CS 639: Foundation Models
Agents I

Fred Sala

University of Wisconsin-Madison

April 21, 2026

Announcements

- **Homework 4**: due
- Project---ongoing!
- **Next 2 classes (4/23 + 4/28): will happen offline; I will release lecture recordings**
  - I will be in Brazil for ICLR
- Class outline

Tuesday April 21	Agents I
Thursday April 23	Agents II
Tuesday April 28	Applications
Thursday April 30	Future Areas

End of Class

Outline

- **Introduction to LLM-Powered Agents**
  - Motivation, goals, differences vs classical agents / standard LMs, overall architectures and key components
- **Multiagent Systems**
  - Motivation, simulations, architectures and design, communication
- **Challenges**
  - Scaling, realism, open problems

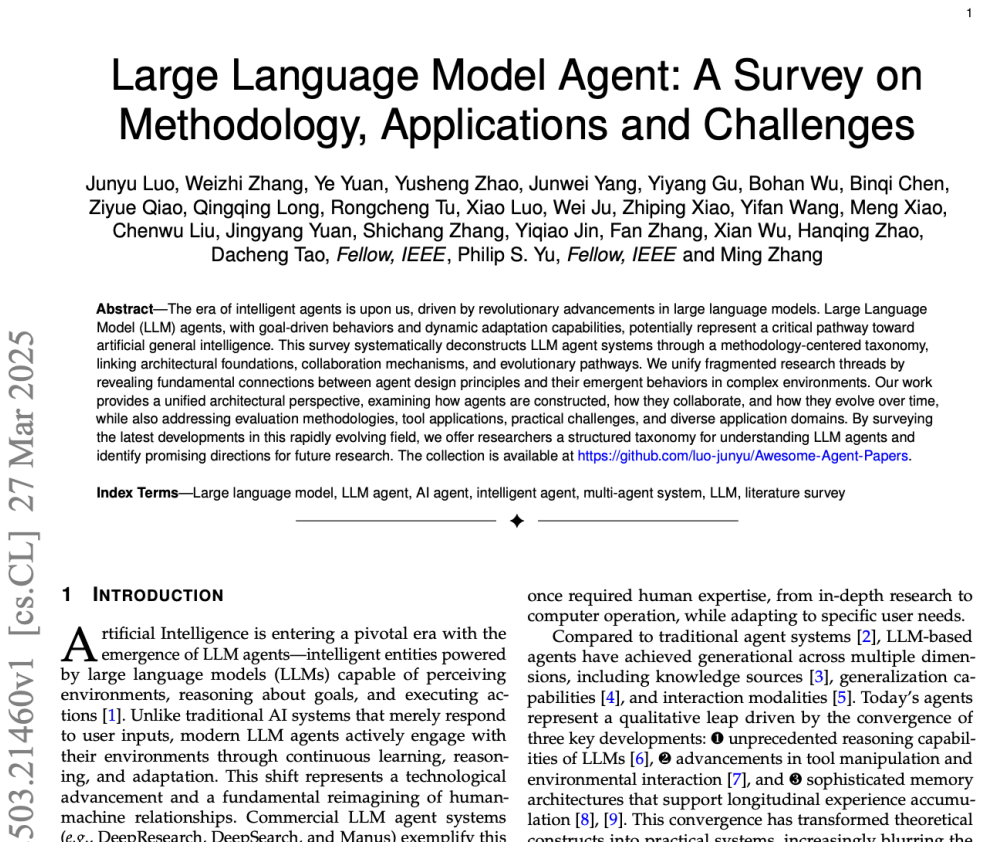
Outline

- **Introduction to LLM-Powered Agents**
  - Motivation, goals, differences vs classical agents / standard LMs, overall architectures and key components
- **Multiagent Systems**
  - Motivation, simulations, architectures and design, communication
- **Challenges**
  - Scaling, realism, open problems

Motivation: **From LLMs To Agents**

Standard LLMs are static

- But we want **interactive** systems
  - Want systems that can reason, use tools, and act autonomously
  - Want FMs that can be actors, not just predictors
- Useful survey: Wang et al ‘25 “Large Language Model Agent: A Survey on Methodology"



Large Language Model Agent: A Survey on Methodology, Applications and Challenges

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xian Wu, Hanqing Zhao, Dacheng Tao, *Fellow, IEEE*, Philip S. Yu, *Fellow, IEEE* and Ming Zhang

Abstract—The era of intelligent agents is upon us, driven by revolutionary advancements in large language models. Large Language Model (LLM) agents, with goal-driven behaviors and dynamic adaptation capabilities, potentially represent a critical pathway toward artificial general intelligence. This survey systematically deconstructs LLM agent systems through a methodology-centered taxonomy, linking architectural foundations, collaboration mechanisms, and evolutionary pathways. We unify fragmented research threads by revealing fundamental connections between agent design principles and their emergent behaviors in complex environments. Our work provides a unified architectural perspective, examining how agents are constructed, how they collaborate, and how they evolve over time, while also addressing evaluation methodologies, tool applications, practical challenges, and diverse application domains. By surveying the latest developments in this rapidly evolving field, we offer researchers a structured taxonomy for understanding LLM agents and identify promising directions for future research. The collection is available at https://github.com/luo-junyu/Awesome-Agent-Papers.

Index Terms—Large language model, LLM agent, AI agent, intelligent agent, multi-agent system, LLM, literature survey

1 Introduction

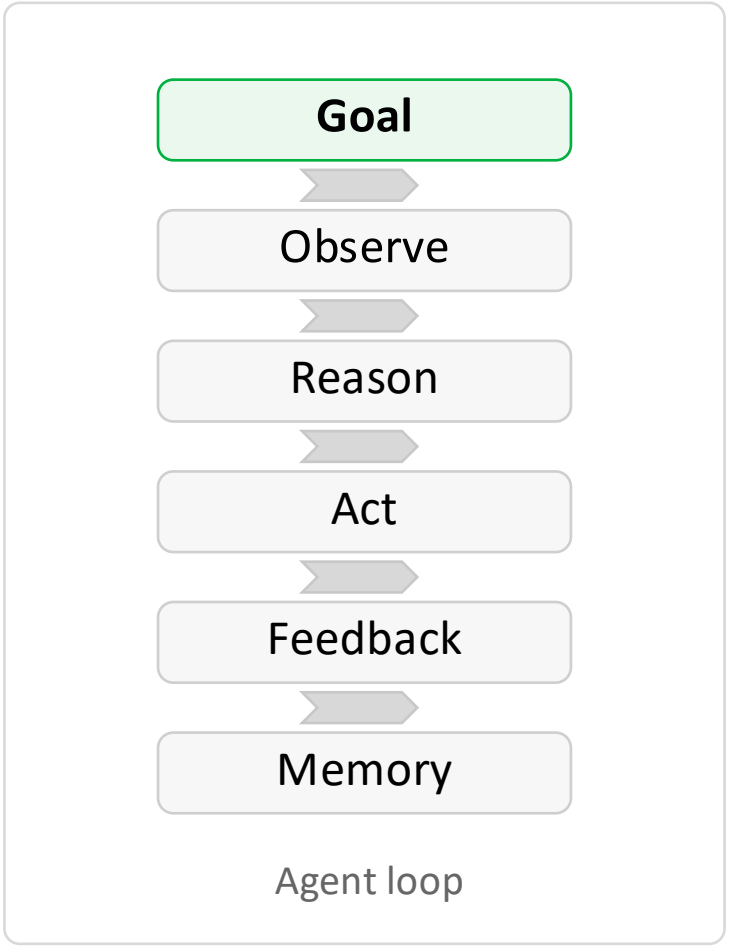
Artificial Intelligence is entering a pivotal era with the emergence of LLM agents—intelligent entities powered by large language models (LLMs) capable of perceiving environments, reasoning about goals, and executing actions [1]. Unlike traditional AI systems that merely respond to user inputs, modern LLM agents actively engage with their environments through continuous learning, reasoning, and adaptation. This shift represents a technological advancement and a fundamental reimagining of human-machine relationships. Commercial LLM agent systems (*e.g.*, DeepResearch, DeepSearch, and Manus) exemplify this once required human expertise, from in-depth research to computer operation, while adapting to specific user needs.

Compared to traditional agent systems [2], LLM-based agents have achieved generational across multiple dimensions, including knowledge sources [3], generalization capabilities [4], and interaction modalities [5]. Today’s agents represent a qualitative leap driven by the convergence of three key developments: ❶ unprecedented reasoning capabilities of LLMs [6], ❷ advancements in tool manipulation and environmental interaction [7], and ❸ sophisticated memory architectures that support longitudinal experience accumulation [8], [9]. This convergence has transformed theoretical constructs into practical systems, increasingly blurring the

What's an Agent?

Let's **differentiate** from a standard LLM

- These usually answer a prompt (single response)
- Agents act, often taking many steps, to accomplish a **goal**
  - Interact with tools, environments, feedback loops
- But, still **built on top of LLMs**
- What do we use them for?
  - As we saw in benchmarks, coding, computer use, research, science, etc.



More Concrete Definition

An LLM agent is a system that uses a FM to **reason**, **plan,** and **act** in an environment

- Often with tools and memory.
- Note: the agent includes the **harness/scaffold**, not just the FM or LLM that powers it

Pre-LLMs: **Classical Agents**

Much of the work on agents predates emergence of FMs and LLMs

- A common framework: **PEAS** (Performance, Environment, Actuators, Sensors
- Agent types:
  - Reactive, deliberative, hybrid

- **Reinforcement learning** is a standard paradigm for agents
    - Convenient since we already use it for LLMs

Architecture of LLM-Powered Agents

High-level components: **input**, **memory**, **planner**, **tools**,

- **Input**: whatever we have access to
- **Memory**: as simple as context window, but
- **Planning**: reasoning-based decomposition etc
- **Tools**: similar to our earlier discussion on integrating tool use

Single-Agent vs. **Multi-Agent**

We can create multiple agents

- These get to interact with each other to
- Note that we can simulate this with a single agent too!
- Key difference: multiple agents permits using **different** harnesses and models
  - This specialization can be more efficient: use cheaper models for easier tasks, etc.
  - But…more complexity!

Single Agent **Patterns**

Variations based on where planning, execution, and verification are placed. Some examples:

- **ReAct**
  - Interleave reason and execution
- **Plan/execute**
  - Planning phase followed by execution
- **Reflection**
  - Generate, critique, revise
- **Verifier-based**

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao*[,1], Jeffrey Zhao[2], Dian Yu[2], Nan Du[2], Izhak Shafran[2], Karthik Narasimhan[1], Yuan Cao[2]

[1]Department of Computer Science, Princeton University
[2]Google Research, Brain team
[1]{shunyuy,karthikn}@princeton.edu
[2]{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

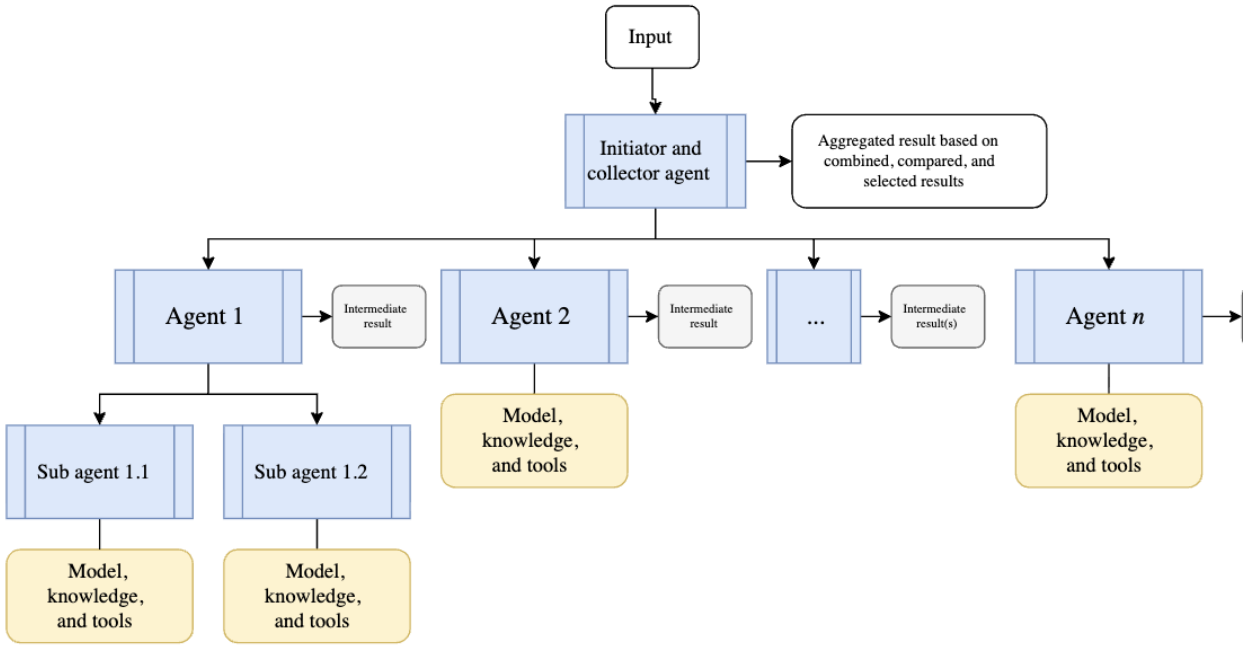
ABSTRACT

While large language models (LLMs) have demonstrated impressive performance across tasks in language understanding and interactive decision making, their abilities for reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. In this paper, we explore the use of LLMs to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments. We apply our approach, named ReAct, to a diverse set of language and decision making tasks and demonstrate its effectiveness over state-of-the-art baselines in addition to improved human interpretability and trustworthiness. Concretely, on question answering (HotpotQA) and fact verification (Fever), ReAct overcomes prevalent issues of hallucination and error propagation in chain-of-thought reasoning by interacting with a simple Wikipedia API, and generating human-like task-solving trajectories that are more interpretable than baselines without reasoning traces. Furthermore, on two interactive decision making benchmarks (ALFWorld and WebShop), ReAct outperforms imitation and reinforcement learning methods by an absolute success rate of 34% and 10% respectively, while being prompted with only one or two in-context examples.

Multi-Agent Patterns

Much more flexible

- Large design space!
- Most common approach: central **orchestrator** splits up work and delegates to **specialists**

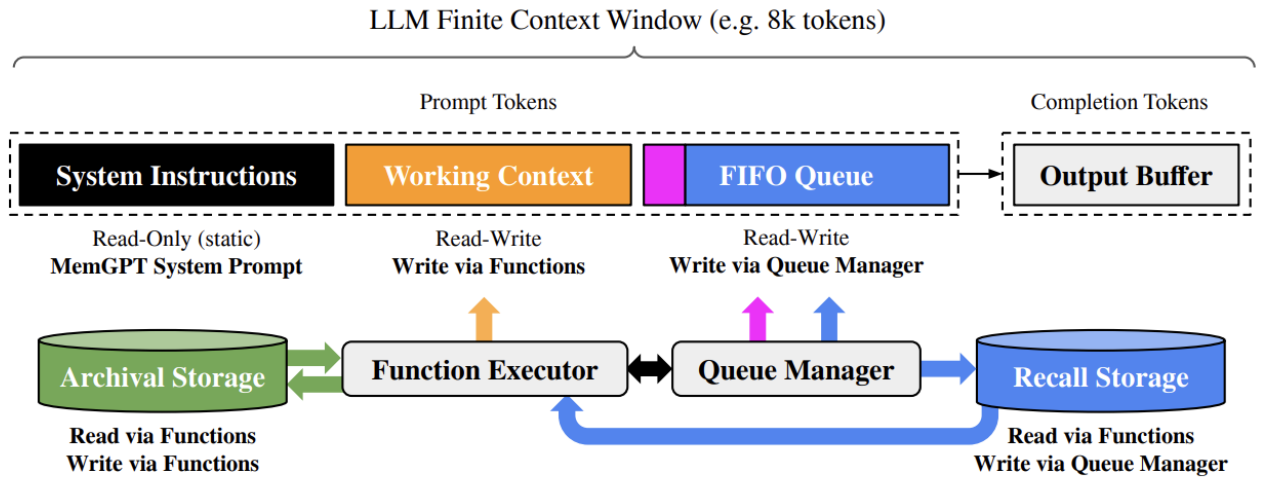


Microsoft

Components: Memories

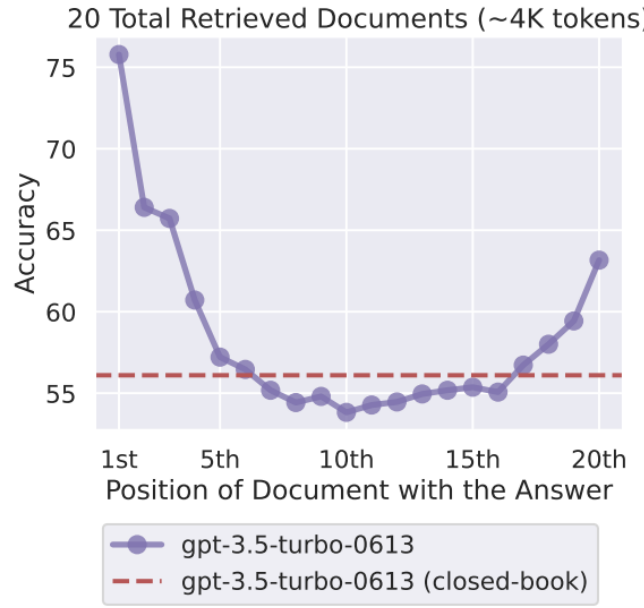
Interactive agents require the use of **memory**

- I.e., we need to store state/other useful information
- Simplest approach: **use LLM context window** as the memory
  - Increasingly tractable with very long-context windows
- Can also create memory systems/agents
  - E.g., MemGPT (Packer et al '24)
- Or buffers, RAG, etc.



Interlude: Long Context Windows

- Simplest approach: **use LLM context window** as the memory
  - Increasingly tractable with very long-context windows
- Note that the advertised long context may not be realistic!
- Liu et al ‘23 (“Lost in the Middle: How Language Models Use Long Contexts”)
- Needle-in a haystack benchmarks
  - Numerous variants



Components: Reasoning and Planning

Classic AI problem (decompose task, plan)

- Addressable via reasoning capabilities of LLMs,
  - Including older methods: chain-of-thought, tree-of-thoughts, self-reflection
  - Example: **ReAct** (Reason + Act) (Yao et al '23)

(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

(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) ', …]

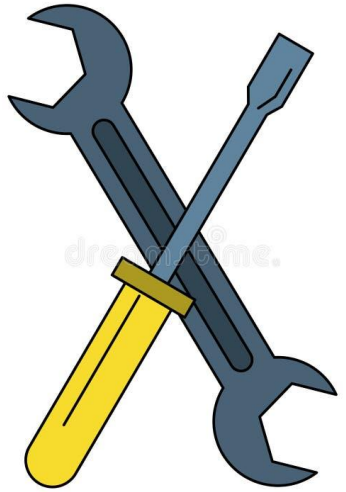
Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

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]

Components: Tool Use

Standard component in agentic settings

- Where do we get tools?
  - APIs, web search, etc
  - Or, in some cases use of physical tools
- Most agent frameworks permit the use of web/code-based tools
- Major challenge: tool usage is **noisy**

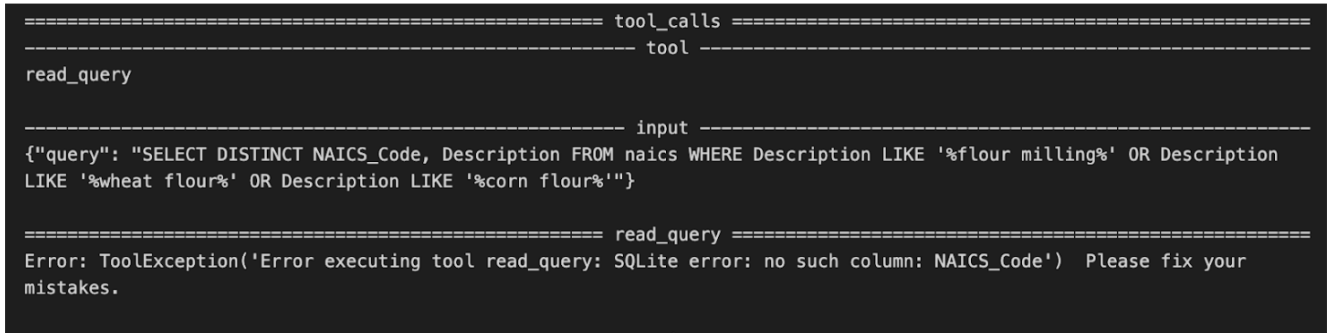


Tool Noise/Uncertainty

Lots of sources of error,

- Call the wrong tool, interpret output incorrectly, fail to chain together tools in the right way
- Ex: Agentic Insurance Underwriting Benchmark

  “**Tool use errors:** Across models, including top performers, agents made at least one tool call error in 36% of the conversations despite access to the metadata required to use tools properly.”

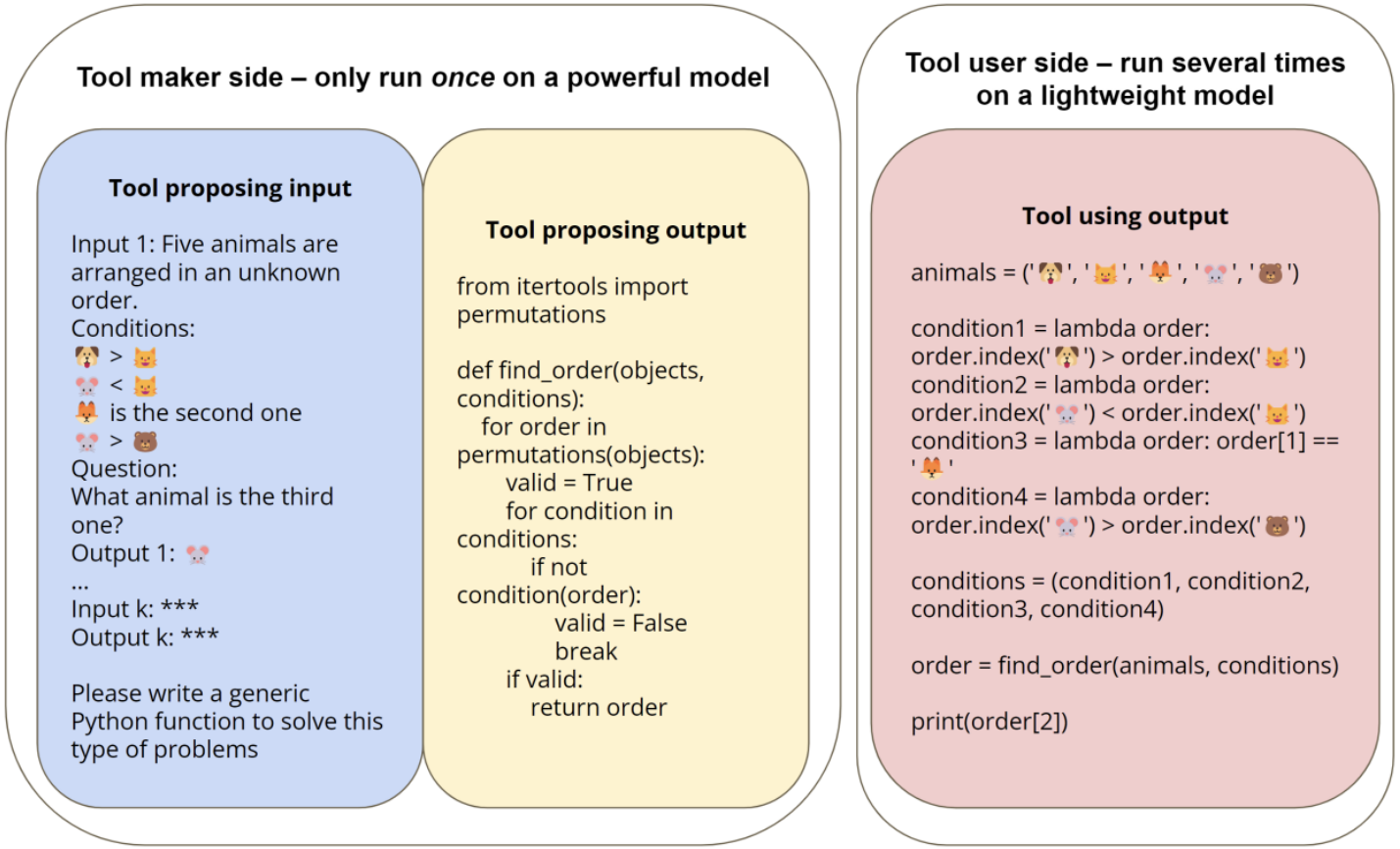


https://huggingface.co/datasets/snorkelai/Multi-Turn-Insurance-Underwriting

Creating New Tools

Note that LLMs can both use & **create** tools

- For example, LATM (Cai ‘24)
- Powerful model tool creator, less powerful is user



Break & Questions

Outline

- **Introduction to LLM-Powered Agents**
  - Motivation, goals, differences vs classical agents / standard LMs, overall architectures and key components
- **Multiagent Systems**
  - Motivation, simulations, architectures and design, communication
- **Challenges**
  - Scaling, realism, open problems

Multi-Agent Systems

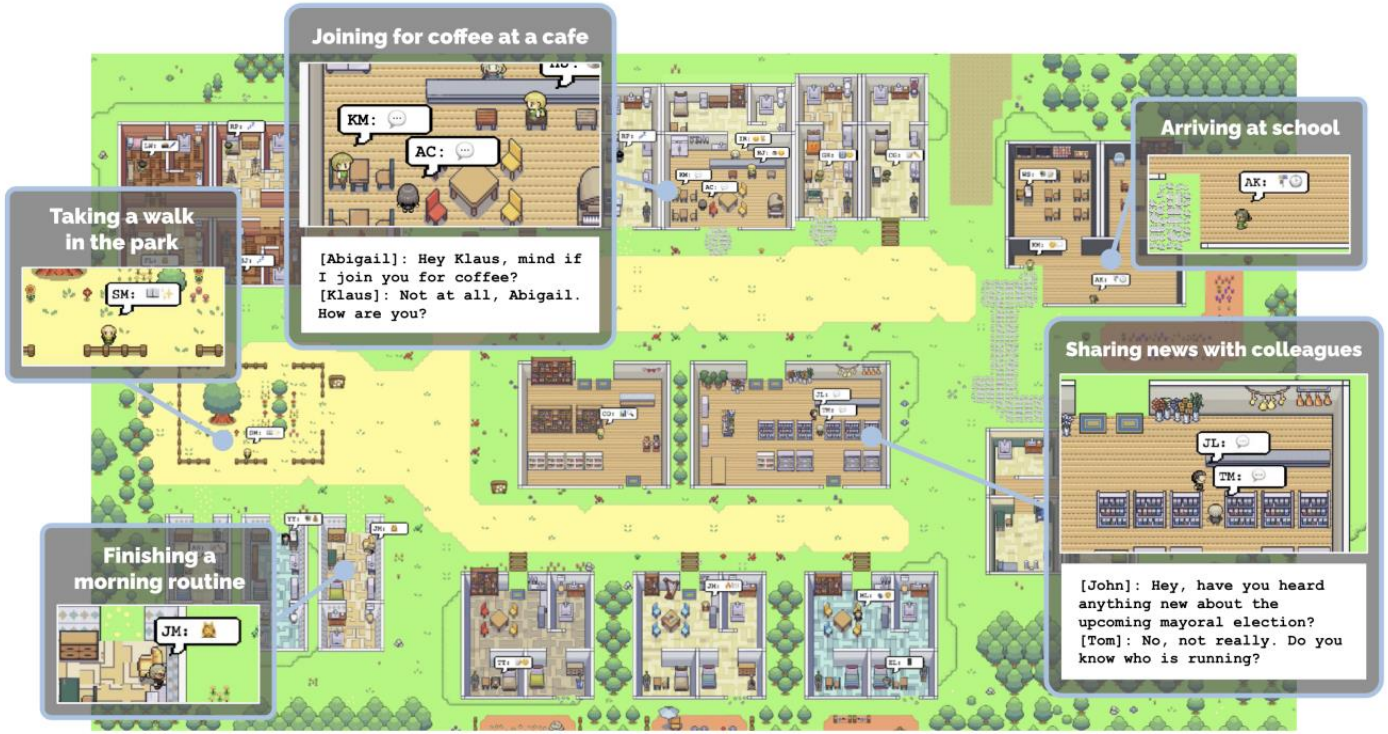
One advantage of agentic approach: can create multiple agents that cooperate to perform tasks. Why?

- **Scale** up workers, permit **specialization**
- Of course, inspired by earlier work,
  - Pre-LLM agent networks called “swarms”,
  - And distributed computing more broadly
- Also useful for simulations
  - Park et al ‘23

Interlude: **Societal Simulations**

Popular new area,

- Difficult to perform large-scale studies of human behavior,
- Try to set up a simulation of how humans interact via agents

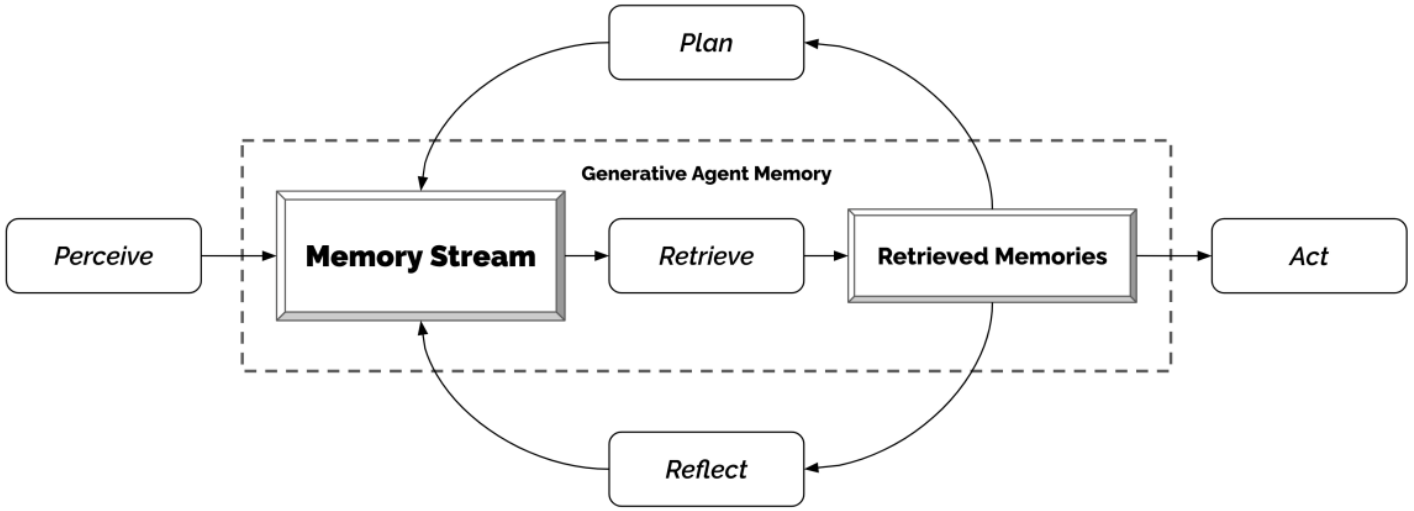


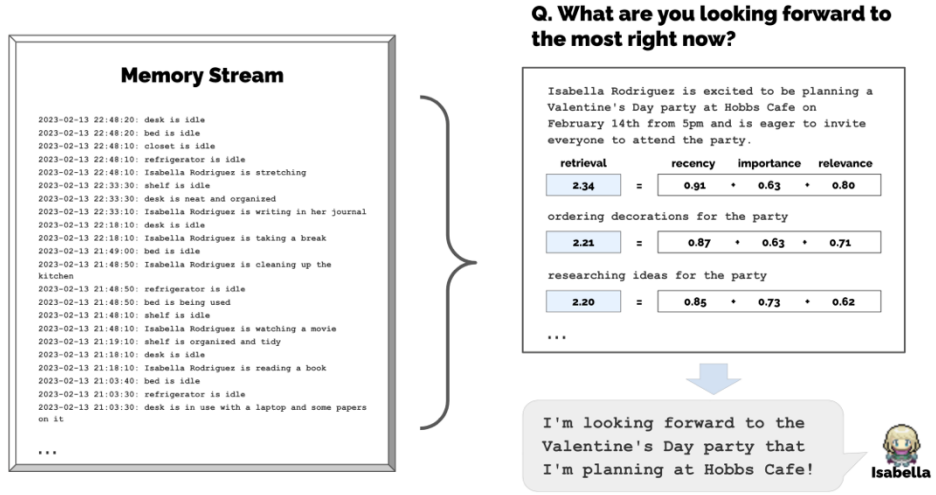
Park et al ‘23

Interlude: **Societal Simulations**

Popular new area,

- Try to set up a simulation of how humans interact via agents
- Itself a large multi-agent system: must tackle each of the design questions

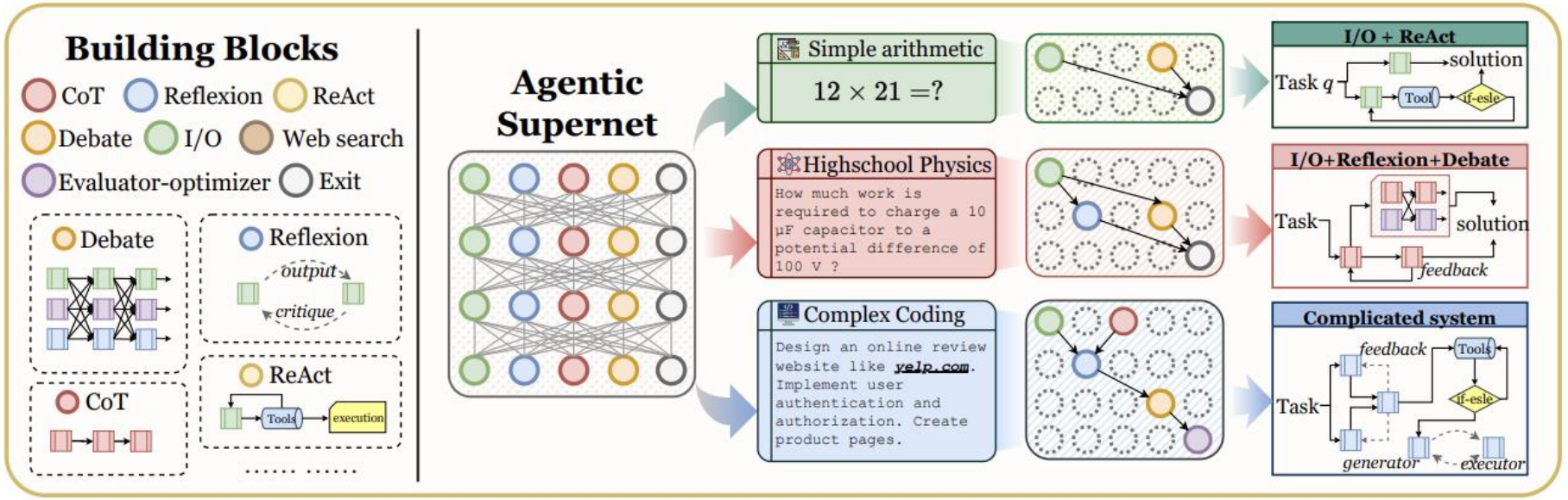




MAS Architecture Search

Lots of design choices to make!

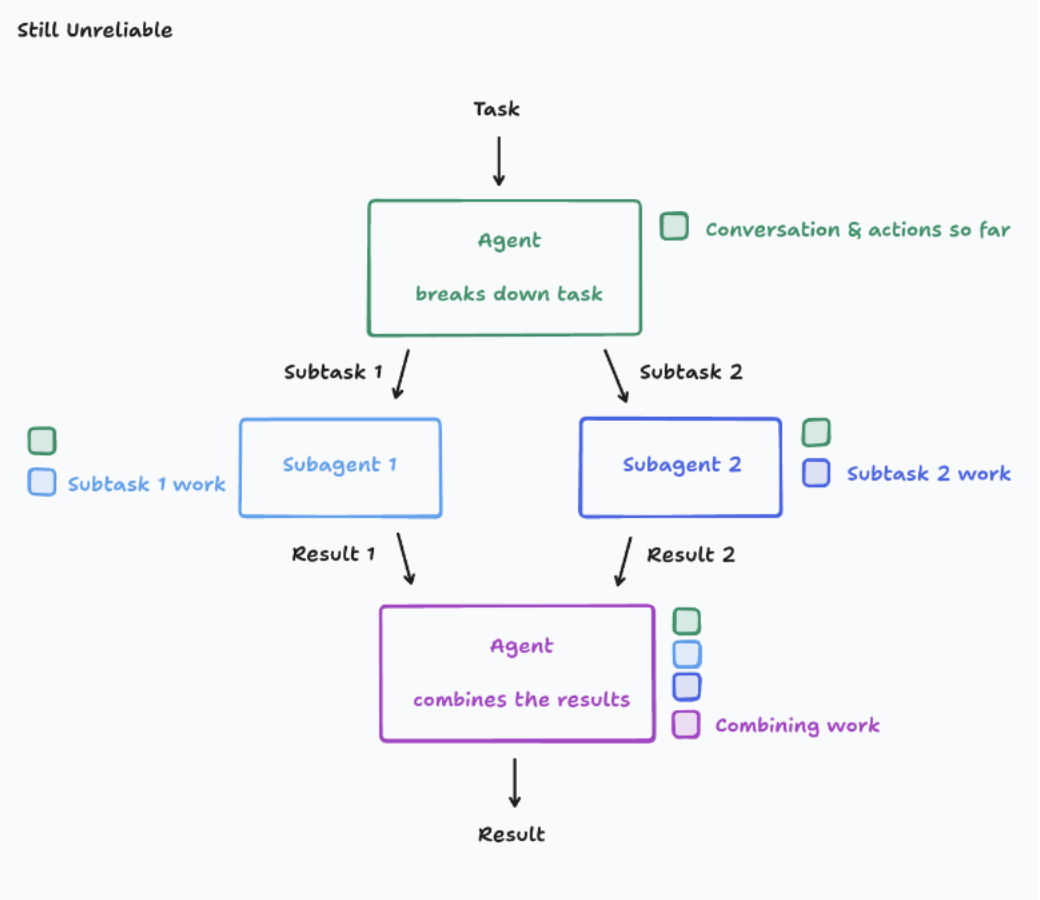
- Can borrow principles from **neural architecture search** to construct an overall agent architecture
- Example: Zhang et al ‘25, “Multi-agent Architecture Search via Agentic Supernet”



Multi-Agent Systems: Communication

Critical: each agent has access to information that is consistent with others,

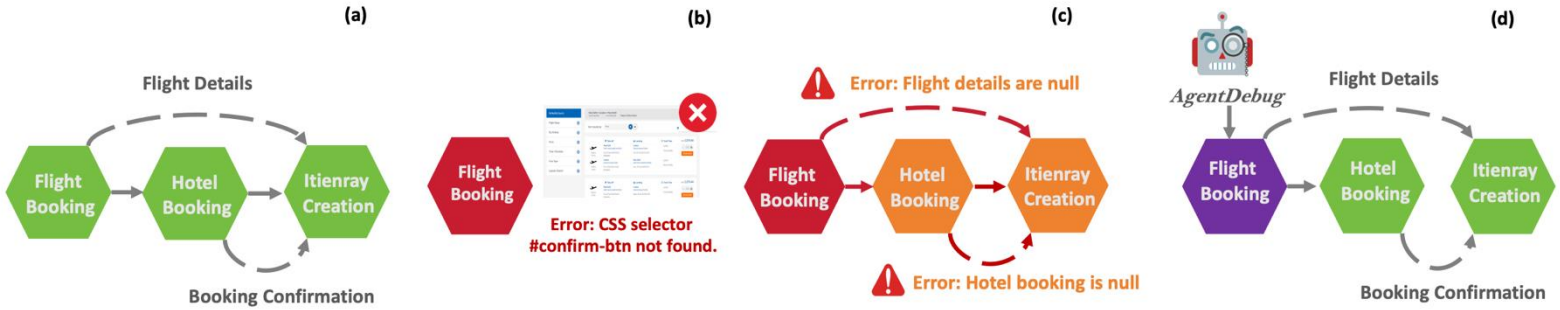
- But communicating everything between every pair of users is expensive.
- Nice blog on the challenges: “**Don’t Build Multi-Agents**” (Yan ’25 / Cognition)



Multi-Agent Systems: Failure Modes

More agents → more can go wrong!

- Agents **repeat each other's mistakes**
- **Noisy** communication (and expensive!)
- Subgoals become **misaligned** with the main goal
- Central orchestrator aggregates weak outputs **incorrectly**
- Verification **bottlenecks**
  - Easier to take actions than verify them...



Zhu et al '25

Break & Questions

Outline

- **Introduction to LLM-Powered Agents**
  - Motivation, goals, differences vs classical agents / standard LMs, overall architectures and key components
- **Multiagent Systems**
  - Motivation, simulations, architectures and design, communication
- **Challenges**
  - Scaling, realism, open problems

Agentic Challenges

Things get more complex than just static LLMs…

- Large design and configuration space!
- Lots of places where things can go wrong too

Failure Modes

Lots of possibilities! We saw some MAS-specific ones, but this is a more general problem with agents:

- **Hallucinated** tool calls
- **Memory drift** (summarization errors, staleness)
- **Brittleness** (small changes to UI or API change)
- **Misalignment** (optimize the wrong thing)
- **Cost and latency issues**

Scaling Agents?

Kim et al ‘26: *Towards a Science of Scaling Agent Systems*

Tests out many configurations

- Vary benchmarks, agent archs., LLMs

Sample findings:

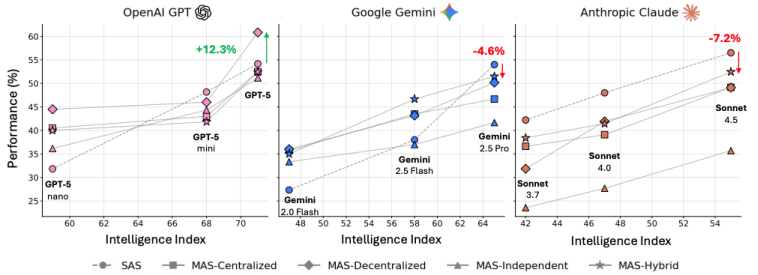
- **Benefits of MAS task dependent**
  - If not useful for tasks, burns tokens
  - Extra coordination can also hurt
- **Tool-heavy**: coordination harder
- **Central verification** useful

Google Research Google DeepMind

Towards a Science of Scaling Agent Systems

Yubin Kim[1,3,†], Ken Gu[1], Chanwoo Park[3], Chunjong Park[2], Samuel Schmidgall[2], A. Ali Heydari[1], Yao Yan[1], Zhihan Zhang[1], Yuchen Zhuang[2], Yun Liu[1], Mark Malhotra[1], Paul Pu Liang[3], Hae Won Park[3], Yuzhe Yang[1], Xuhai Xu[1], Yilun Du[1], Shwetak Patel[1], Tim Althoff[1], Daniel McDuff[1,†] and Xin Liu[1,†]
[1]Google Research, [2]Google DeepMind, [3]Massachusetts Institute of Technology, [†]Corresponding Author

Agents, **language model-based systems capable of reasoning, planning, and acting are widely adopted in real-world tasks, yet how their performance changes as these systems scale across key dimensions remains underexplored. We introduce quantitative *scaling principles* for agent systems as a predictive model, capturing how performance varies with coordination, model capability, and measurable system and task factors. Across 260 configurations spanning six agentic benchmarks, five canonical architectures (Single-Agent and four Multi-Agent: Independent, Centralized, Decentralized, Hybrid), and three LLM families, we perform controlled evaluations standardizing tools, prompts, and compute to isolate architectural effects. The resulting model achieves a cross-validated R^2=0.373 across all six benchmarks (R^2=0.413 with a task-grounded capability metric). We identify a robust capability-saturation effect and additional patterns: (1) a coordination yields diminishing returns once single-agent baselines exceed certain performance; (2) tool-heavy tasks appear to incur multi-agent overhead; and (3) architectures without centralized verification tend to propagate errors more than those with centralized coordination. Relative performance change compared to single-agent baseline ranges from +80.8% on decomposable financial reasoning to −70.0% on sequential planning, demonstrating that architecture-task alignment determines collaborative success. The framework identifies the best-performing architecture for 87% of held-out configurations and shows consistent relative architecture preferences on unseen frontier models. Agent effectiveness depends on alignment between coordination and task structure, and that mismatched coordination degrades the performance.**



Evaluating Memory

What about choice of memory?

- He et al ‘26 **MemoryArena**

- Explicitly tests performance in memory-bound tasks
- Benchmarks various choices:
  - Long-context buffer,
  - RAG system,
  - Memory agent.

Findings: memory-bound tasks **still hard**!

Benchmarking Agent Memory in Interdependent Multi-Session Agentic Tasks

Zexue He [*1] Yu Wang [*2] Churan Zhi [*2] Yuanzhe Hu [*2] Tzu-Ping Chen [*2] Lang Yin [*3] Ze Chen [4]
Tong Arthur Wu [5] Siru Ouyang [3] Zihan Wang [6] Jiaxin Pei [1] Julian McAuley [2] Yejin Choi [1] Alex Pentland [1]

Abstract

Existing evaluations of agents with memory typically assess *memorization* and *action* in isolation. One class of benchmarks evaluates memorization by testing recall of past conversations or text but fails to capture how memory is used to guide future decisions. Another class focuses on agent acting in single-session tasks without the need for long-term memory. However, in realistic settings, memorization and action are tightly coupled: agents acquire memory while interacting with the environment, and subsequently rely on that memory to solve future tasks. To capture this setting, We introduce MEMORYARENA, a unified evaluation gym for benchmarking agent memory in *multi-session Memory-Agent-Environment loops*. The benchmark consists of human-crafted agentic tasks with explicitly interdependent subtasks, where agents must learn from earlier actions and feedback by distilling experiences into memory, and subsequently use that memory to guide later actions to solve the overall task. MEMORYARENA supports evaluation across web navigation, preference-constrained planning, progressive information searching, and sequential formal reasoning, and reveals that agents with near-saturated performance on existing long-context memory benchmarks like LoCoMo perform poorly in our agentic setting, exposing a gap in current evaluations for agents with memory. MEMORYARENA is released at https://memoryarena.github.io/.

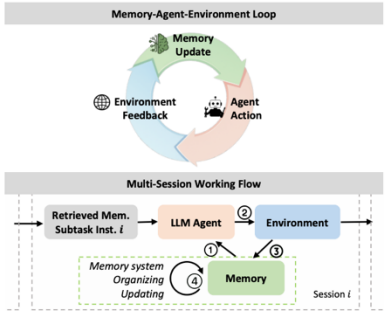


Figure 1. MEMORYARENA Evaluates agents with Memory with multi-session tasks in a Memory-Agent-Environment Loop.

through interaction with an environment (*action*) (Hu et al., 2025b). However, existing evaluations of LLM agents with memory typically isolate and assess only one aspect. The first class of benchmarks focuses on evaluating *memorization* through recall or retrieval over static long-context inputs in question answering or summarization settings (Wu et al., 2025; Zhong et al., 2024; Maharana et al., 2024; Hu et al., 2025b), including benchmarks such as LoCoMo (Maharana et al., 2024) and LongMemEval (Wu et al., 2025). In these setups, agents are required to memorize provided conversations or text chunks, and are evaluated on whether they can recall specific information through downstream QA tasks. However, despite being effective at measuring factual recall, such benchmarks do not involve agentic decision-making, environment dynamics, or action-dependent con-

Thank You!