AI Agent and Agentic Workflow based in Large Language Model

Networked Intelligence for Comprehensive Efficiency (NICE) Lab
College of Information Science and Electronic Engineering
Zhejiang University
http://nice.rongpeng.info/

Aug. 25, 2025

Content

Background and Motivation

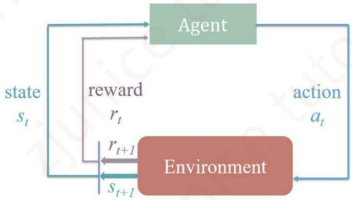
OpenAI's Five-Level Definition of AGI

What are the 5 steps to AGI?

	Name	Description
Level 1	Chatbots	AI with natural conversation language abilities
Level 2	Reasoners	AI's with human-levels of problem solving across a broad range of topics
Level 3	Agents	AI systems that can take actions independently or from human instruction
Level 4	Innovators	AI that can aid in the invention of new ideas and contribute to human knowledge
Level 5	Organizations	AI that is capable of doing all of the work of an organization independently

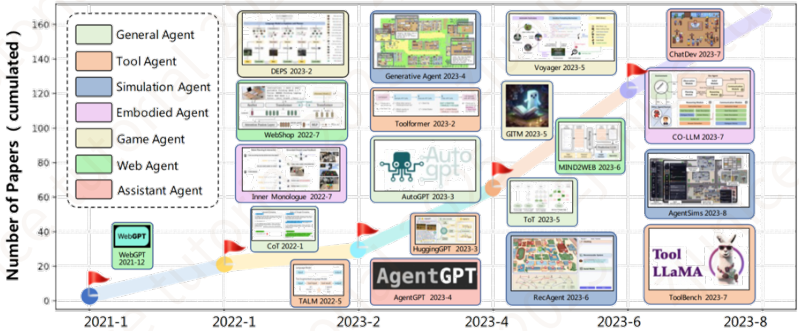
The Evolution of Agents

- **Rule-Based Agents**
  Early agents relied on manually created knowledge bases and expert systems to solve problems but lacked flexibility.
- **Reinforcement Learning-Based Agents**
  - With the rise of machine learning, agents learned strategies via reinforcement learning. AlphaGo demonstrated the power of reinforcement learning in mastering complex games.
  - The problem with RL is its long training time, low sample efficiency, and stability issues in real-world environments.



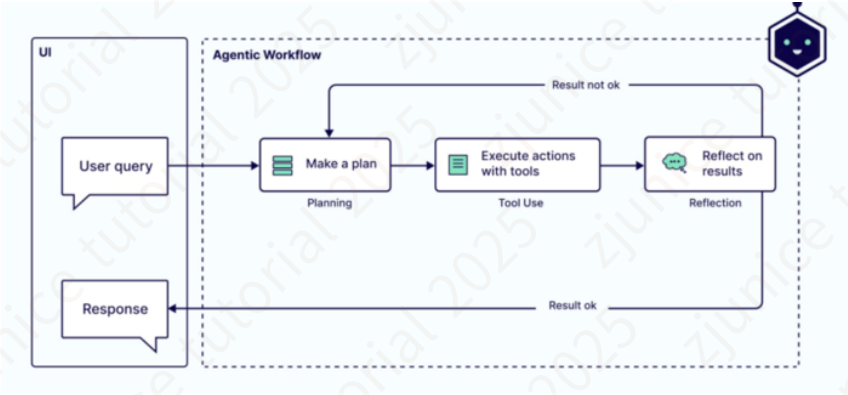
The Evolution of Agents

- LLM-Based Agents
  - Traditional prompting lacks multi-step reasoning and adaptability, while agents enable complex, autonomous task execution.
  - LLM-powered agents, with vast knowledge and natural language interfaces, offer greater flexibility and interpretability.



From Single Agents to Agentic Workflows

- Agents need defined roles and goals, guided by a clear framework.
- Agentic workflows are dynamic, self-guided processes that enable agents to autonomously break down tasks and optimize decision-making.



The Four Core Capabilities of Agents

The Four Core Capabilities of Agents

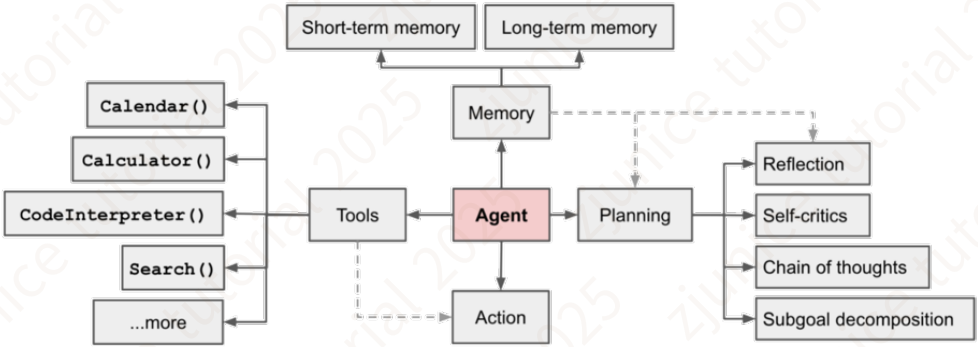
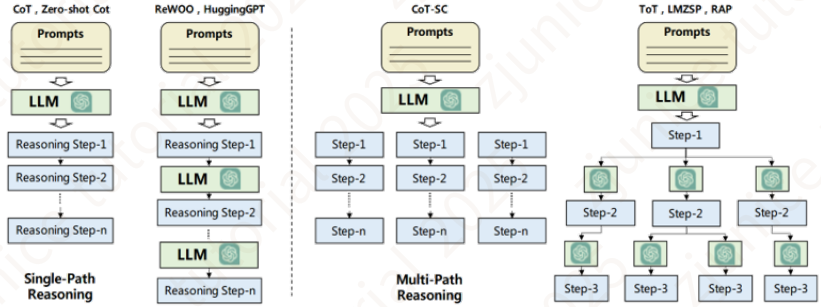


Fig. 1. Overview of a LLM-powered autonomous agent system.

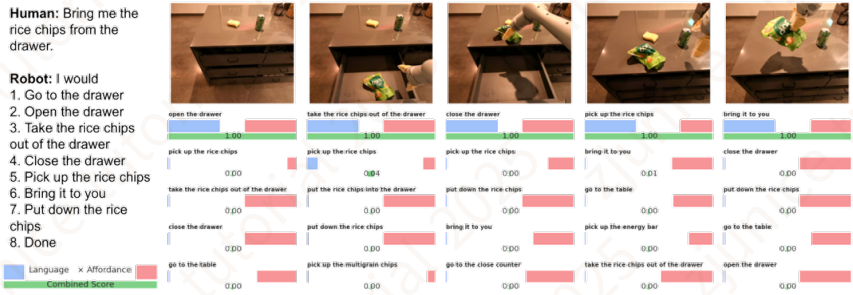
Task Decomposition

- Breaking down a complex problem into smaller, manageable steps, and continuously adjusting the strategy based on feedback.
- LLMs excel in planning by quickly generating flexible strategies using pre-trained knowledge, without the need for extensive trial and error like RL.



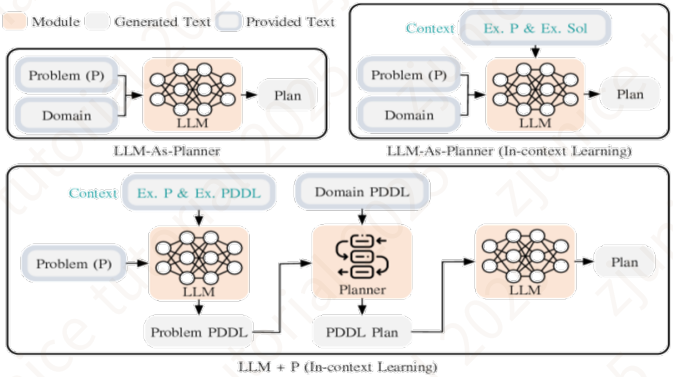
SayCan

- LLMs are prompted to directly predict the actions a robot might take (e.g., [action, object] pairs), which are then re-ranked using an affordance model based on the visual environment;
- LLM then selects feasible and context-appropriate natural language actions.



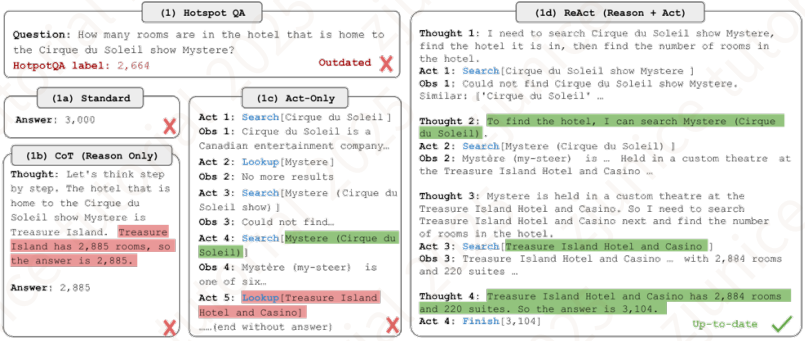
LLM+P

- Use the Planning Domain Definition Language (PDDL) as an intermediary interface to describe planning problems;
- First, LLM convert the task description into formal PDDL, then use an external planner to process the PDDL. Finally, the generated results are converted back to natural language through LLMs.



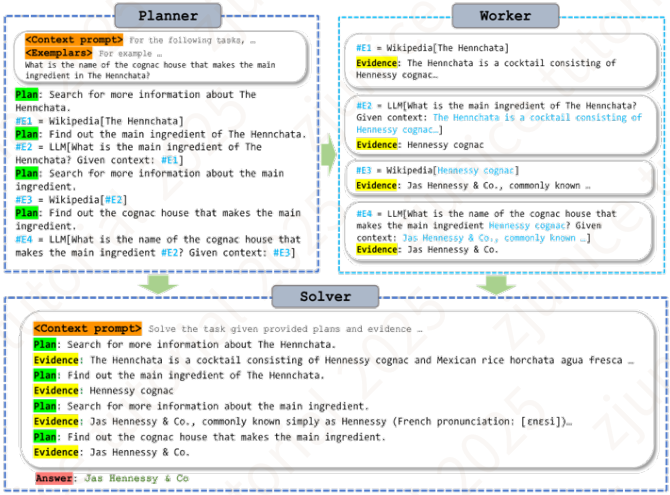
ReAct

- ReAct suggests using a Think-Act-Observe triplet to construct prompts.
- The purpose of the "Think" component is to facilitate higher-level reasoning and planning to guide the agent's behavior.



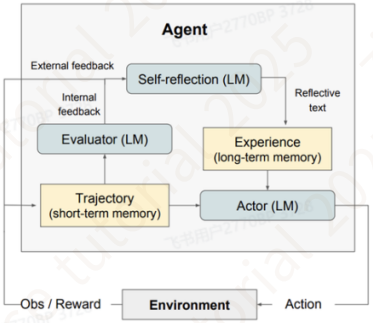
ReWoo

- ReWoo separates planning from external observations.



Self-reflection

- Use a short-term sliding window to capture recent feedback, combined with persistent long-term storage to retain condensed insights.



Algorithm 1 Reinforcement via self-reflection

Initialize Actor, Evaluator, Self-Reflection: M_a, M_e, M_{sr}
Initialize policy $\pi_\theta(a_i|s_i)$, $\theta = \{M_a, mem\}$
Generate initial trajectory using π_θ
Evaluate τ_0 using M_e
Generate initial self-reflection sr_0 using M_{sr}
Set $mem \leftarrow [sr_0]$
Set $t = 0$
while M_e not pass or $t <$ max trials **do**
  Generate $\tau_t = [a_0, o_0, \dots a_i, o_i]$ using π_θ
  Evaluate τ_t using M_e
  Generate self-reflection sr_t using M_{sr}
  Append sr_t to mem
  Increment t
end while
return

Function calling

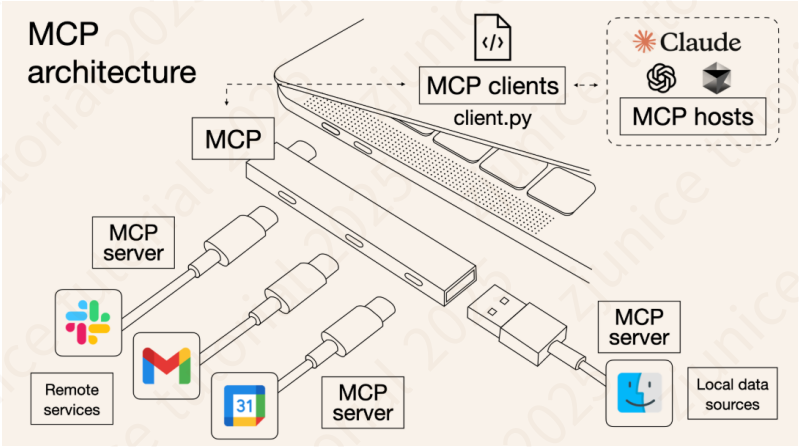
- Allow LLMs to interact with external tools and services through natural language instructions, converting natural language into specific API calls.
- The lack of unified standards, leading to difficulties in reusing various tools and fragmented ecosystems.

```python
def execute_command(command_name, arguments):
    """Execute the command and return the result"""
    memory = get_memory(cfg)

    try:
        if command_name == "google":
            # Check if the Google API key is set and use the official search method
            # If the API key is not set or has only whitespaces, use the unofficial
            # search method
            key = cfg.google_api_key
            if key and key.strip() and key != "your-google-api-key":
                return google_official_search(arguments["input"])
            else:
                return google_search(arguments["input"])
        elif command_name == "memory_add":
            return memory.add(arguments["string"])
        elif command_name == "start_agent":
```

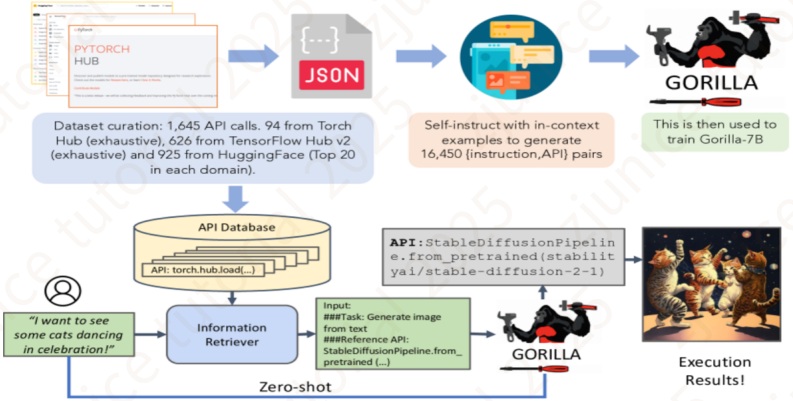
MCP

- MCP (Model-Context Protocol) is an open protocol that provides LLMs with a standardized way to access external capabilities and context.



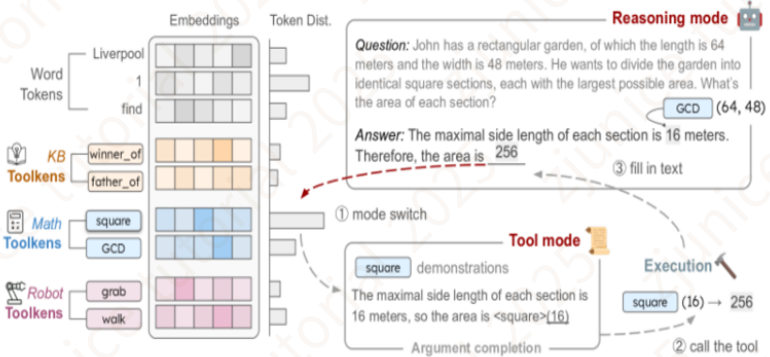
Gorilla

- Integrated with a document retrieval system, which reduces hallucination issues and significantly improves the accuracy and reliability of API calls.
- However, its performance still highly depends on the quality of the API documentation.



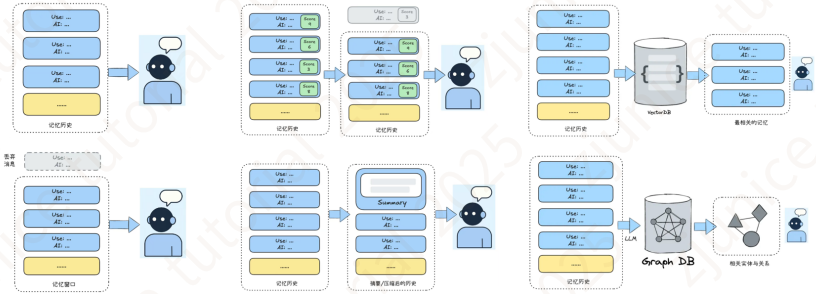
ToolkenGPT

- Combines self-supervised learning with fine-tuning.
- With a few API demonstrations, the model can autonomously learn when to call the API, what parameters to pass, and how to utilize the results returned by the API.
- It can handle dynamically changing API environments and stays synchronized with the latest APIs through the document retrieval system.



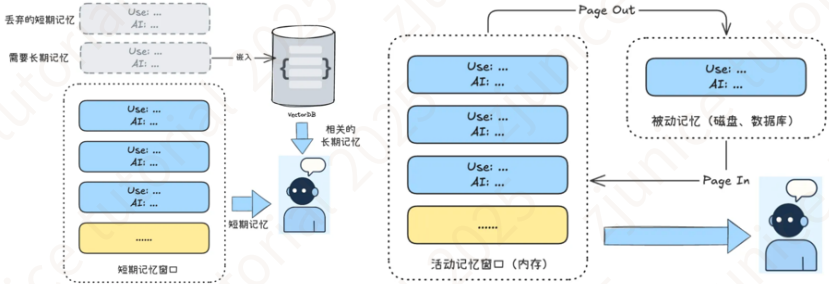
Main methods of agent memory

- Full memory -> sliding window strategy
- Relevance filtering -> Summarization/compression
- Vector database: semantic retrieval memory;
- Knowledge graph: structured memory



Main methods of agent memory

- Hierarchical memory: combining short-term and long-term memory;
- OS-like memory management: simulating memory swap.



Summary of different memory methods

technique name	level	advantage	drawback
Sequential	Beginner	Simple implementation	Very high cost
Sliding Window	Beginner	Cost-effective	Forgets old context
Summarization	Intermediate	Reduces token usage	Loses factual details
Retrieval Based	Advanced	Highly relevant context	Complex implementation
Memory Augmented Transformers	Advanced	Retains critical facts	Expensive extra calls
Hierarchical	Advanced	Hybrid and sophisticated	Complex to manage
Graph Based	Advanced	Understands relationships	Difficult to populate
Compression & Consolidation	Intermediate	Extremely token-efficient	Loses nuance
OS-Like Memory Management	Advanced	Limitless memory potential	Conceptually complex

MIRIX - Multi-Agent Personal Assistant with an Advanced Memory System

- 8 specialized agents working collaboratively
- 6 memory components for organized data storage
- Coordinated workflow for efficient processing

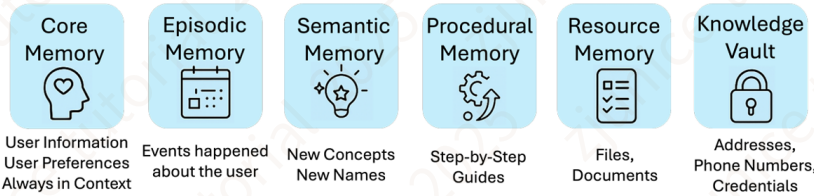


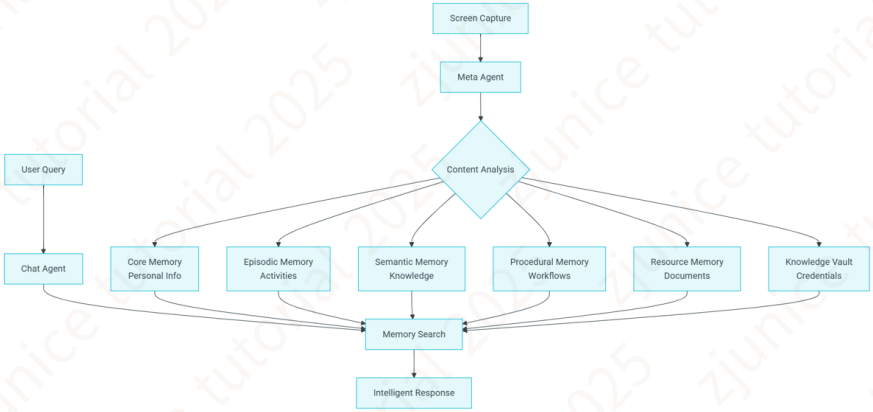
Figure 1: The six memory components of MIRIX, each providing specialized functionality.

MIRIX - Multi-Agent Personal Assistant with an Advanced Memory System

- Natural language conversation interface
  - Handles user queries and conversations
  - Searches across all memory components
  - Synthesizes retrieved information into contextual responses
  - Maintains conversation flow and context
- Central coordinator and content analyzer
  - Analyzes incoming user content (text, images, voice)
  - Determines which memory components need updates
  - Routes the input content to relevant Memory Managers
- Each memory component has a dedicated agent that specializes in managing that specific type of information.

MIRIX - Multi-Agent Personal Assistant with an Advanced Memory System

- Workflow of MIRIX.



MIRIX - Multi-Agent Personal Assistant with an Advanced Memory System

- The memory update process and dialogue retrieval process.

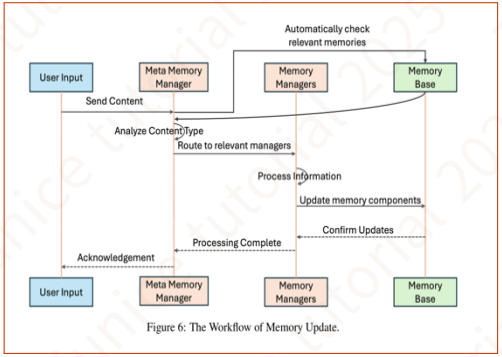


Figure 6: The Workflow of Memory Update.

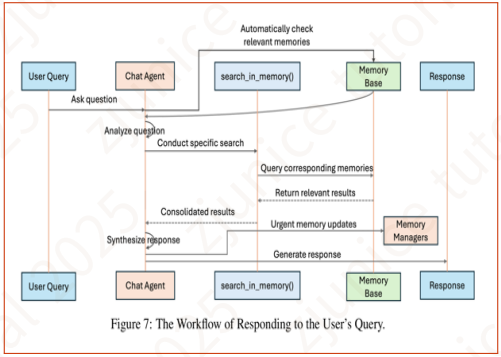
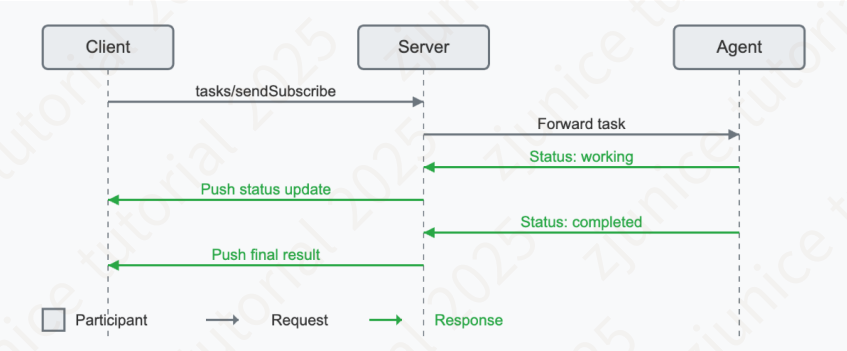


Figure 7: The Workflow of Responding to the User's Query.

Multi-Agent Collaboration

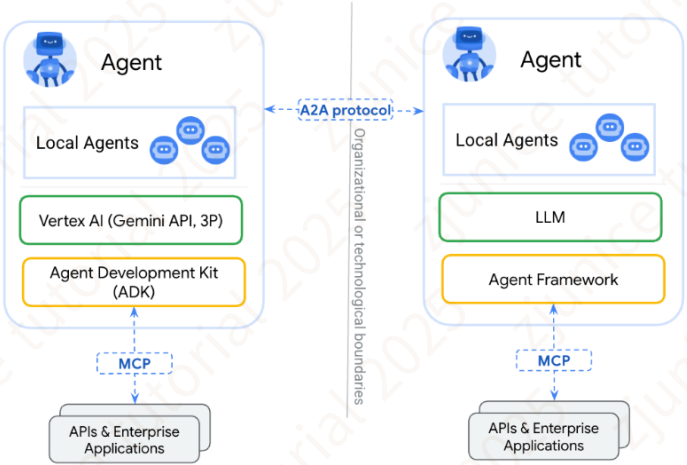
A2A Protocol

- A2A (Agent-to-Agent) protocol enables communication and coordination between multiple agents, allowing them to share information and collaborate on tasks.



MCP vs A2A

- A2A connects agent to agent, while MCP connects agents with tools.



Generative Agents: interactive simulacra of human behavior

- Use generative models to simulate human behavior in virtual worlds and demonstrate their ability to generate reliable individual and group behaviors.
- Consists of three main components: Memory Stream, Reflection and Plan.

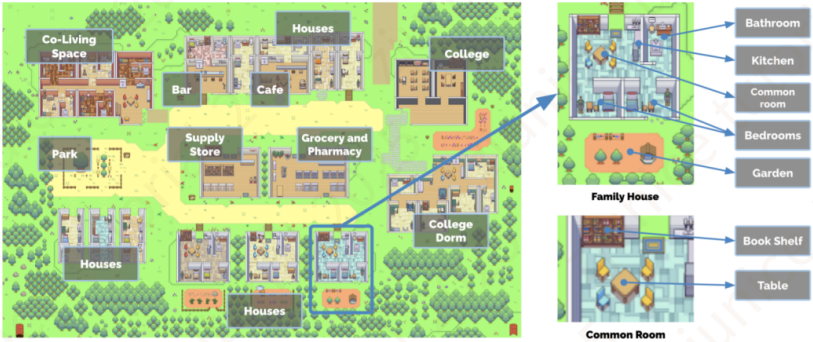


Figure 2: The Smallville sandbox world, with areas labeled. The root node describes the entire world, children describe areas (e.g., houses, cafe, stores), and leaf nodes describe objects (e.g., table, bookshelf). Agent remember a subgraph reflecting the parts of the world they have seen, in the state that they saw them.

Challenges and Research Frontiers

Challenges and Research Frontiers

- Reliability and Stability: Long-term planning can deviate, external states from tools may not align with context, and lack of transactional guarantees (atomicity, idempotency) can amplify errors.
- Memory Governance: Writing guidelines, eviction strategies, conflict resolution, and traceability mechanisms are immature; memory pollution and privacy compliance need systematic governance.
- Collaboration Costs and Convergence: Multi-agent communication/review is costly and prone to loops; clear termination conditions, budgets, and monitoring metrics are needed.
- Security: Prompt injection, privilege escalation, data leakage; input sanitization, least privilege, isolation, and auditing are necessary.
- Evaluation and Verifiability: How to standardize metrics like task completion rate, cost, and latency? How to generate verifiable execution traces?

References

References

1 Wang, Lei, et al. "A survey on large language model based autonomous agents." Frontiers of Computer Science 18.6 (2024): 186345.

2 Ahn M, Brohan A, Brown N, et al. Do as i can, not as i say: Grounding language in robotic affordances[J]. arxiv preprint arxiv:2204.01691, 2022.

3 Liu B, Jiang Y, Zhang X, et al. Llm+p: Empowering large language models with optimal planning proficiency[J]. arxiv preprint arxiv:2304.11477, 2023.

4 Yao, Shunyu, et al. "React: Synergizing reasoning and acting in language models." International Conference on Learning Representations (ICLR). 2023.

5 Xu, Binfeng, et al. "Rewoo: Decoupling reasoning from observations for efficient augmented language models." arXiv preprint arXiv:2305.18323 (2023).

References

6 Shinn, Noah, et al. ”Reflexion: Language agents with verbal reinforcement learning.” Advances in Neural Information Processing Systems 36 (2023): 8634-8652.

7 Patil S G, Zhang T, Wang X, et al. Gorilla: Large language model connected with massive apis[J]. arxiv preprint arxiv:2305.15334, 2023.

8 Hao S, Liu T, Wang Z, et al. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings[J]. Advances in neural information processing systems, 2023, 36: 45870-45894.

9 Wang, Yu, and Xi Chen. ”Mirix: Multi-agent memory system for llm-based agents.” arXiv preprint arXiv:2507.07957 (2025).

10 Park, Joon Sung, et al. ”Generative agents: Interactive simulacra of human behavior.” Proceedings of the 36th annual acm symposium on user interface software and technology. 2023.

Thank you

Networked Intelligence for Comprehensive Efficiency (NICE) Lab
College of Information Science and Electronic Engineering
Zhejiang University
https://nice.rongpeng.info