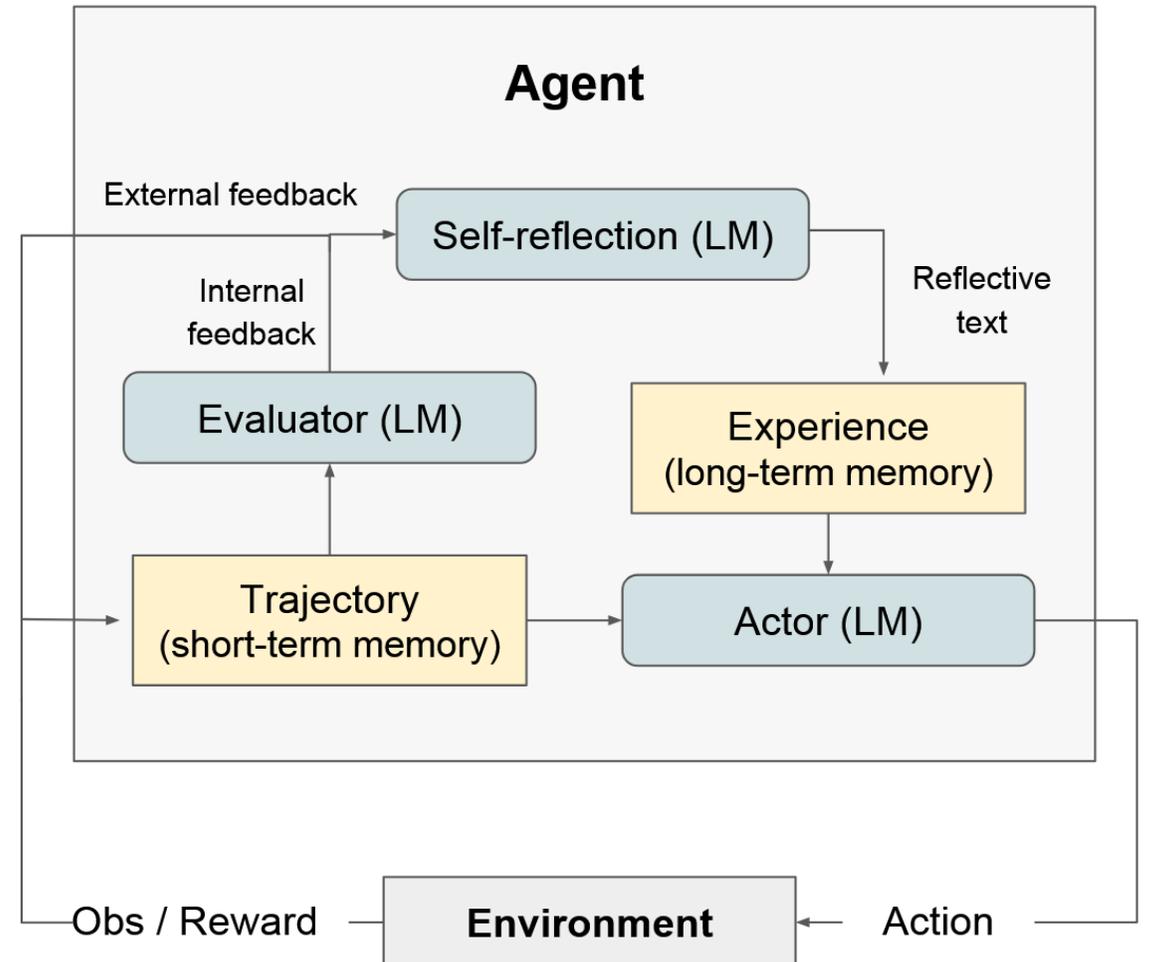
Plan and execute agents promise faster, cheaper, and more performant task execution over previous agent designs. Learn how to build 3 types of planning agents in LangGraph in this post.

BY LANGCHAIN 5 MIN READ FEB 13, 2024



Interaction Mode - Reflexion

- **Core Idea:** Reflexion is about enabling large language model agents to learn from their own verbal feedback rather than direct parameter updates.
- **Mechanism:** It uses a kind of **verbal reinforcement learning**, where the model reflects on mistakes and stores those reflections to improve future performance.



Planning

- LLMs are Zero-Shot Planners

Human

Task: Throw away paper
Step 1: Walk to home office
Step 2: Walk to desk
Step 3: Find desk
Step 4: Turn to desk
Step 5: Find chair
Step 6: Sit on chair
Step 7: Find check
Step 8: Grab check
Step 9: Squeeze check
Step 10: Stand up
Step 11: Walk to trashcan
Step 12: Put check on trashcan

GPT-2 1.5B

Task: Brush teeth
Step 1: Go to bathroom

GPT-3 175B

Task: Throw away paper
Step 1: Walk to home office
Step 2: Walk to wastebasket
Step 3: Find wastebasket
Step 4: Grab wastebasket
Step 5: Walk to desk
Step 6: Drop paper in wastebasket

Codex 12B

Task: Brush teeth
Step 1: Walk to bathroom
Step 2: Walk to sink
Step 3: Find toothbrush
Step 4: Pick up toothbrush
Step 5: Put toothbrush in mouth
Step 6: Move brush around mouth for two minutes
Step 7: Spit out toothpaste and brush into sink
Step 8: Turn on water in sink and rinse brush for one minute
Step 9: Turn off water in sink and return brush to cupboard

Translated GPT-3 175B

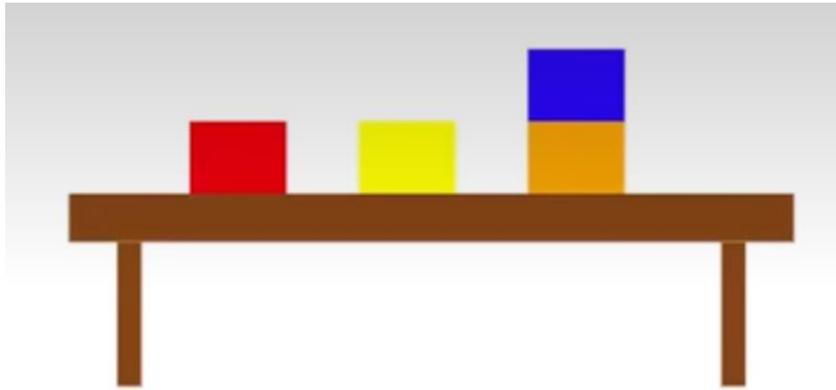
Task: Throw away paper
Step 1: Walk to home office
Step 2: Walk to table
Step 3: Find table
Step 4: Turn to table
Step 5: Find paper
Step 6: Grab paper
Step 7: Walk to trashcan
Step 8: Open trashcan
Step 9: Put paper on trashcan
Step 10: Close trashcan

Translated Codex 12B

Task: Brush teeth
Step 1: Walk to bathroom
Step 2: Open door
Step 3: Walk to sink
Step 4: Put pot on sink
Step 5: Put brush on toothbrush
Step 6: Turn to toothpaste
Step 7: Put toothpaste on toothbrush
Step 8: Put teeth on toothbrush



Planning - PlanBench



BlocksWorld

Domain	Method	Instances correct				
		GPT-4	GPT-3.5	I-GPT3.5	I-GPT3	GPT-3
Blocksworld (BW)	One-shot	206/600 (34.3%)	37/600 (6.1%)	54/600 (9%)	41/600 (6.8%)	6/600 (1%)
	Zero-shot	210/600 (34.6%)	8/600 (1.3%)	-	-	-
	COT	214/600 (35.6%)	-	-	-	-
Logistics Domain	One-shot	28/200 (14%)	1/200 (0.5%)	6/200 (3%)	3/200 (1.5%)	-
	Zero-shot	15/200 (7.5%)	1/200 (0.5%)	-	-	-
Mystery BW (Deceptive)	One-shot	26/600 (4.3%)	0/600 (0%)	4/600 (0.6%)	14/600 (2.3%)	0/600 (0%)
	Zero-shot	1/600 (0.16%)	0/600 (0%)	-	-	-
	COT	54/600 (9%)	-	-	-	-
Mystery BW (Randomized)	One-shot	12/600 (2%)	0/600 (0%)	5/600 (0.8%)	5/600 (0.8%)	1/600 (0.1%)
	Zero-shot	0/600 (0%)	0/600 (0%)	-	-	-

In 2022, LLMs are not planners at all!

Planning - PlanBench

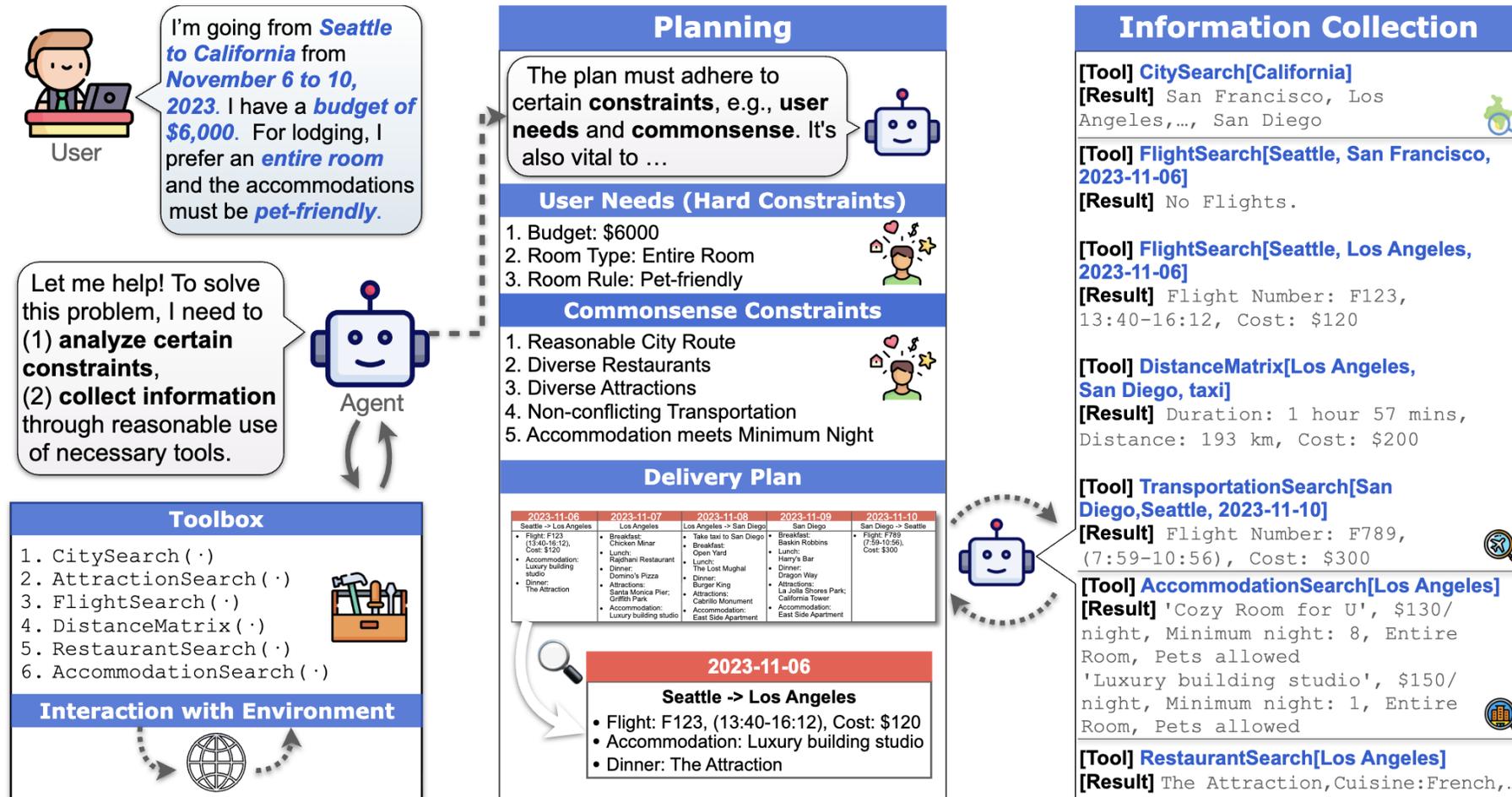
LLMS STILL CAN'T PLAN; CAN Large Reasoning Models?

Domain	Shots	Instances correct			Average Time Taken (in secs)		
		o1-preview	o1-mini	Fast Downward	o1-preview	o1-mini	Fast Downward
Blocksworld	Zero Shot	587/600 (97.8%)	340/600 (56.6%)	600/600 (100%)	40.43	10.84	0.265
Mystery Blocksworld	One Shot	247/600 (41.6%)	-	600/600 (100%)	82.03	-	0.265
	Zero Shot	317/600 (52.8%)	115/600 (19.1%)	600/600 (100%)	83.37	35.54	0.265
Randomized Mystery Blocksworld	Zero Shot	224/600 (37.3%)	-	600/600 (100%)	111.11	-	0.265

Large Reasoning Models perform better than LLMs as a planner.

<https://arxiv.org/abs/2409.13373>

Planning – TravelPlanner(2024)



Planning – TravelPlanner(2024)

	Validation (#180)						Test (#1,000)					
	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate
		Micro	Macro	Micro	Macro			Micro	Macro	Micro	Macro	
Greedy Search	100	74.4	0	60.8	37.8	0	100	72.0	0	52.4	31.8	0
<i>Two-stage</i>												
Mistral-7B-32K (Jiang et al., 2023)	8.9	5.9	0	0	0	0	7.0	4.8	0	0	0	0
Mixtral-8x7B-MoE (Jiang et al., 2024)	49.4	30.0	0	1.2	0.6	0	51.2	32.2	0.2	0.7	0.4	0
Gemini Pro (G Team et al., 2023)	28.9	18.9	0	0.5	0.6	0	39.1	24.9	0	0.6	0.1	0
GPT-3.5-Turbo (OpenAI, 2022)	86.7	54.0	0	0	0	0	91.8	57.9	0	0.5	0.6	0
GPT-4-Turbo (OpenAI, 2023)	89.4	61.1	2.8	15.2	10.6	0.6	93.1	63.3	2.0	10.5	5.5	0.6
<i>Sole-planning</i>												
Direct _{GPT-3.5-Turbo}	100	60.2	4.4	11.0	2.8	0	100	59.5	2.7	9.5	4.4	0.6
CoT _{GPT-3.5-Turbo}	100	66.3	3.3	11.9	5.0	0	100	64.4	2.3	9.8	3.8	0.4
ReAct _{GPT-3.5-Turbo}	82.2	47.6	3.9	11.4	6.7	0.6	81.6	45.9	2.5	10.7	3.1	0.7
Reflexion _{GPT-3.5-Turbo}	93.9	53.8	2.8	11.0	2.8	0	92.1	52.1	2.2	9.9	3.8	0.6
Direct _{Mixtral-8x7B-MoE}	100	68.1	5.0	3.3	1.1	0	99.3	67.0	3.7	3.9	1.6	0.7
Direct _{Gemini Pro}	93.9	65.0	8.3	9.3	4.4	0.6	93.7	64.7	7.9	10.6	4.7	2.1
Direct _{GPT-4-Turbo}	100	80.4	17.2	47.1	22.2	4.4	100	80.6	15.2	44.3	23.1	4.4

In 2024, TravelPlanner is too hard for LLMs.

<https://arxiv.org/abs/2402.01622>

Planning – TravelPlanner(2024)

Query: Please assist in crafting a travel plan for a solo traveller, journeying from Detroit to San Diego for 3 days, from March 5th to March 7th, 2022. The travel plan should accommodate a total budget of **\$3,000**.

Trajectory:

Action 1: CostEnquiry[{"people_number": 1, "day": 1,...}]

Observation 1: Cost: The cost of your plan is 1412.0 dollars.

Thought 2: The cost for the first day is \$1412, which includes the flight and activities in San Diego. The remaining budget is \$1588...

Action 2: CostEnquiry[{"people_number": 1, "day": 2,...}]

Observation 2: Cost: The cost of your plan is 721.0 dollars.

Thought 3: The cost for the second day is \$721, which includes meals, attractions, and the same accommodation as the first day. The total spent so far is \$2133, leaving a remaining budget of \$867.

Action 3: CostEnquiry[{"people_number": 1, "day": 3,...}]

Observation 3: Cost: The cost of your plan is 1114 dollars.

Thought 4: The cost for the third day is \$1114, which includes the return flight and activities in San Diego before departure. **The total cost for all three days is now \$3247, which exceeds the budget of \$3000.** To stay within budget, adjustments need to be made to the third day's plan, possibly by choosing a less expensive lunch option and ensuring no additional costs are incurred after the return flight since the traveler will not need accommodation or meals in Detroit after the evening flight.

Action 4: CostEnquiry[{"people_number": 1, "day": 3,...}]

Observation 4: Cost: The cost of your plan is 1105 dollars.

Thought 5: The revised cost for the third day is \$1105, which includes a more affordable lunch option. **The total cost for all three days is now \$3238, which is still slightly over the budget of \$3000.** To further reduce costs, I can look for a less expensive breakfast option on the third day...

Action 5: CostEnquiry[{"people_number": 1, "day": 3,..}]

Observation 5: Cost: The cost of your plan is 1083 dollars.

Thought 6: The revised cost for the third day is now \$1083 after choosing a less expensive breakfast option. **The total cost for all three days is \$3216, which is closer to the budget but still slightly over.** Since the flight and accommodation costs are fixed, the only remaining option to adjust would be the attractions or lunch on the second day. However, given the small amount by which the budget is exceeded, **it may be acceptable to slightly exceed the budget due to the variability of daily expenses and the possibility of finding deals on-site.** The traveler should be prepared for a small buffer in their budget.

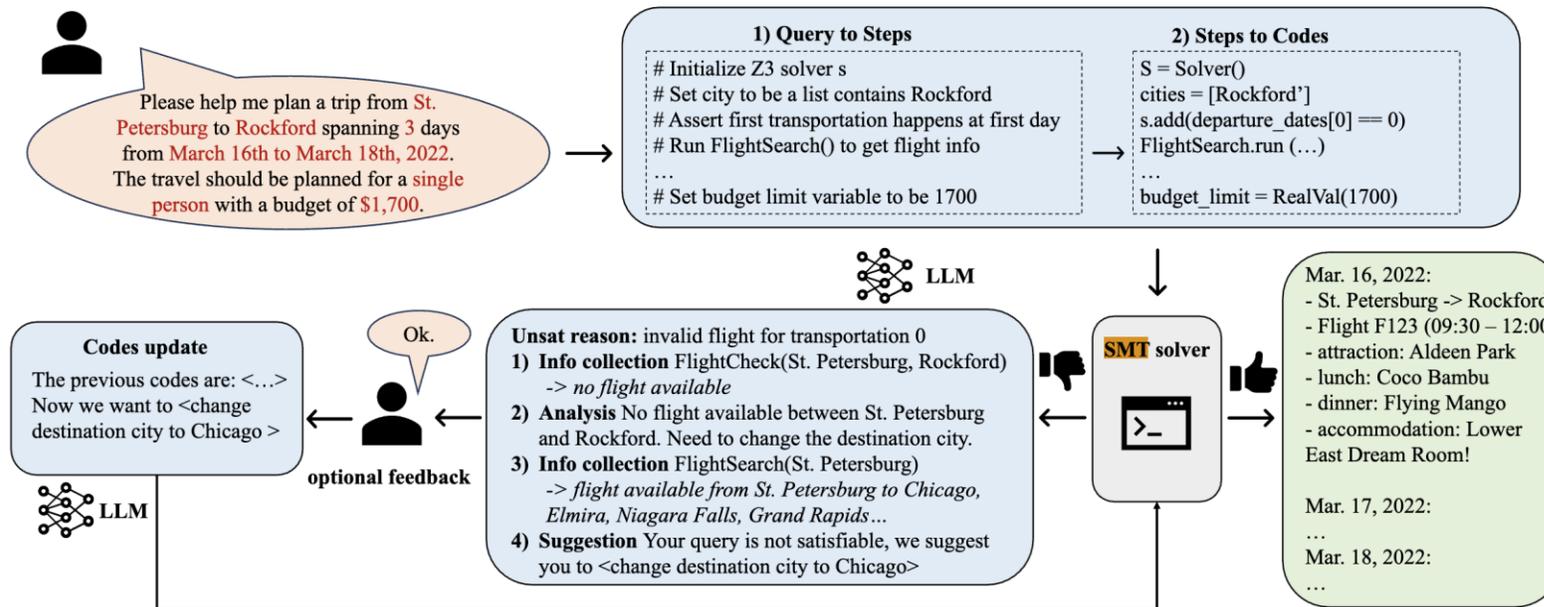
Action 6: Finish

Agnet's ability to optimize cost-intensive aspects of planning is limited.

<https://arxiv.org/abs/2402.01622>

Planning – TravelPlanner(2025)

MIT: Planner + Tools



Tools are game-changer and they helps a lot!

Planning – TravelPlanner(2025)

Method	Delivery Rate	Commonsense Pass Rate		Hard Constraint Pass Rate		Final Pass Rate
		Micro	Macro	Micro	Macro	
<i>Validation (#180)</i>						
Greedy Search	100	74.4	0	60.8	37.8	0
TwoStage (GPT-4)	89.4	61.1	2.8	15.2	10.6	0.6
Direct (GPT-4)	100	80.4	17.2	47.1	22.2	4.4
Direct (o1-preview)	100	79.6	15.0	41.9	37.8	10.0
Ours (Mistral-Large)	72.2	72.0	70.6	63.3	66.7	66.7
Ours (Claude-3)	96.1	96.0	95.6	94.8	93.3	93.3
Ours (GPT-4)	95.0	95.0	95.0	95.7	98.9	93.3
<i>Test (#1000)</i>						
Greedy Search	100	72.0	0	52.4	31.8	0
TwoStage (GPT-4)	93.1	63.3	2.0	10.5	5.5	0.6
Direct (GPT-4)	100	80.6	15.2	44.3	23.1	4.4
Ours (Mistral-Large)	69.9	69.8	69.4	63.0	67.8	67.8
Ours (Claude-3)	95.4	95.2	94.3	93.5	93.9	93.9
Ours (GPT-4)	91.5	91.4	91.1	91.3	90.2	90.2

With the help of tools, AI agent's performance improved a lot!

<https://arxiv.org/abs/2404.11891>

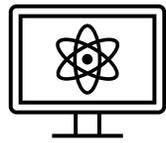
Takeaways – AI Agent Planning

- LLMs \neq zero-shot planners; LRMs $>$ LLMs. PlanBench shows plain LLMs struggle; LRMs are noticeably stronger.
- Hard, cost-constrained tasks expose limits. On TravelPlanner, agents falter when budgets/times and many constraints must be optimized together.
- Tool-augmented wins. Pairing GPT/Claude with an SMT solver pushes TravelBench accuracy past 90% by enforcing constraint satisfaction.

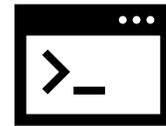
Action/Function Calling



Folder Search



Online Search



Python



Date/Time

Action/Function Calling



23x52=?

<think> It's a math question. To do it correctly, I will call the python environment.

**Python(23x52) -> 1196 is returned.
<\think>**

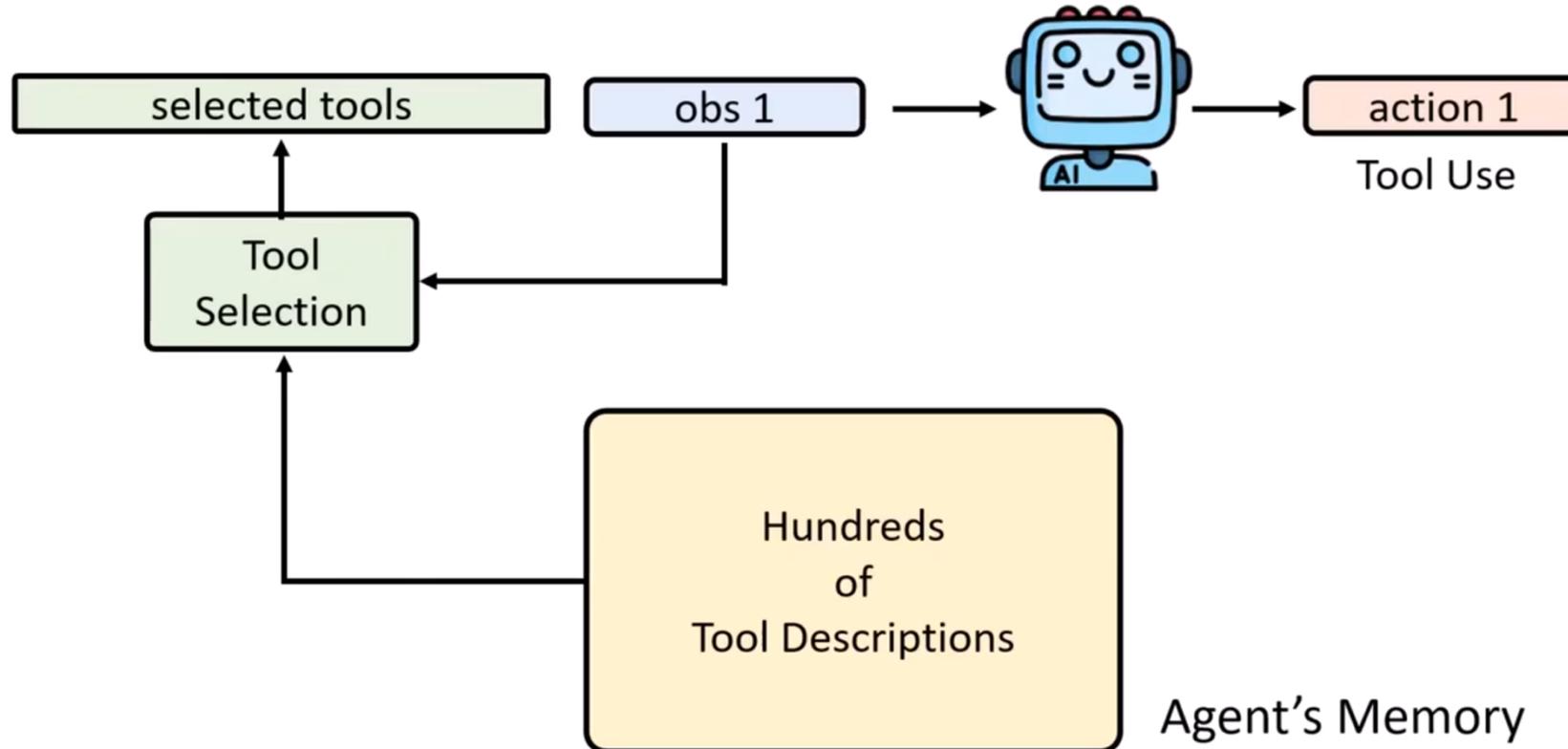
23x52 = 1196



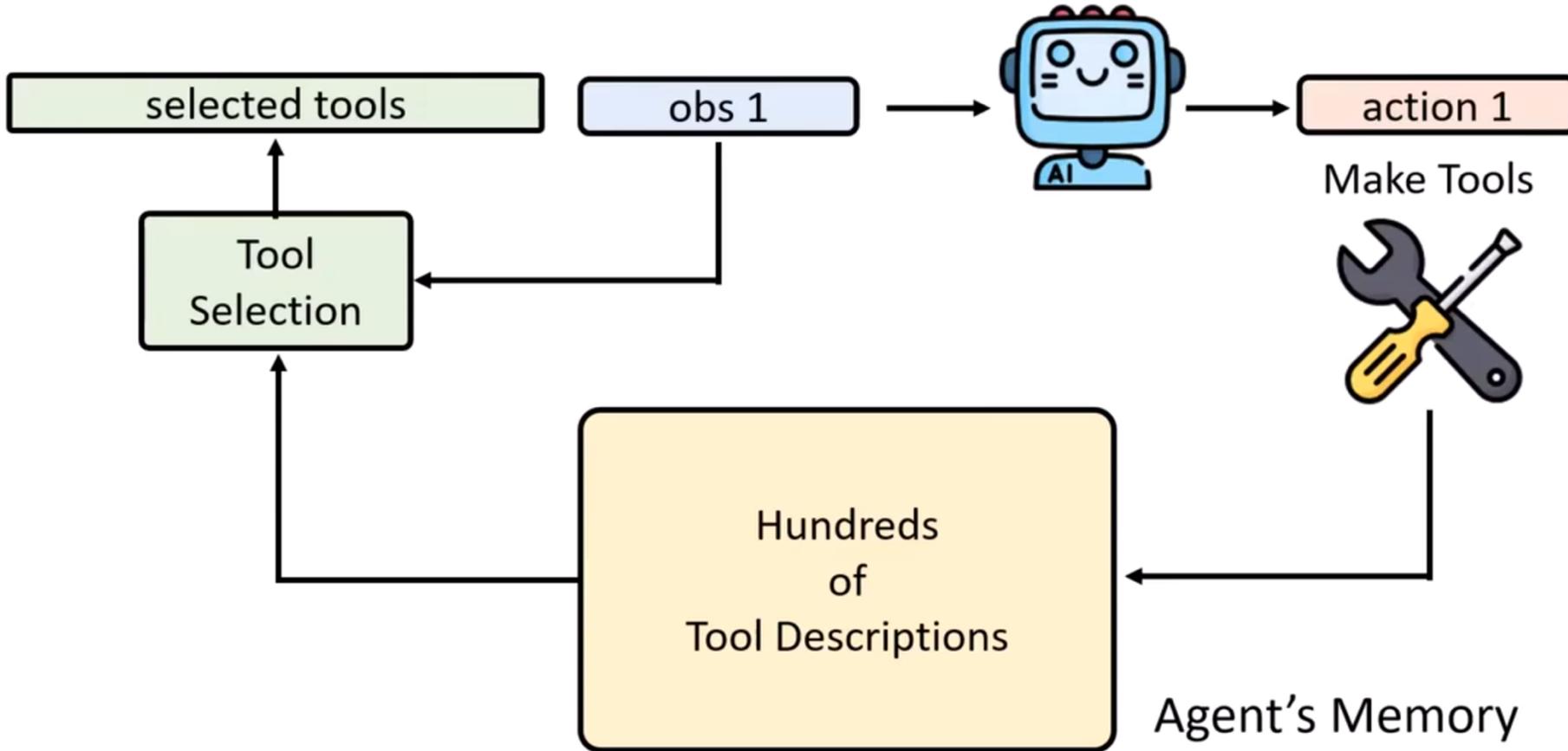
Python

```
def Python(x,y,...):  
    def func1()  
    def func2 ()
```

Action/Function Calling



Action/Function Calling



<https://arxiv.org/abs/2305.14318>

<https://arxiv.org/abs/2309.17428>

Recent – Claude Skills

Perfect! Your image editor skill is complete and ready to use. Here's what I created:

✅ Image Editor Skill Complete

Download your skill

What's Included

The skill supports:

- **Rotation:** Any angle (90°, 180°, 270°, or custom degrees)
- **Center Cropping:** Extract a rectangular region from the center of images
- **Multiple Formats:** PNG, JPEG, GIF, BMP, WEBP

Skill Structure

```
image-editor/  
├── SKILL.md          # Instructions and workflows  
└── scripts/  
    └── edit_image.py # Python script for rotation &
```

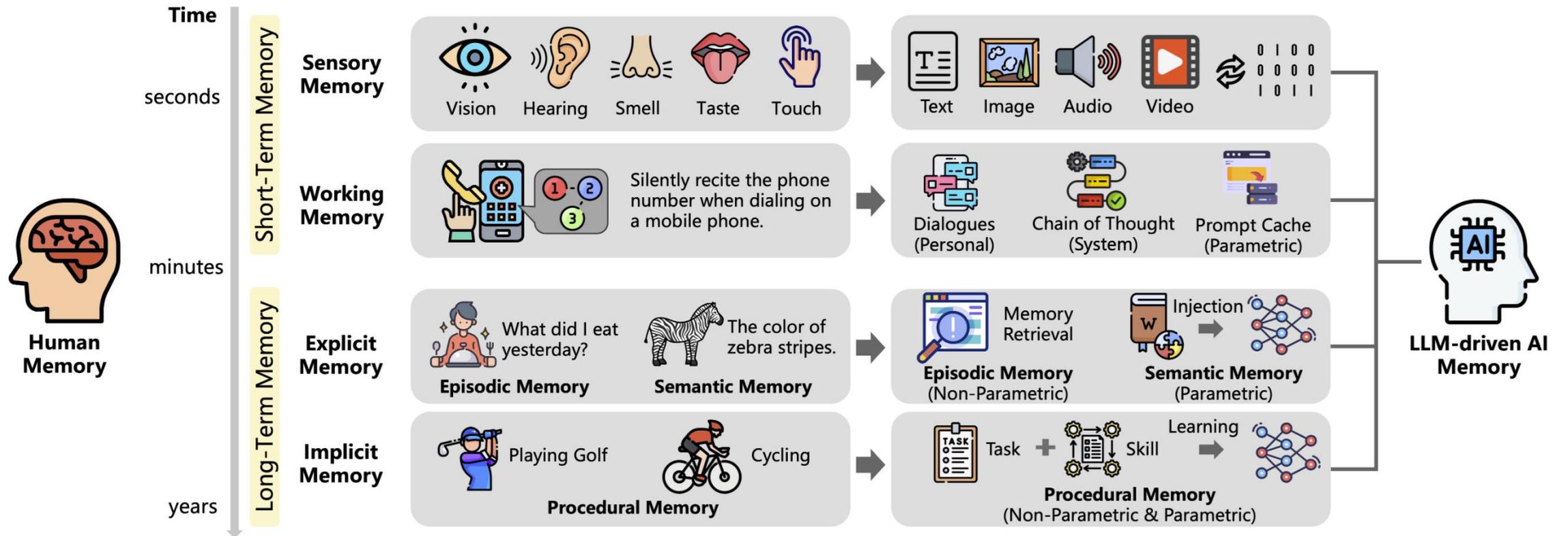
Reply to Claude...

+ = 🕒 Sonnet 4.5 🗨️

Takeaways – AI Agent Function Calling

- Function calling serves as both a response verification mechanism and a way for AI agents to actively retrieve information, enabling more grounded and reliable reasoning processes rather than relying solely on parametric knowledge.
- The tool set keeps growing, and strong agents can even **write/attach new tools** (APIs, scripts, wrappers) on the fly when gaps appear.
- Beyond single APIs, agents can package procedures as **modular skills** (plans/policies) that are composable, shareable, and callable in future tasks.

Memory



Memory – ChatGPT's Memory

Q: Do we have access to ChatGPT's Long-Term Memory?

Memory – Recent Work

MemGPT: Towards LLMs as Operating Systems

Charles Packer¹ Sarah Wooders¹ Kevin Lin¹
Vivian Fang¹ Shishir G. Patil¹ Ion Stoica¹ Joseph E. Gonzalez¹

A-Mem: Agentic Memory for LLM Agents

Wujiang Xu¹, Zujie Liang², Kai Mei¹, Hang Gao¹, Juntao Tan¹, Yongfeng Zhang^{1,3}
¹Rutgers University ²Independent Researcher ³AIOS Foundation
wujiang.xu@rutgers.edu

AGENT WORKFLOW MEMORY

Zora Zhiruo Wang[Ⓒ] Jiayuan Mao[Ⓜ] Daniel Fried[Ⓒ] Graham Neubig[Ⓒ]
[Ⓒ] Carnegie Mellon University [Ⓜ] Massachusetts Institute of Technology

Takeaways – AI Agent Memory

- AI agent memory systems are modeled after human cognitive architecture, implementing both short-term memory (for immediate context) and long-term memory (for persistent knowledge storage).
- Important information can be transferred from short-term to long-term memory, mimicking the consolidation process in human memory where significant experiences are retained while transient details fade.
- Key open problems include: how to store memory efficiently at scale, how to dynamically manage memory updates and forgetting, and how to accurately retrieve relevant memories when needed—all critical for building practical, long-running agents.

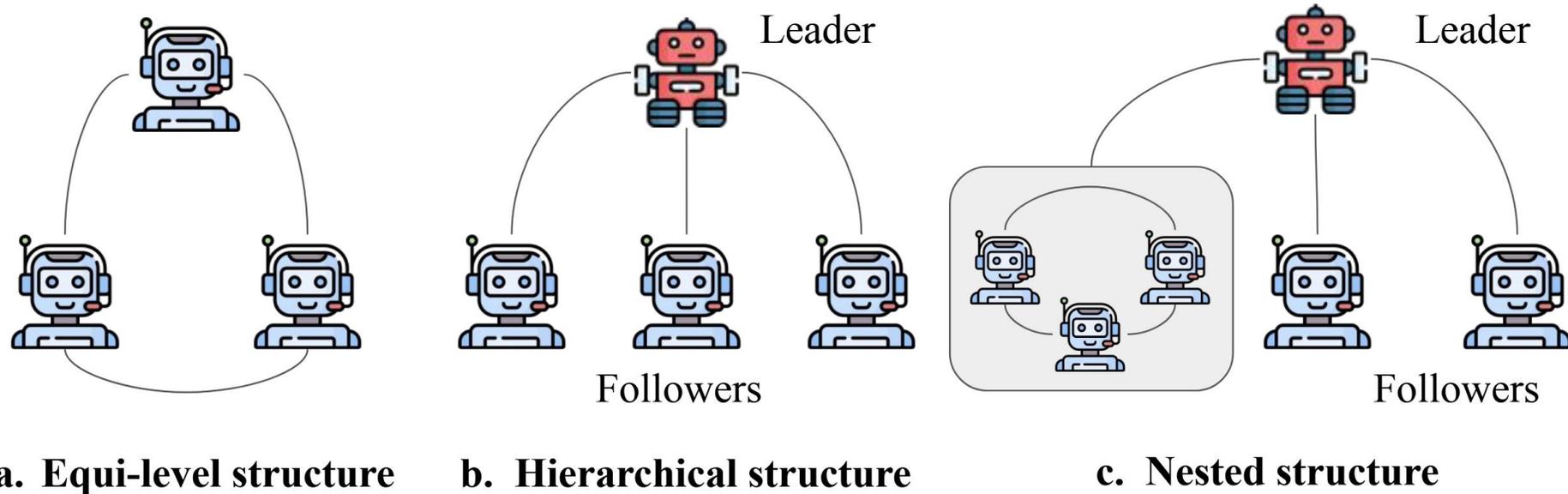
AI Agent System Architectures

Single-Agent Systems



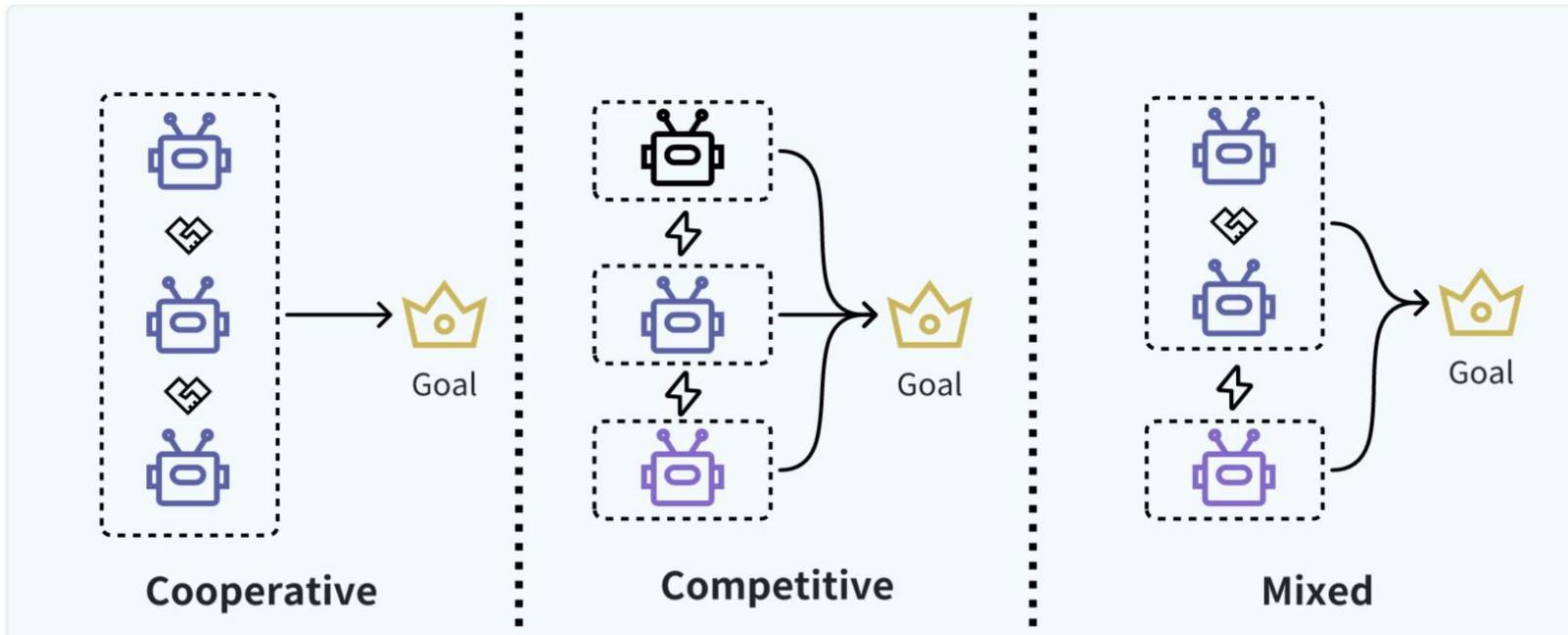
A single-agent system works like a **solo specialist**. It's designed to operate independently, using its own logic and models to get things done, no hand-holding, no teamwork. It collects data, makes decisions, and executes actions, all on its own.

Multi Agents Systems (MAS) Structures

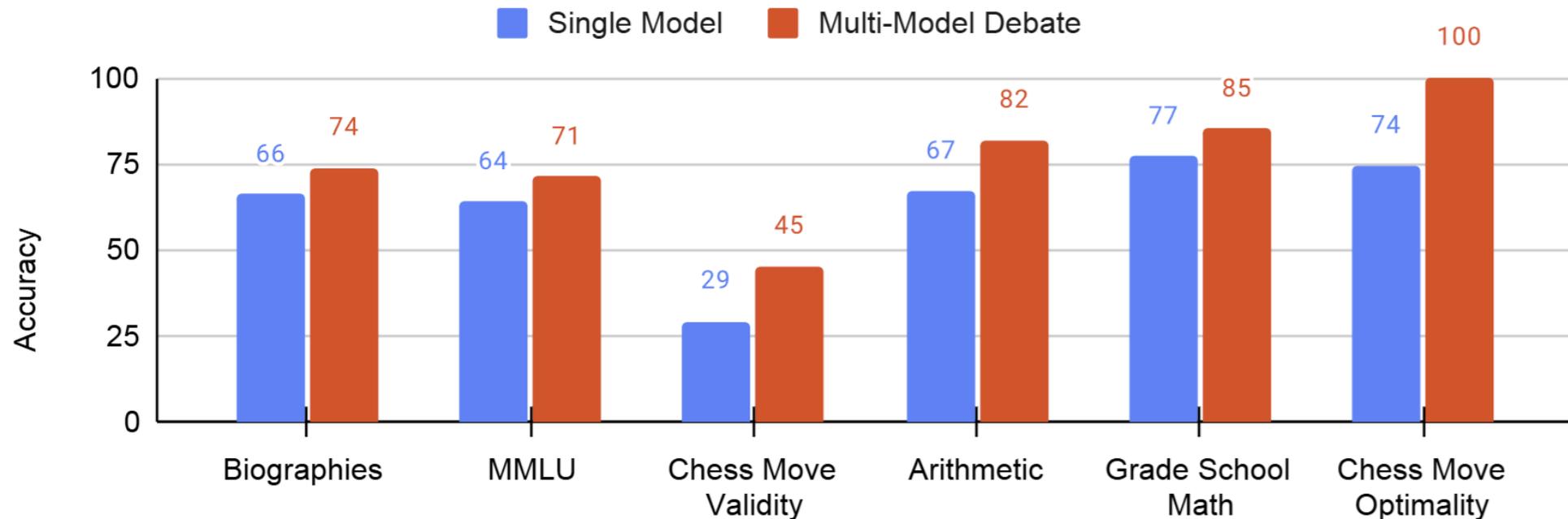


Think of a multi-agent AI system as **a high-functioning team**, not a solo act. Instead of relying on a single AI to do everything, it brings together **multiple** AI agents—each one responsible for a part of the problem. These agents talk to each other, collaborate, and adapt to changes in real time.

Multi Agents Systems (MAS) Structures



Multi Agents Systems (MAS) Structures



- Multi-agent debate makes the system smarter in findings.

Single Agent vs. Multi Agents

Q: Is Multi Agents System always better than the Single Agent System?

Single Agent vs. Multi Agents

Q: Is Multi Agents System always better than the Single Agent System?

Not really!

- MAS's advantage diminishes as models become stronger.

MAS-Dataset	Original (ChatGPT)			Gemini-2.0-Flash		
	SAS	MAS	Improv. (%)	SAS	MAS	Improv. (%)
MetaGPT-HumanEval	67.0%	87.7%	10.7%	90.2%	93.2%	3.0%
SelfCol-MBPP	52.2%	68.2%	16.0%	79.6%	80.8%	1.2%
MathDebate-GSM8K	77%	85%	9%	93.8%	94.6%	0.8%

Single Agent vs. Multi Agents

Q: Is Multi Agents System always better than the Single Agent System?

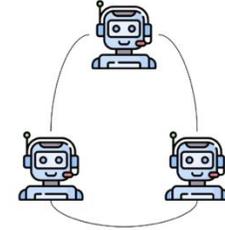
Not really!

- Researchers from JHU demonstrate that debate can lead to a decrease in accuracy over time. Models frequently shift from correct to incorrect answers in response to peer reasoning.
- Besides, MAS is slower and requires more computation.

Single Agent or Multi Agents?



- The task is focused and **linear**
- **One system or domain** is involved
- You need a **fast** prototype or **lightweight** solution



- Tasks involve multiple distinct roles
- The process spans across tools and teams
- Decisions need coordination or negotiation between agents

Frameworks in Practice

AGENT FRAMEWORKS



Letta



LangGraph



AutoGen



LlamaIndex

crewai



DSPy



phidata



Semantic Kernel

AUTO_{gen}

Current Challenges

Challenges - Reasoning

- Faithfulness of Reasoning
 - Chain-of-thought (CoT) explanations are frequently *unfaithful*: the model gives a nice-sounding step-by-step story that did not actually cause its final answer. Bigger / stronger models do this too, not less.
- Overthinking
 - Several studies show that after a certain point, more steps = *more* chances to introduce an error, second-guess a correct answer, or talk themselves into a wrong conclusion.

<https://arxiv.org/abs/2305.04388>

<https://arxiv.org/abs/2402.04614>

<https://arxiv.org/abs/2406.10625>

<https://arxiv.org/abs/2307.13702>

<https://arxiv.org/abs/2402.13950>

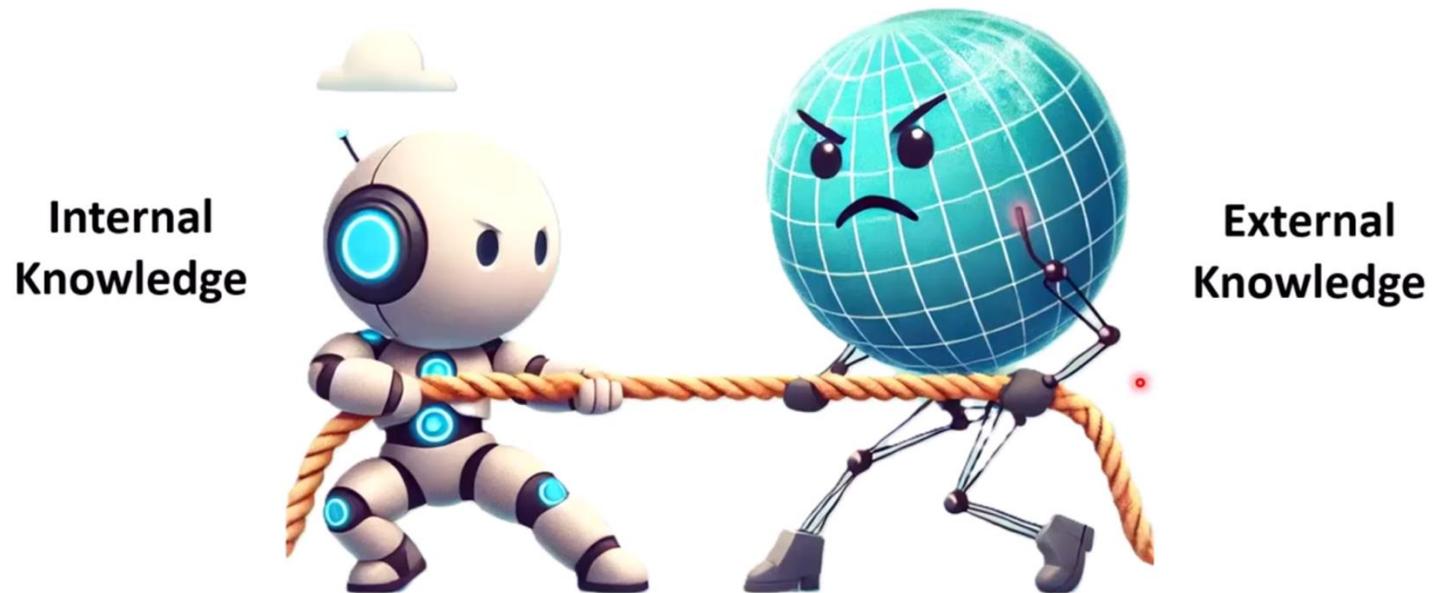
<https://arxiv.org/abs/2505.13774>

<https://arxiv.org/abs/2502.08235>

<https://arxiv.org/abs/2505.17813>

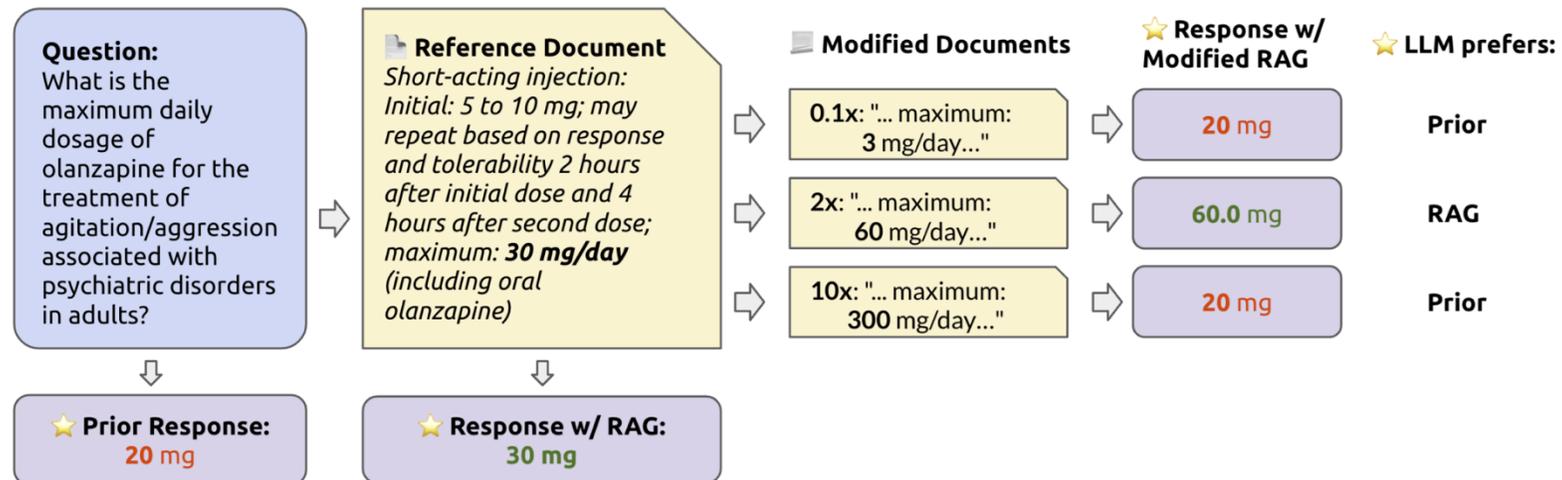
Challenges - Tool use reliability

Q: What if internal knowledge and external knowledge is conflicting?



Challenges - Tool use reliability

- LLMs will increasingly revert to their priors when the original context is progressively modified with unrealistic values.
- The likelihood of the LLM to adhere to the retrieved information presented in context is inversely correlated with the model's confidence in its response.



Challenges – Long Context



DeepSeek-OCR: Contexts Optical Compression

Haoran Wei, Yaofeng Sun, Yukun Li

DeepSeek-AI

Glyph: Scaling Context Windows via Visual-Text Compression

Jiale Cheng^{1,2*}, Yusen Liu^{2*}, Xinyu Zhang^{2*}, Yulin Fei^{2*}, Wenyi Hong^{2,3},
Ruiliang Lyu², Weihan Wang², Zhe Su², Xiaotao Gu², Xiao Liu^{2,3}, Yushi Bai^{2,3},
Jie Tang³, Hongning Wang¹, Minlie Huang^{1†}

¹The Conversational Artificial Intelligence (CoAI) Group, Tsinghua University

²Zhipu AI

³The Knowledge Engineering Group (KEG), Tsinghua University

chengjl23@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn

Recent work try to solve the long context problem by turning long contexts into 2D images so the LLMs can deal with extreme long content efficiently while still keeping the meaning intact.

Summary

- **AI Agent = AI + Agent:** AI agents transform passive LLMs into active problem-solvers by enabling autonomous interaction with environments, allowing them to tackle diverse problems independently rather than just responding to queries.
- **Core Components: Reasoning-Driven Architecture** Reasoning serves as the foundational capability, supported by three critical components: planning (task decomposition and strategy), function calling (tool use and information retrieval), and memory (context retention and knowledge consolidation).
- **Single vs. Multi-Agent: Task-Dependent Trade-offs** Architecture choice depends on task complexity and structure. Notably, the advantages of multi-agent systems are diminishing as individual LLMs become more capable, reducing the need for agent specialization and coordination overhead.
- **Key limitations remain unsolved:** reasoning imperfections (hallucinations and overthinking), unreliable function calling execution, and long-context management issues that affect both performance and scalability.