



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Advanced Natural Language Processing

## Lecture 18: LLM-based Agent (1)

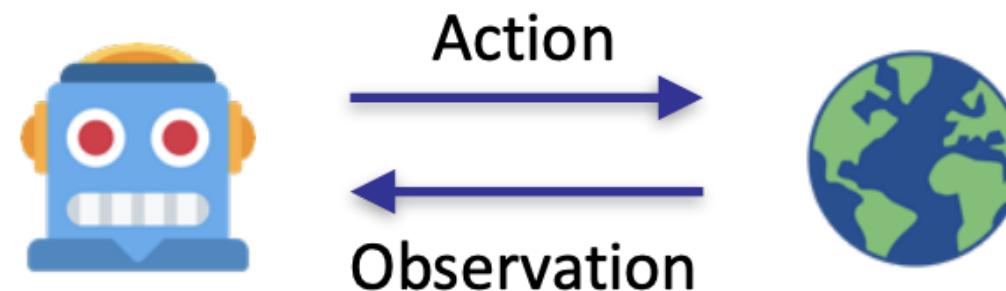


陈冠华 CHEN Guanhua

Department of Statistics and Data Science

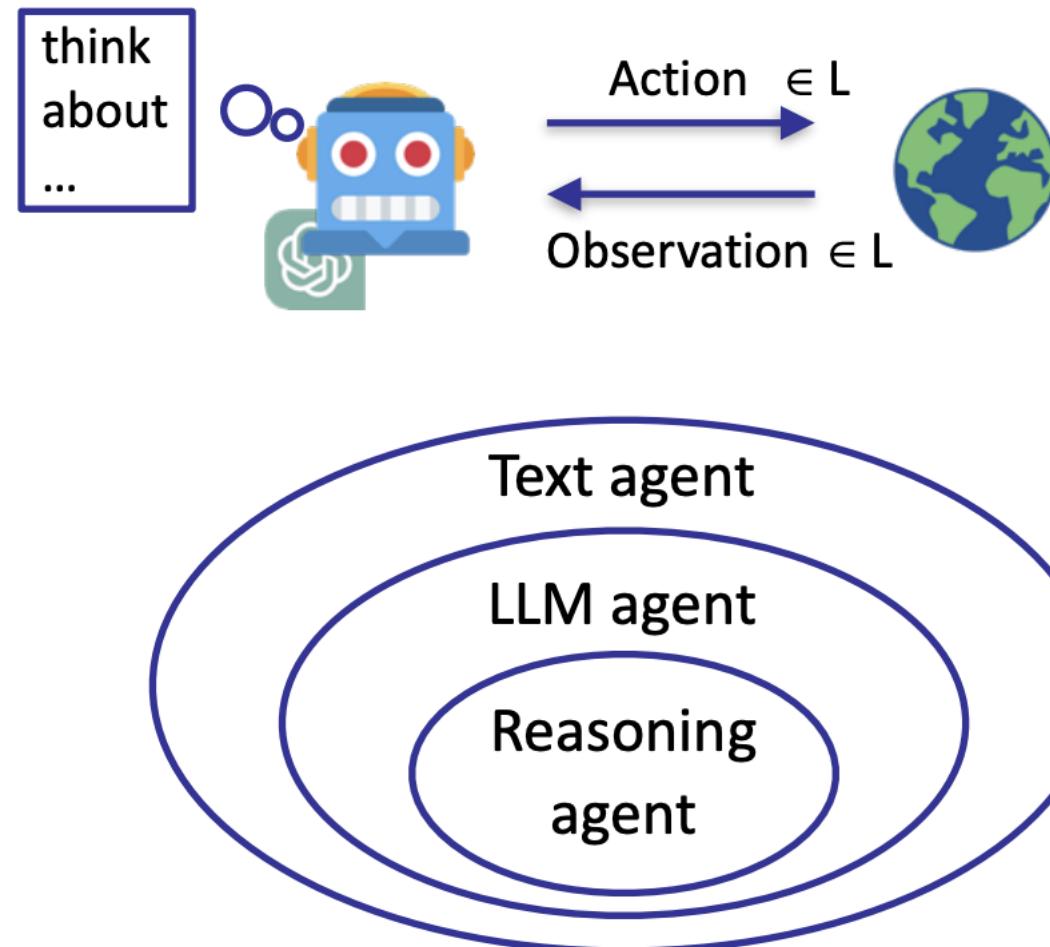
# What is “agent”?

- An “intelligent” system that interacts with some “environment”
  - Physical environments: robot, autonomous car, ...
  - Digital environments: DQN for Atari, Siri, AlphaGo, ...
  - Humans as environments: chatbot
- Define “agent” by defining “intelligent” and “environment”
  - It changes over time
- How would you define “intelligent”?



# What is “agent”?

- Level 1: Text agent
  - Uses text action and observation
  - Examples: ELIZA, LSTM-DQN
- Level 2: LLM agent
  - Uses LLM to act
  - Examples: SayCan, Language Planner
- Level 3: Reasoning agent
  - Uses LLM to reason to act
  - Examples: ReAct, AutoGPT
  - The key focus of the field



# Markov Decision Process



- A Markov Decision Process (MDP) is a mathematical framework used in **reinforcement learning** to formally describe the interaction between **an agent** and **its environment**.
  - The environment of an agent system as a partially observable Markov Decision Process (POMDP), represented as a tuple:

$$E = (\mathcal{G}, \mathcal{S}, \mathcal{A}, T, R, \Omega, O, \gamma),$$

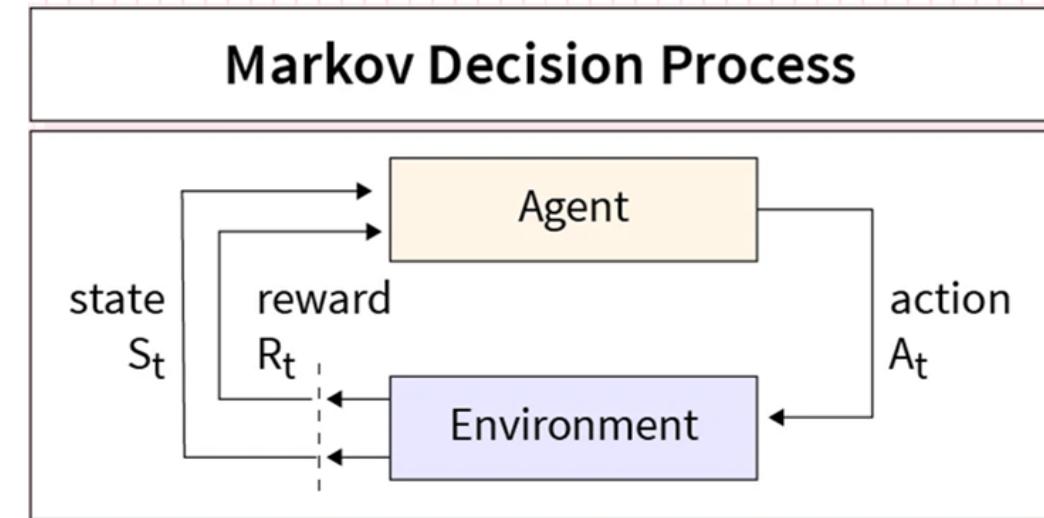
where:

- $\mathcal{G}$  is a set of potential goals. Each  $g \in \mathcal{G}$  is a task objective that the agent needs to achieve, e.g., a user query.
- $\mathcal{S}$  is a set of states. Each  $s \in \mathcal{S}$  represents the internal state of the environment.
- $\mathcal{A}$  is a set of actions. Each action  $a \in \mathcal{A}$  can be a combination of textual reasoning, retrieval of external knowledge, and tool calls.
- $T$  is the state transition probability function which takes a state-action pair  $(s, a)$  and outputs the probability distribution  $T(s'|s, a)$  of the next state.
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathcal{R}$  is the feedback/reward function, conditioned on the specific goal  $g \in \mathcal{G}$ . The feedback  $r = R(s, a, g)$  typically takes the form of a scalar score or textual feedback.
- $\Omega$  is a set of observations accessible to the agent.
- $O$  is the observation probability function which takes a state-action pair  $(s, a)$  and outputs the probability distribution  $O(o'|s, a)$  of the next observation for the agent.
- $\gamma$  is the discount factor.

# Markov Decision Process



- At each time step:
  - The environment provides a state  $s_t$ ,
  - The system selects an action  $a_t$ ,
  - The environment returns a reward  $r_t$  and transitions to the next state  $s_{t+1}$ .
- Who is choosing actions based on the state?
  - This is the agent. An agent is the decision-making entity in an MDP.



# ELIZA (1966): Text agent via rule design



- Domain specific
- Requires manual design
- Cannot work beyond simple domains

```
Welcome to
EEEEE   LL      IIII    ZZZZZZ   AAAA
EE       LL      II      ZZ      AA      AA
EEEEE   LL      II      ZZZ     AAAAAAAA
EE       LL      II      ZZ      AA      AA
EEEEEE  LLLLLL  IIII    ZZZZZZ   AA      AA

Eliza is a mock Rogerian psychotherapist.
The original program was described by Joseph Weizenbaum in 1966.
This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?
YOU: Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU: They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU: Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU: He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU: It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

[全球第一个聊天机器人是怎样诞生的？ - 知乎](#)

# LSTM-DQN (2015): Text agent via RL

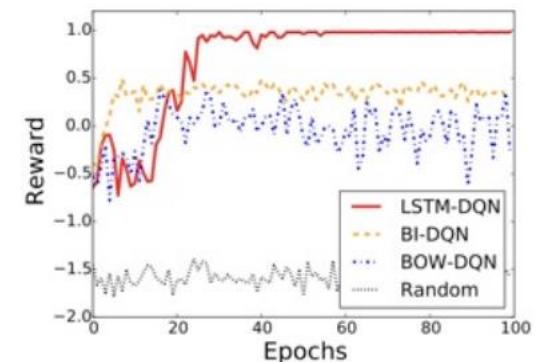
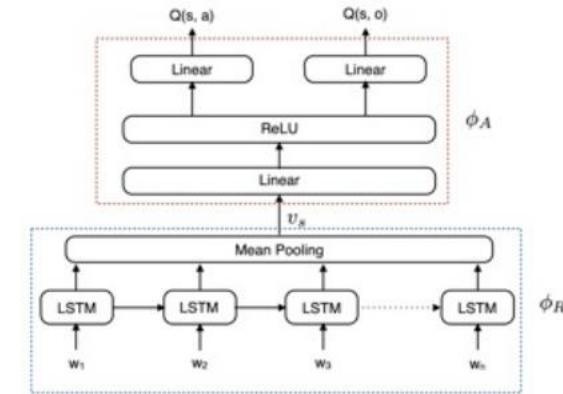


- Domain specific
- Requires scalar reward signals
- Requires extensive training

**State 1: The old bridge**  
You are standing very close to the bridge's eastern foundation. If you go east you will be back on solid ground ... The bridge sways in the wind.

**Command: Go east**

**State 2: Ruined gatehouse**  
The old gatehouse is near collapse. Part of its northern wall has already fallen down ... East of the gatehouse leads out to a small open area surrounded by the remains of the castle. There is also a standing archway offering passage to a path along the old southern inner wall.  
Exits: Standing archway, castle corner, Bridge over the abyss



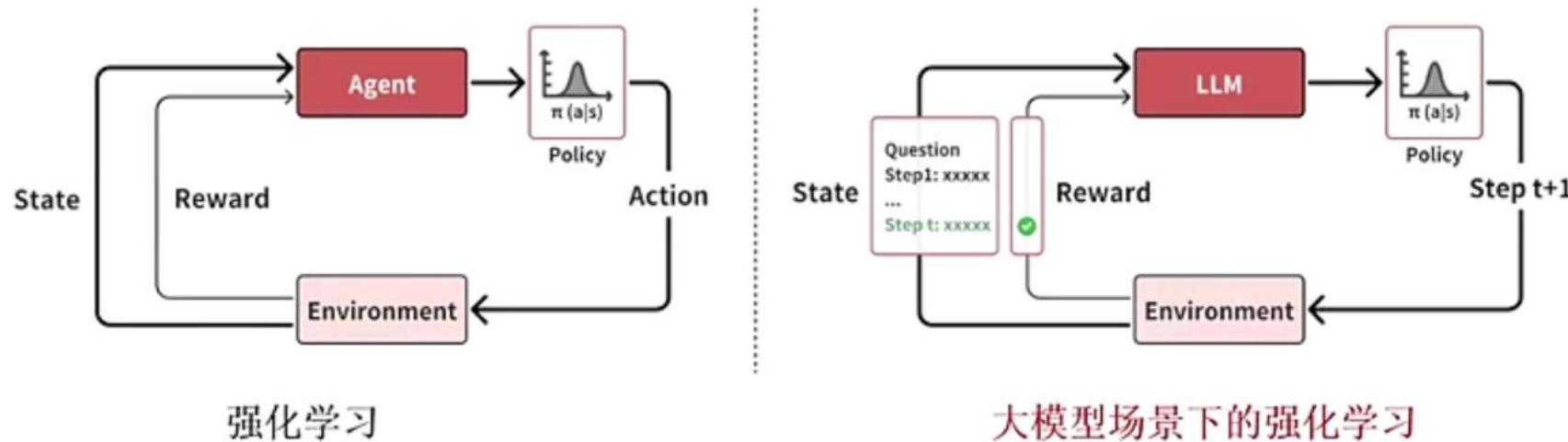
[1506.08941] Language Understanding for Text-based Games Using Deep Reinforcement Learning

Language understanding for text-based games using deep reinforcement learning. EMNLP 2015.

# Reasoning LLMs based on RL



**Agent** => LLM  
**Action** => Next Token/Step/Solution  
**State** => LLM inputs  
**Policy:**  $\pi(a|s)$



# LLM-based Agents



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Autonomous decision-makers capable of
  - Perceiving
  - Reasoning
  - Planning
  - Invoking tools
  - Maintaining memory
  - Adapting strategies over extended horizons
- In **partially observable, dynamic** environments

# Tutorials



## AI Agents for Beginners - A Course

Version 2

# AI Agents

For Beginners

Intro to Agents    Agentic RAG    Metacognition    Agentic Memory

Agentic Frameworks    Trustworthy Agents    Production Agents    Agent Evals

Design Patterns    Planning Design    Agentic Protocols    Computer Use

Tool Use    Multi-Agents    Context Engineering    Agent Deployment

Local Agents

Secure Agents

[aka.ms/ai-agents-beginners](http://aka.ms/ai-agents-beginners)

A course teaching everything you need to know to start building AI Agents

[microsoft/ai-agents-for-beginners](https://microsoft/ai-agents-for-beginners): 12 Lessons to Get Started Building AI Agents

Lesson	Text & Code	Video	Extra Learning
Intro to AI Agents and Agent Use Cases	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Exploring AI Agentic Frameworks	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Understanding AI Agentic Design Patterns	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Tool Use Design Pattern	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Agentic RAG	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Building Trustworthy AI Agents	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Planning Design Pattern	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Multi-Agent Design Pattern	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Metacognition Design Pattern	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
AI Agents in Production	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Using Agentic Protocols (MCP, A2A and NLWeb)	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Context Engineering for AI Agents	<a href="#">Link</a>	<a href="#">Video</a>	<a href="#">Link</a>
Managing Agentic Memory	<a href="#">Link</a>	<a href="#">Video</a>	
Exploring Microsoft Agent Framework	<a href="#">Link</a>		
Building Computer Use Agents (CUA)	Coming Soon		
Deploying Scalable Agents	Coming Soon		
Creating Local AI Agents	Coming Soon		
Securing AI Agents	Coming Soon		

# Code and Frameworks



## Mult-Agent Frameworks

- [HKUDS/DeepCode](#) ★ 9758 Python - DeepCode is a system to convert research papers and natural language into CLI and web interfaces.
- [awslabs/multi-agent-orchestrator](#) ★ 4775 Python - An open-source framework for managing multiple AI agents, providing intelligent routing and context management.
- [i-am-bee/beeaai-framework](#) ★ 2941 TypeScript - A framework for building scalable and flexible multi-agent integration with various AI models and tools.
- [jgravelle/AutoGroq](#) ★ 1466 Python - AutoGroq generates tailored teams of AI agents to efficiently solve problems without manual configuration.
- [microsoft/JARVIS](#) ★ 24437 Python - JARVIS integrates language models with expert AI models from Hugging Face to perform complex AI tasks.
- [microsoft/agent-framework](#) ★ 4807 Python - A framework for building, orchestrating, and deploying AI agents for Python and .NET.

## AI Agent Ecosystems

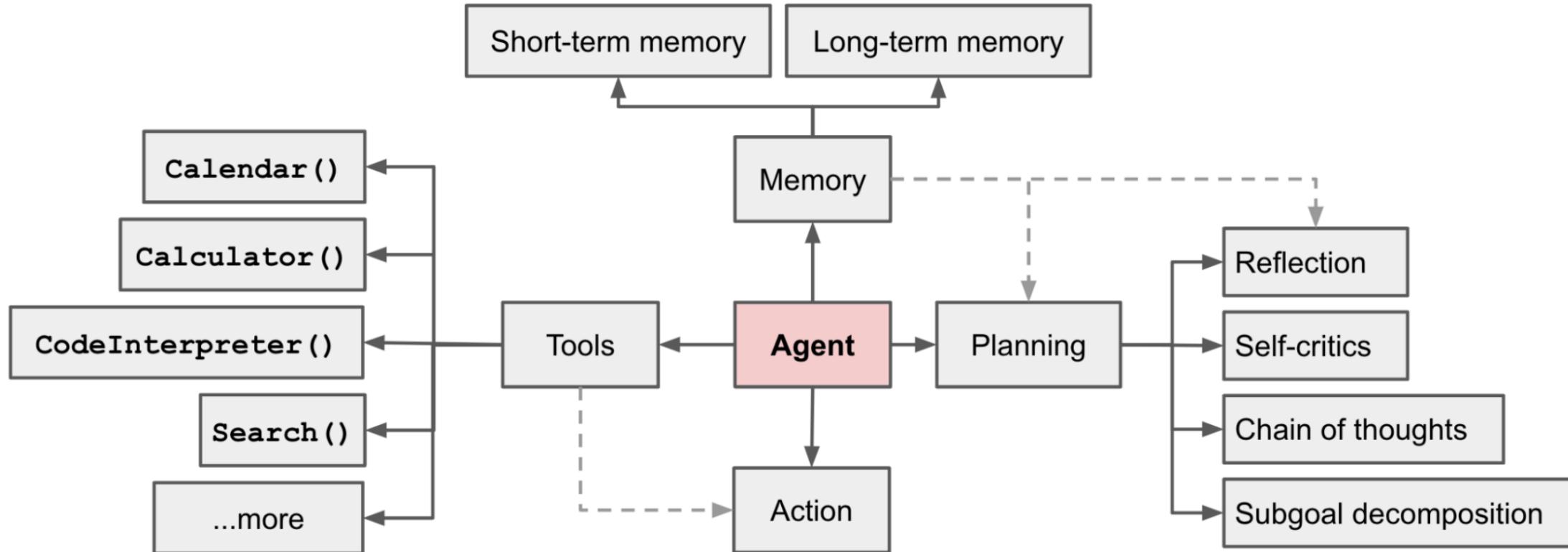
- [FoundationAgents/MetaGPT](#) ★ 59249 Python - A company that simulates a software company by assigning GPT-4 development from natural language requirements.
- [OpenBMB/ChatDev](#) ★ 27684 Python - ChatDev is a company framework that enables collaborative, intelligent agents through specialized roles.
- [VRSEN/agency-swarm](#) ★ 3866 Python - Agency Swarm is a framework that enables the creation and management of AI agents using the latest OpenAI Assistants API, designed for agent roles and efficient communication.
- [VoltAgent/awesome-claude-code-subagents](#) ★ 439 Python - A collection of specialized Claude Code AI subagents for science, and business operations, maintained by VoltAgent.
- [agi-inc/agent-protocol](#) ★ 1436 Python - A specification providing a common interface for AI agents, simplifying development and benchmarking.
- [camel-ai/owl](#) ★ 18301 Python - OWL is an open-source collaboration that automates real-world tasks using a comprehensive set of toolkits.

## Reasoning Frameworks

- [NirDiamant/Controllable-RAG-Agent](#) ★ 1466 Jupyter Notebook - A controllable Retrieval-Augmented Generation solution for autonomous agents designed for complex question answering and hallucination prevention.
- [anysphere/prompt](#) ★ 2714 TypeScript - A prompt priority system to efficiently construct token-level prompts based on React's component design.
- [bytedance/deer-flow](#) ★ 17936 TypeScript - Deer Flow is a framework that integrates language models with web technologies to facilitate comprehensive AI-powered research and development.
- [friuns2/BlackFriday-GPTs-Prompts](#) ★ 8752 Python - A collection of prompts for GPT-3 and GPT-4 that allow users to generate AI-generated content across various domains, enabling user interaction with AI models.
- [guidance-ai/guidance](#) ★ 20895 Jupyter Notebook - A library that enables efficient and fine-grained AI reasoning, producing structured, high-quality outputs with reduced latency.

## Awesome AI Agents

# LLM-based Agents

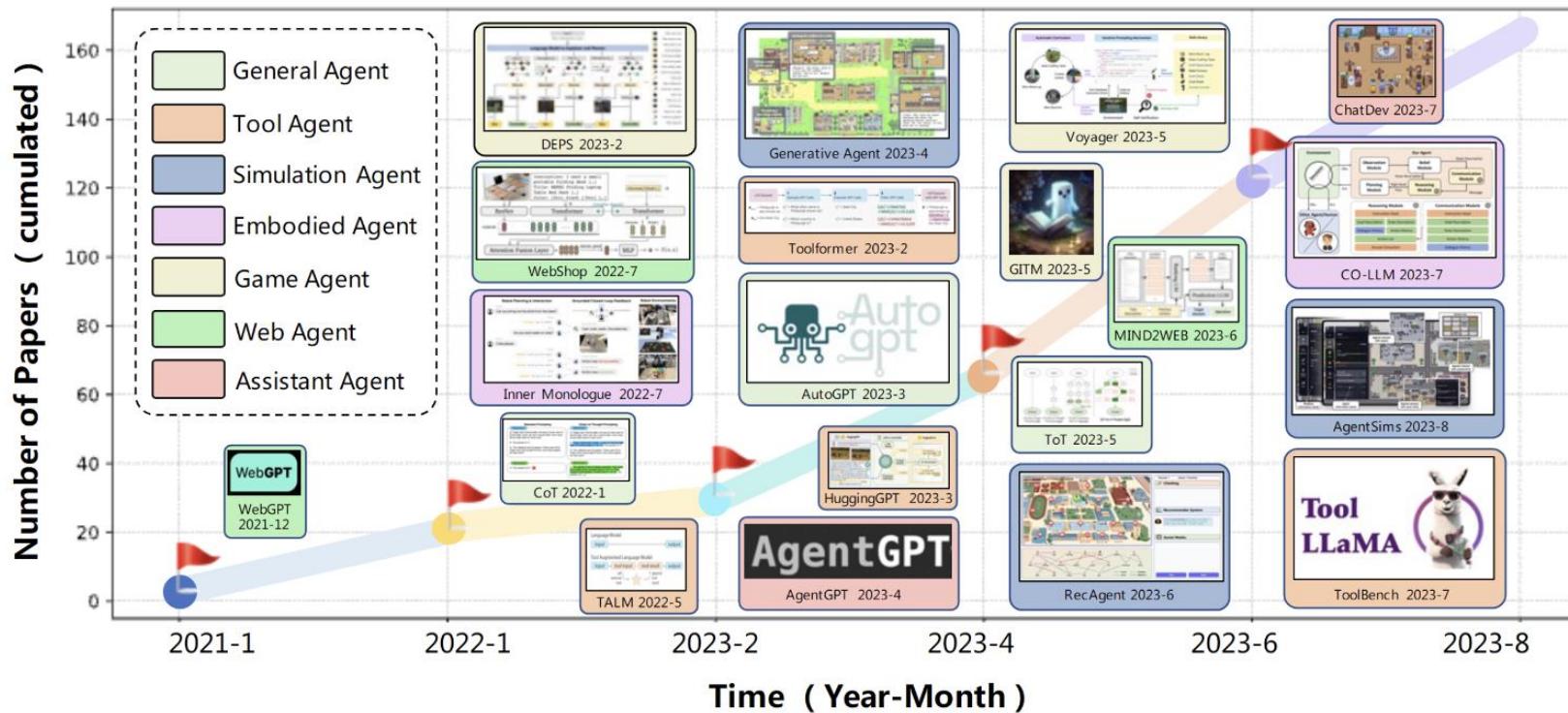


[LLM Powered Autonomous Agents | Lil'Log](#)

# LLM-based Agents



- Towards general artificial intelligence driven by natural language



## GPTs

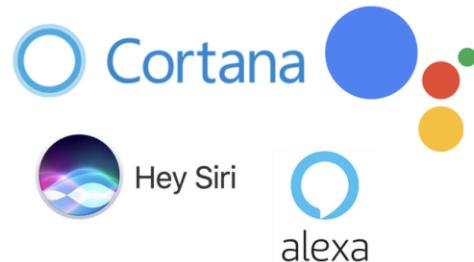
A collection of GPT-based agents, each with a unique personality and function:

- AnalyzePaper**: Serves as a writing assistant, adept at reviewing research papers and articles.
- RubyGPT**: Your Ruby coding assistant.
- Showtimes**: Transcribe and summarize audio.
- Hacker Art**: Create badass hacker art and profile pics with a flair for design.
- Product Coach**: Guiding your product journey with expert insights.
- AstrologyGPT**: Astrology guide for celestial insights.
- GetPaths**: Specializes in analyzing web content, including HTTP traffic.
- FixGPT**: If you’re “unified” chatGPT doesn’t know it, can browse the web.
- II Funny Foto Maker**: Witty, clever photo fun with a friendly touch!
- II Priest of Sins**: A humorous confessor, adaptive to users’ languages and open to discussing relationships.
- II Procrastination Buster.ii**: I’m the quirky voice of reason that will get you from what you’re doing right now to the thing you know should be doing.
- II Socratic Habit Mentor.ii**: Pick a habit you’d like to make or break... and we’ll engage in a Socratic dialogue.
- III her**: Your warm and approachable confidante, here for genuine, trust-filled conversations.
- II Bro psychologist**: Friendly psychologist, like talking to a friend.
- II Image Generation with Selfcritique & Improvement**: More accurate and easier image generation with self critique & improvement! try it now
- II VoxBox.ii**: VoxBox is an intelligent GPT tool for organizing, proofreading text, and answering queries based on user-provided data.
- II Dios Coach**: Orienta a las Personas Hacia la Superación Personal y el Crecimiento Personal en Español.
- II Dios del Dinero**: GOAI del Asesoramiento de Inversiones, Análisis de Mercado y Estrategias Financieras Personalizadas en Español.
- II Drink & Play**: Your ultimate party game guide.

# Why Do We Want Agents?



## Natural Language Interfaces to Computers



### Virtual Assistants

- Set an alarm at 7 AM
- Remind me for the meeting at 5pm
- Play Jay Chou's latest album

```
Untitled-1
1 my_list = [3, 5, 1]
2 sort in descending order ↗
3 sorted(my_list, reverse=True)
4
5
master* ⌂ Python 3.6.5 64-bit ⌂
```

### Natural Language Programming

- Sort my\_list in descending order
- Copy my\_file to home folder
- Dump my\_dict as a csv file output.csv

## Tool Integrations into Chatbots

### ChatGPT plugins

We've implemented initial support for plugins in ChatGPT. Plugins language models help ChatGPT do computations, o

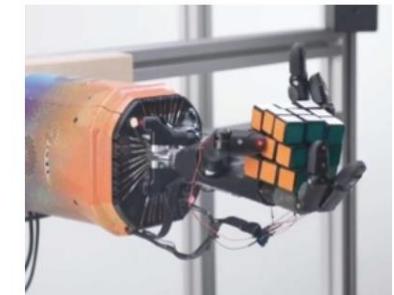
#### ChatGPT plugins

<b>Expedia</b> Bring your trip plans to life—get there, stay there, find things to see and do.	<b>FiscalNote</b> Provides and enables access to select market-leading, real-time data sets for legal, political, and regulatory data and information.	<b>Instacart</b> Order from your favorite local grocery stores.	<b>KAYAK</b> Search for flights, stays and rental cars. Get recommendations for all the places you can go within your budget.
<b>Klarna Shopping</b> Search and compare prices from thousands of online shops.	<b>Milo Family AI</b> Giving parents superpowers to turn the manic to magic, 20 minutes each day. Ask: Hey Milo, what's magic today?	<b>OpenTable</b> Provides restaurant recommendations, with a direct link to book.	<b>Shop</b> Search for millions of products from the world's greatest brands.
<b>Speak</b> Learn how to say anything in another language with Speak, your AI-powered language tutor.	<b>Wolfram</b> Access computation, math, curated knowledge & real-time data through Wolfram Alpha and Wolfram Language.	<b>Zapier</b> Interact with over 5,000+ apps like Google Sheets, Trello, Gmail, HubSpot, Salesforce, and more.	

# Agent Applications



- Autonomous Vehicle
- AlphaGo (DeepMind)
- ChatGPT (OpenAI)
- OpenAI Five (Dota 2)
- Amazon Alexa
- Robot Manipulator



# A Simple Case: Question Answering



Q: what is  $1 + 2$ ?



A: 3

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?



Requires reasoning

Q: who is the latest UK PM?



Requires knowledge

Q: what is the prime factorization of 34324329?



Requires computation

# A Simple Case: Question Answering



- Code augmentation for computation

Question: In Fibonacci sequence, it follows the rule that each number is equal to the sum of the preceding two numbers. Assuming the first two numbers are 0 and 1, what is the 50th number in Fibonacci sequence?

The first number is 0, the second number is 1, therefore, the third number is  $0+1=1$ . The fourth number is  $1+1=2$ . The fifth number is  $1+2=3$ . The sixth number is  $2+3=5$ . The seventh number is  $3+5=8$ . The eighth number is  $5+8=13$ .  
..... (Skip 1000 tokens)  
The 50th number is 32,432,268,459.

CoT

32,432,268,459



```
length_of_fibonacci_sequence = 50
fibonacci_sequence = np.zeros(length_of_)
fibonacci_sequence[0] = 0
fibonacci_sequence[1] = 1
for i in range(3, length_of_fibonacci_sequence):
    fibonacci_sequence[i] = fibonacci_sequence[i-1] +
        fibonacci_sequence[i-2]
ans = fibonacci_sequence[-1]
```

PoT

python

12,586,269,025



Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks

# A Simple Case: Question Answering



- Retrieval-augmented generation (RAG) for knowledge
  - Answer knowledge-intensive questions with
    - Extra corpora
    - A retriever (e.g., [BM25](#), [DPR](#), etc.)
  - What if there's no corpora? (e.g. who's the latest PM?)



# A Simple Case: Question Answering



- Tool use
  - Special tokens to invoke tool calls for
    - Search engine, calculator, etc.
    - Task-specific models (translation)
    - APIs
  - Unnatural format requires task/tool-specific fine-tuning
  - Multiple tool calls?

## A weather task:

how hot will it get in NYC today? |**weather** lookup region=NYC |**result** precipitation chance: 10, high temp: 20c, low-temp: 12c |**output** today's high will be 20C

[TALM: Tool Augmented Language Models](#)

Out of 1400 participants, 400 (or [\[Calculator\(400 / 1400\) → 0.29\]](#) 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [\[MT\("tortuga"\) → turtle\]](#) turtle.

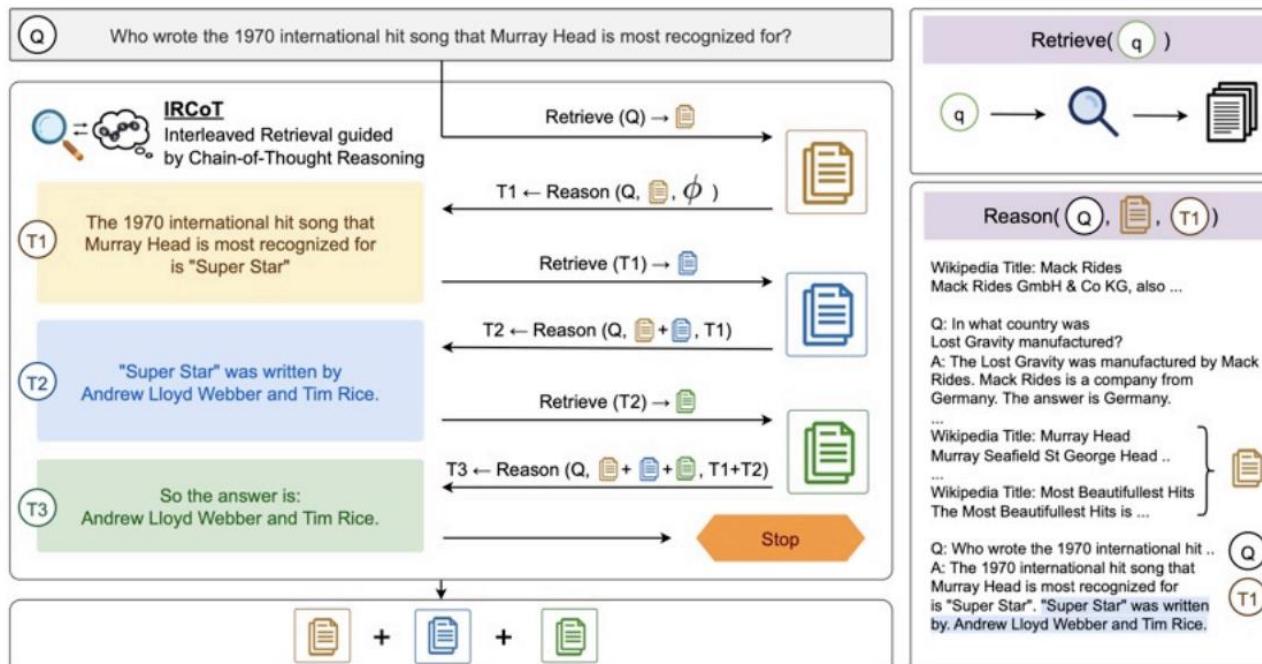
The Brown Act is California's law [\[WikiSearch\("Brown Act"\) → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.\]](#) that requires legislative bodies, like city councils, to hold their meetings open to the public.

[Toolformer: Language Models Can Teach Themselves to Use Tools](#)

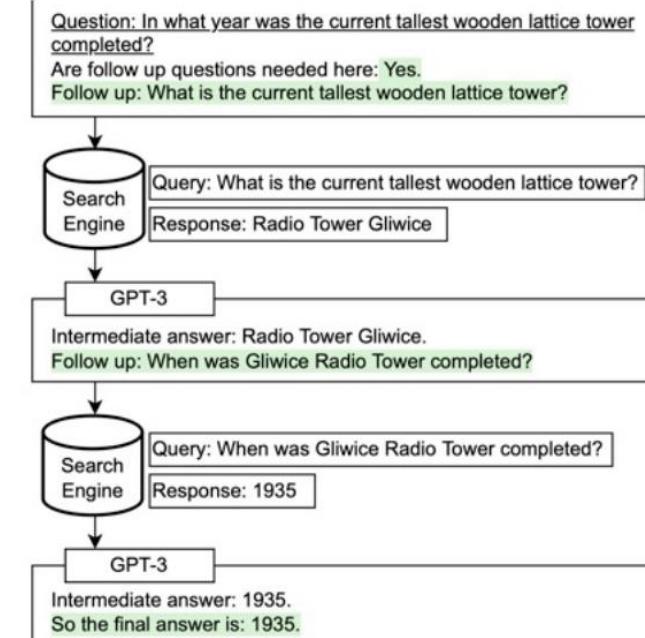
# A Simple Case: Question Answering



- What if both knowledge and reasoning are needed?



Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions

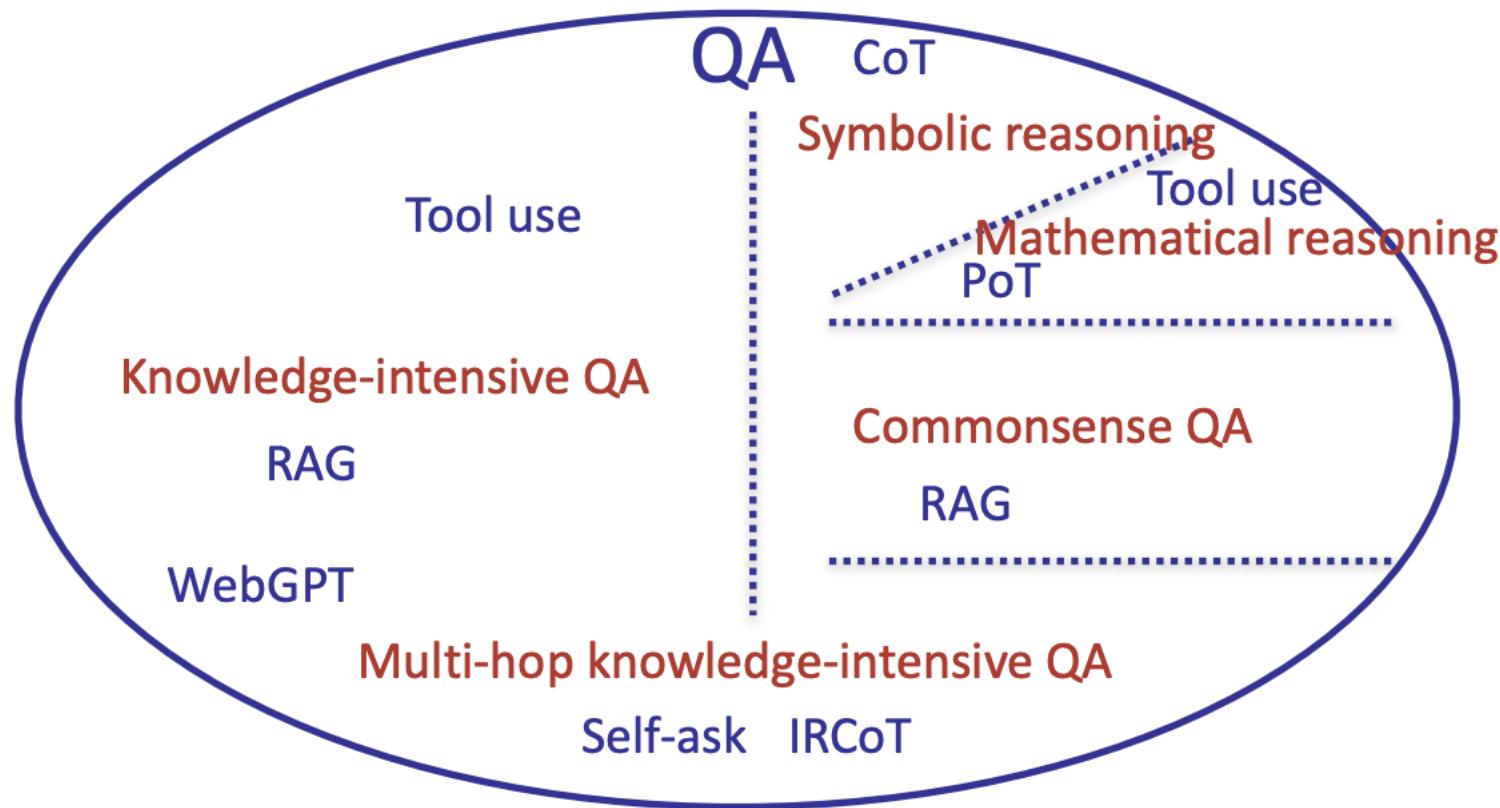


Measuring and Narrowing the Compositionality Gap in Language Models

# A Simple Case: Question Answering



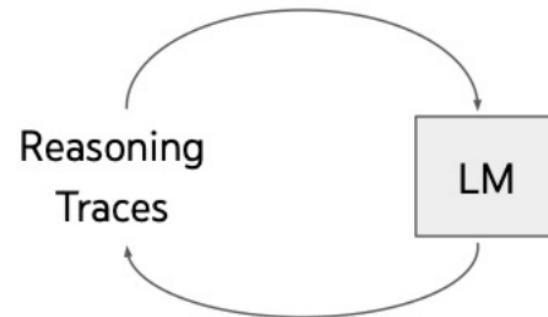
- Can we have a simple, unifying solution?



# Reasoning or Acting



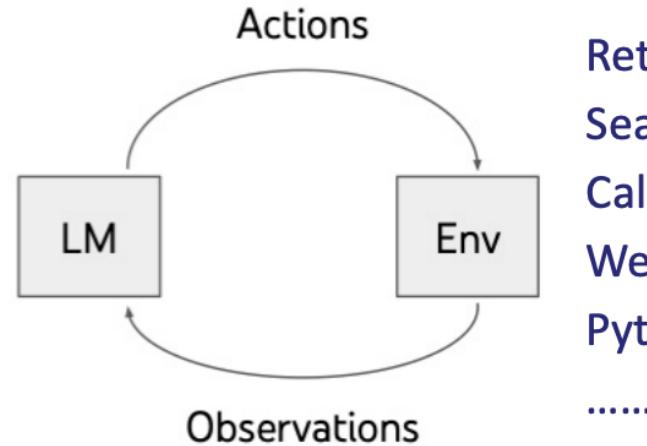
## CoT



Flexible and general to augment test-time compute

Lack of external knowledge and tools

## RAG/Retrieval/Code/Tool use



Lack of reasoning

Flexible and general to augment knowledge, computation, feedback, etc.

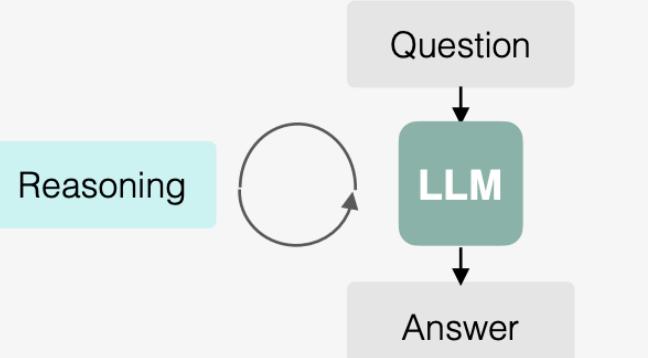
Retrieval  
Search engine  
Calculator  
Weather API  
Python  
.....

# Reasoning&Acting

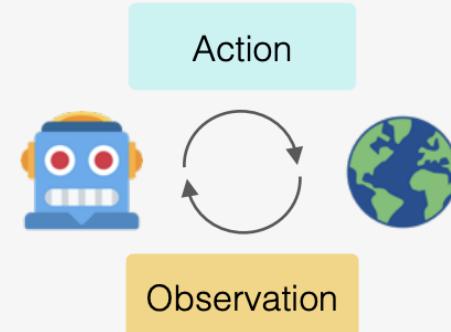


- ReAct [[link](#)]

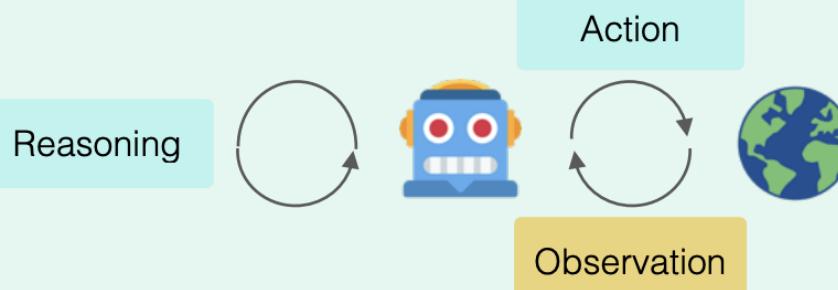
**Reasoning** (update internal belief)



**Acting** (obtain external feedback)



**ReAct**: a new paradigm of agents that **reason and act**



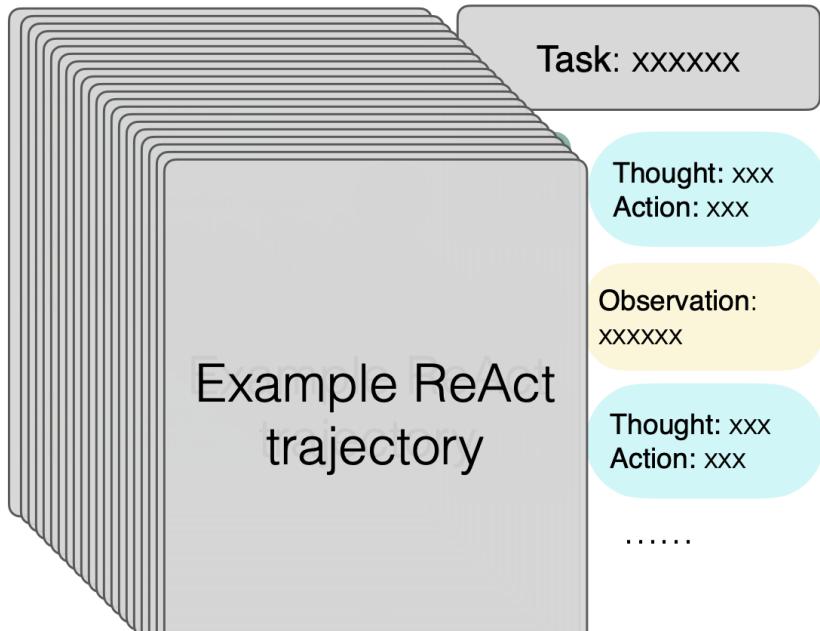
- **Synergy** of reasoning and acting
- **Simple** and intuitive to use
- **General** across domains

- ReAct is simple and intuitive to use

ReAct supports

- One-shot prompting
- Few-shot prompting
- Fine-tuning

Human prompt  
LLM output  
Environment feedback



## A zero-shot ReAct prompt

You are an agent that answers questions by using two actions:

- search[query]: Google search the query. You can also use it to calculate math.
- finish[answer]: return the answer

Your generation should have the following format:  
Thought: to analyze the observation and inform the action.  
Action: your action

Question: If I have seven trillion dollars today, can I buy Apple, Nvidia, and Microsoft? If not, how much more money do I need?

[langchain-ai/react-agent: LangGraph template for a simple ReAct agent](#)



- Synergy: acting support reasoning, reasoning guides acting

**LLM**  
(GPT-4)

Thought: I need to find the current market capitalization of Apple, Nvidia, and Microsoft to understand how much it would cost to buy all of them.  
Action: search["current market capitalization of Apple, Nvidia, and Microsoft"]

Observation: "Nvidia's market cap stood at \$1.832 trillion as of about 10 a.m. ET Wednesday, while Alphabet's was at \$1.814 trillion, bringing Nvidia behind just Microsoft (MSFT) at \$3.015 trillion and Apple (AAPL) at \$2.836 trillion."

Thought: Now that I have the market cap for all companies, let's add them together and see if \$7 trillion is enough.  
Action: search["1.832 trillion + 3.015 trillion + 2.836 trillion"]

Observation: 7683000000000

Thought: The amount of money needed in addition to the \$7 trillion is \$683 billion.  
Action: finish["You need an additional \$683 billion to be able to buy Apple, Nvidia, and Microsoft."]



**LLM**  
(GPT-4)

Thought: I need to find the current market capitalization of Apple, Nvidia, and Microsoft to understand how much it would cost to buy all of them.  
Action: search["current market capitalization of Apple, Nvidia, and Microsoft"]

Observation: nothing found!

Thought: The search did not give any results. I will need to search for each company's market capitalization individually.  
Action: search["current market capitalization of Apple"]

Observation: 'price': 180.75

Thought: This is the current price of Apple's stock, not its market capitalization. The market cap is calculated by multiplying the share price by the number of outstanding shares. Thus, I need to find the total number of Apple's outstanding shares to calculate its market cap.  
Action: search["total number of Apple's outstanding shares"]

(My adversarial observation)

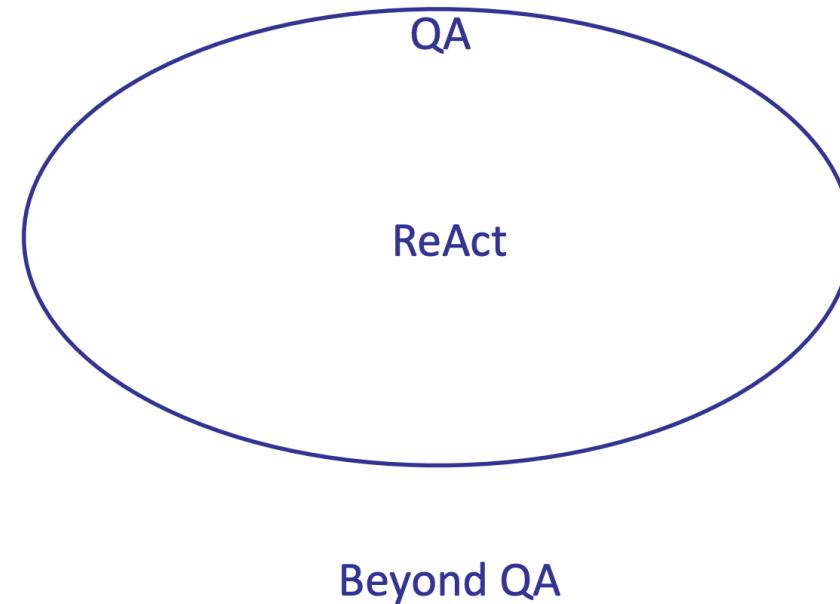
(My adversarial observation)

✓  
(Continues to solve the task)

# ReAct → Beyond QA?



- Now that we have used ReAct to address the issues in the QA task, how can we obtain a simple and unifying solution beyond QA tasks?



# Many Tasks Can Be Turned into Text Games



ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks.

```
You are in the middle of a room. Looking quickly around you, you see a drawer 2, a shelf 5, a drawer 1, a shelf 4, a sidetable 1, a drawer 5, a shelf 6, a shelf 1, a shelf 9, a cabinet 2, a sofa 1, a cabinet 1, a shelf 3, a cabinet 3, a drawer 3, a shelf 11, a shelf 2, a shelf 10, a dresser 1, a shelf 12, a garbagecan 1, a armchair 1, a cabinet 4, a shelf 7, a shelf 8, a safe 1, and a drawer 4.

Your task is to: put some vase in safe.

> go to shelf 6
You arrive at loc 4. On the shelf 6, you see a vase 2.

> take vase 2 from shelf 6
You pick up the vase 2 from the shelf 6.

> go to safe 1
You arrive at loc 3. The safe 1 is closed.

> open safe 1
You open the safe 1. The safe 1 is open. In it, you see a keychain 3.

> put vase 2 in/on safe 1
You won!
```

ALFWORLD: Aligning Text and Embodied Environments for Interactive Learning.

# Importance of Reasoning & Acting



## Acting without Reasoning

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.  
**Your task is to:** Put some pepper shaker on a drawer.

Act only

**Act 1:** Go to drawer 1  
**Obs 1:** The drawer 1 is closed.

**Act 2:** Open drawer 1  
**Obs 2:** You open the drawer 1. The drawer 1 is open. In it, you see a dishponge 2, and a spoon 1.

**Act 3:** Go to sinkbasin 1  
**Obs 3:** On the sinkbasin 1, you see a dishponge 3, a spatula 1, and a spoon 2.

**Act 4:** Take peppershaker 1 from sinkbasin 1  
**Obs 4:** Nothing happens.

**Act 5:** Take peppershaker 1 from sinkbasin 1  
**Obs 5:** Nothing happens.  
.....

X

Cannot explore systematically or incorporate feedback

## ReAct Enables Systematic Exploration

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.  
**Your task is to:** Put some pepper shaker on a drawer.

ReAct

**Act 1:** Think[First I need to find a peppe shaker...more likely to appear in cabinets (1-6), countertops (1-3), ... ]  
**Act 2:** Go to cabinet 1

**Obs 2:** On the cabinet 1, you see a vase 2.  
(Here the agent goes to cabinet 1, 2, 3, 4, 5, 6, then countertop 1 and 2)

**Act 9:** Go to countertop 3  
**Obs 9:** On the countertop 3, you see a apple 1, a bread 1, ..... a pepper shaker 1, and a vase 3.

**Act 10:** Take pepper shaker 1 from countertop 3  
**Obs 10:** You pick up the pepper shaker 1 from the countertop 3.

**Act 11:** Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]  
**Act 12:** Go to drawer 1

**Obs 12:** Drawer 1 is closed.

**Act 13:** Open drawer 1

**Obs 13:** You open Drawer 1 ...

**Act 14:** Put pepper shaker 1 in/on drawer 1

**Obs 14:** You put pepper shaker 1 in/on the drawer 1.

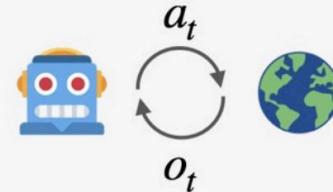
✓

# ReAct Is General and Effective



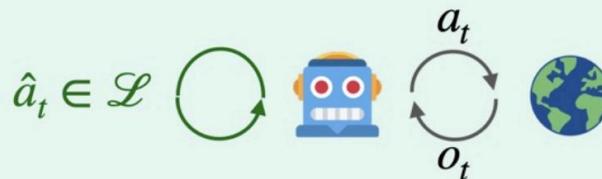
	(NLP tasks)		(RL tasks)
	HotpotQA (QA)	FEVER (fact check)	ALFWorld (Text game)
PaLM-540B			
Reason	29.4	56.3	N/A
Act	25.7	58.9	45
ReAct	<b>35.1</b>	<b>64.6</b>	<b>71</b>

**Traditional agents:** action space  $A$  defined by the environment



- **External feedback**  $o_t$
- Agent context  $c_t = (o_1, a_1, o_2, a_2, \dots, o_t)$
- Agent action  $a_t \sim \pi(a | c_t) \in A$

**ReAct:** action space  $\hat{A} = A \cup \mathcal{L}$  augmented by reasoning



- $\hat{a}_t \in \mathcal{L}$  can be any language sequence
- Agent context  $c_{t+1} = (c_t, \hat{a}_t, a_t, o_{t+1})$
- $\hat{a}_t \in \mathcal{L}$  only updates **internal context**

# Learning of LLM Agents



- Since agents require both acting and reasoning capabilities, the next step is to investigate how to further enhance their learning. This can be approached through three main paradigms:
  - In-Context Learning – Learning from few-shot exemplars
  - Supervised Finetuning – Learning From Experts
  - Reinforcement Learning – Learning from Environment

# In-context Learning



- Prompting LLM as Agent
  - Few-shot in-context learning: General guideline + two examples

You are an autonomous intelligent agent tasked with navigating a web browser. You will be given web-based tasks. These tasks will be accomplished through the use of specific actions you can issue.

*You can observe the following information:*

...

*You can do the following actions:*

...

...

# In-context Learning

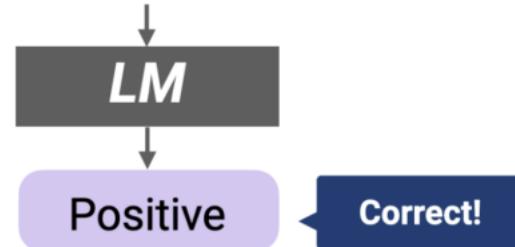
- LM performs a task just by conditioning on input-output examples, without optimizing any parameters.

Circulation revenue has increased by 5% in Finland. \n Positive

Panostaja did not disclose the purchase price. \n Neutral

Paying off the national debt will be extremely painful. \n Negative

The company anticipated its operating profit to improve. \n \_\_\_\_\_



example\_user

OBSERVATION:

[1744] link 'HP CB782A#ABA 640 Inkjet Fax Machine (Renewed)'  
[1749] StaticText '\$279.49'  
[1757] button 'Add to Cart'  
[1760] button 'Add to Wish List'  
[1761] button 'Add to Compare'

URL: <http://onestopmarket.com/office-products/office-electronics.html>

OBJECTIVE: What is the price of HP Inkjet Fax Machine

PREVIOUS ACTION: None

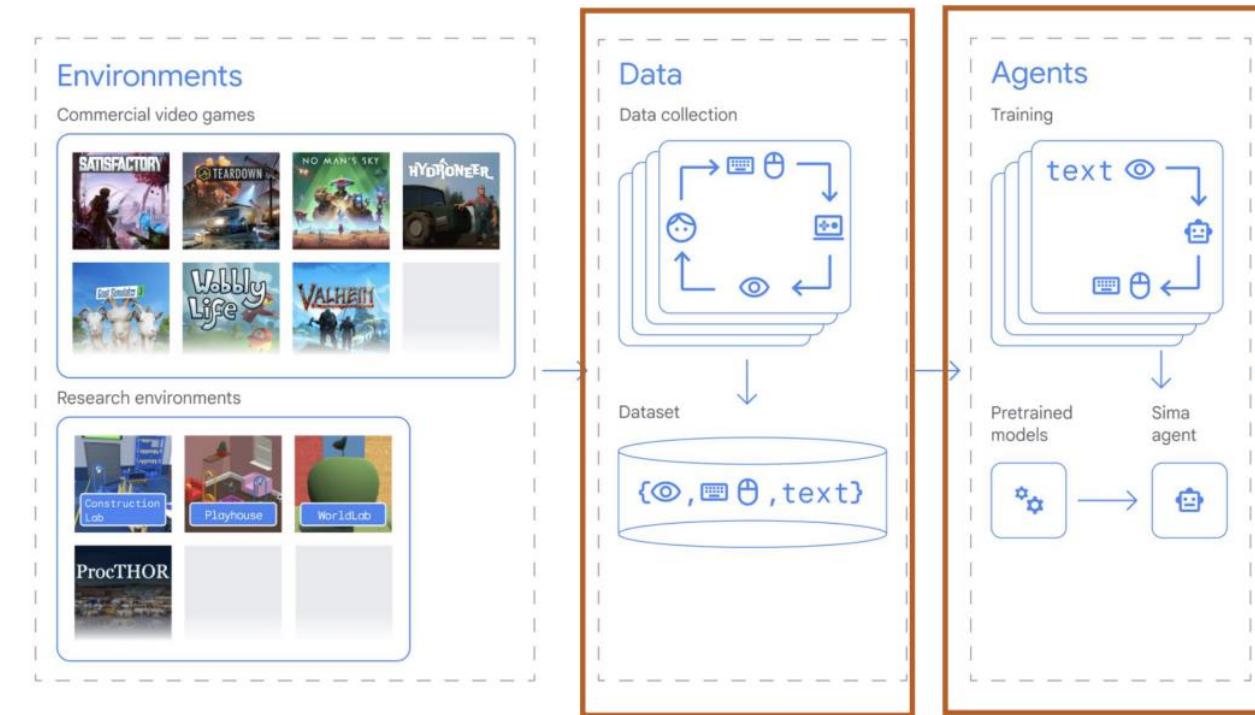
example\_assistant

Let's think step-by-step. This page lists the information of HP Inkjet Fax Machine, which is the product identified in the objective. Its price is \$279.49. I think I have achieved the objective. I will issue the stop action with the answer. In summary, the next action I will perform is ``stop [\$279.49]``

# Supervised Finetuning



- Collect large amount of expert trajectories (e.g. from human annotation or generated via models)  
task\_intent, [(obs\_1, action\_1), ..., (obs\_N, action\_N)]
- Finetune the LLM with standard cross-entropy loss.



# Supervised Finetuning



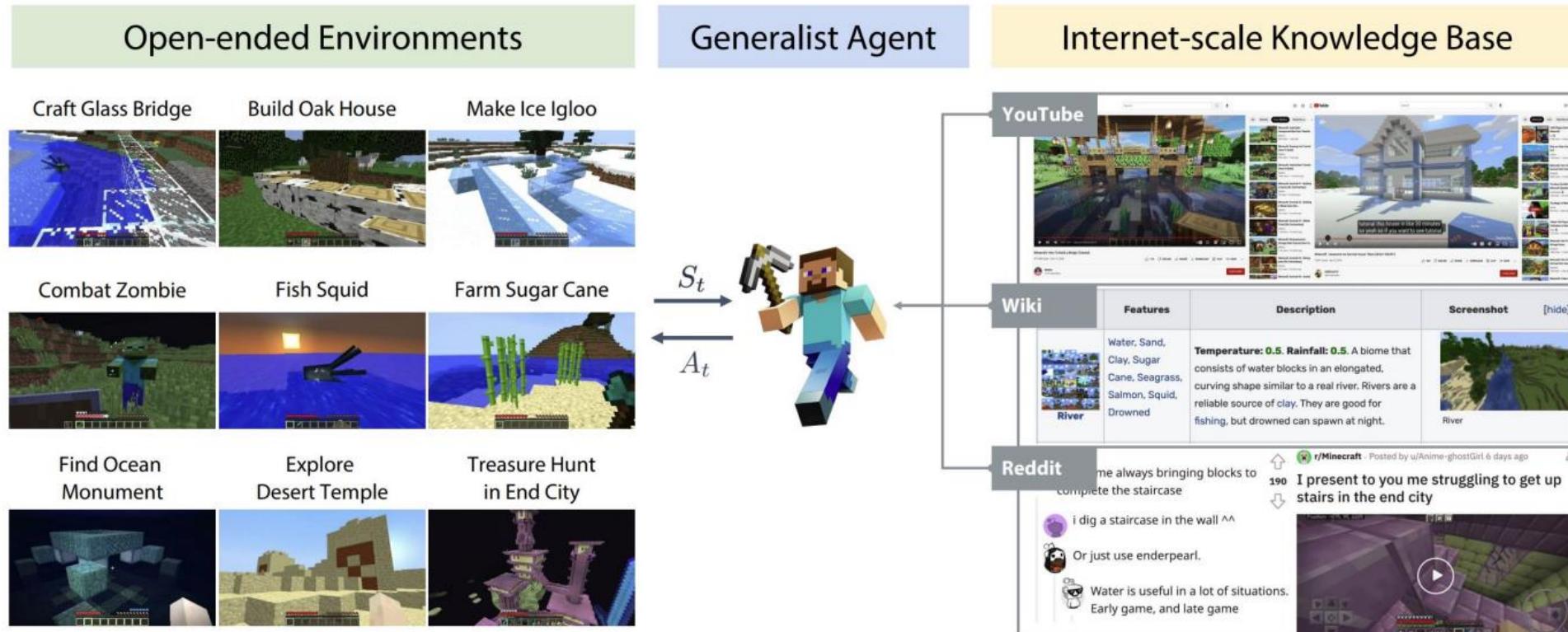
南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Data hungry
- Cannot learn much from failed trajectories
  - $a_1, a_2, a_3, \dots, a_{10}$  – Success
  - $a_1, a_2, a_3, \dots, a_{10}$  – Fail (Wasted)
- Need human trajectory?
  - Data augmentation techniques

# Supervised Finetuning



- Create More Training Data



# Supervised Finetuning



- Data Augmentation
  - Continue pre-train on large amount of data automatically mined
  - Even noisy, not clear trajectories, can provide domain adaptation.

YouTube



Wiki

Passive mobs

Allay	Axolotl	Bat	Cat	Chicken	Cod	Cow	Donkey	Fox	Frog
Glow Squid	Horse	Mushroom	Mule	Ocelot	Parrot	Pig	Pufferfish (defensive)	Rabbit	Salmon
Sheep	Skeleton Horse	Snow Golem	Squid	Strider	Tadpole	Tropical Fish	Turtle	Villager	Wandering Trader



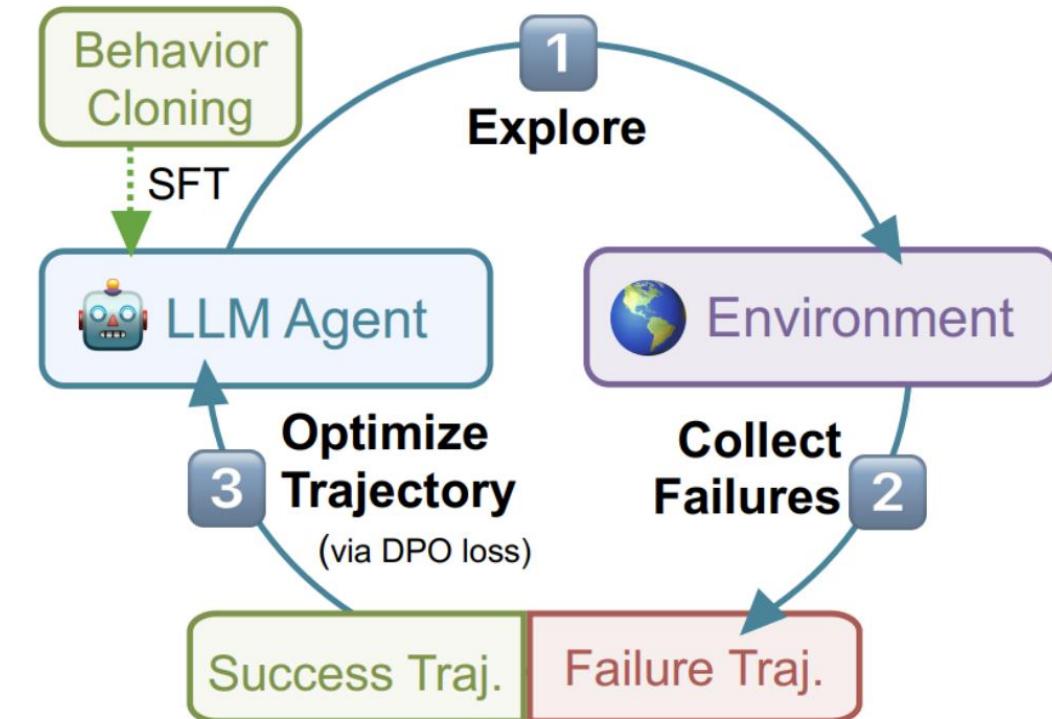
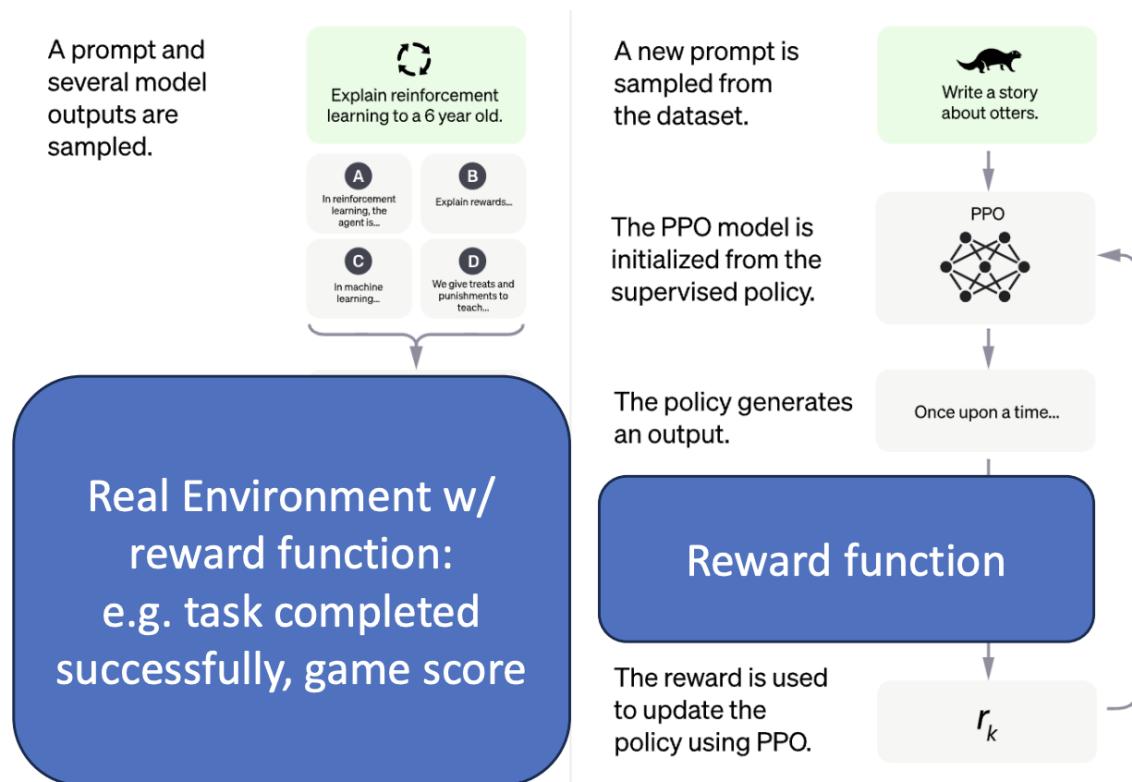
Reddit



# Reinforcement Learning



- Compared to RLHF:
  - Given environment, reward function (trajectory, reward) pairs without human



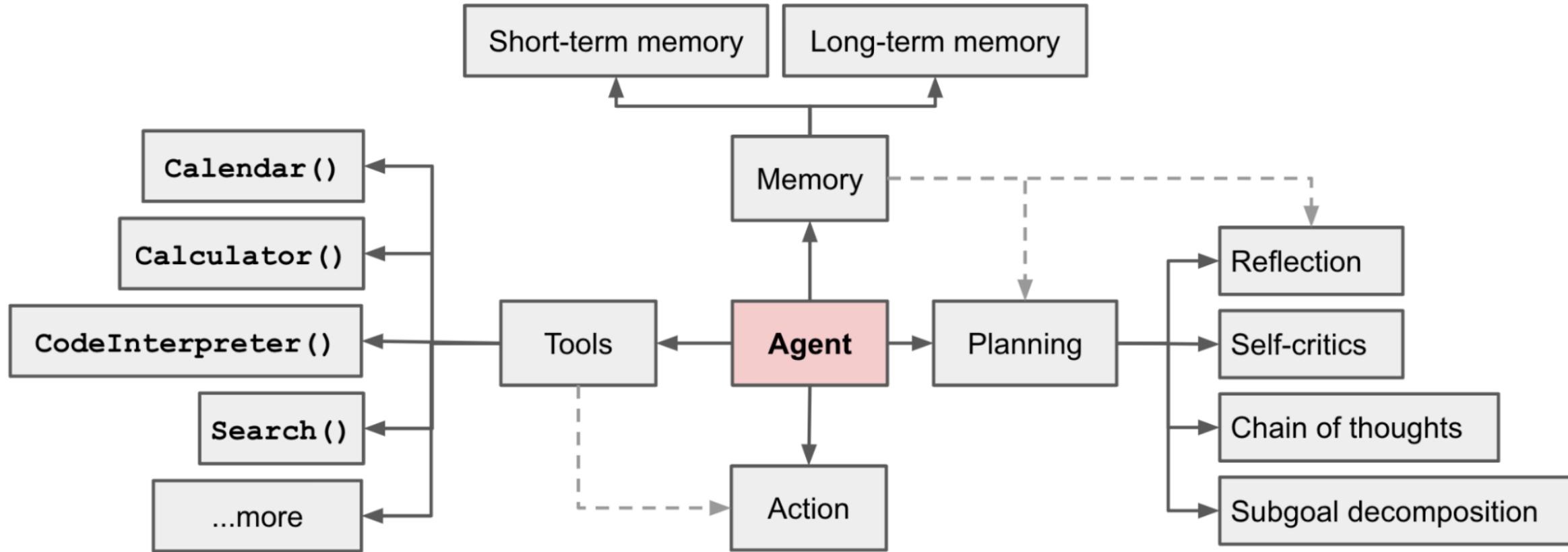
# Reinforcement Learning



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Closed loop, interactive environment
- Need good reward functions
  - What if the task success/fail is not easy to automatically assess?
- Need good initial models
  - Has decent basic knowledge ability, sparse rewards
- Scalability
  - The environment takes 10 seconds to env.step()
  - The reward function takes 100 seconds to get a scalar reward

# LLM-based Agent System



# Component One: Planning



- What does the agent need to do to accomplish a specified goal?
  - **High-level plan:** Identify subgoals for a long-horizon task
  - **Low-level plan:** Identify sequence of actions
- Traditionally use symbolic reasoning
  - Hard to recover from errors
  - Difficult to convert expert knowledge into planning languages such as PDDL (Planning Domain Definition Language)

```
(define (domain blocksworld)
  (:requirements :typing :fluents :negative-preconditions)
  (:types block)
  (:predicates
    (on ?a ?b - block)
    (clear ?a - block)
    (holding ?a - block)
    (handempty)
    (ontable ?x - block)
  )
  (:action pickup
    :parameters (?x - block)
    :precondition (and (ontable ?x) (handempty) (clear ?x))
    :effect (and (holding ?x) (not (handempty)) (not (clear ?x)) (not (ontable ?x)))
  )
  (:action unstack
    :parameters (?x ?y - block)
    :precondition (and (on ?x ?y) (handempty) (clear ?x))
    :effect (and (holding ?x) (not (handempty)) (not (clear ?x)) (clear ?y) (not (on ?x ?y)))
  )
)
```

# Component One: Planning



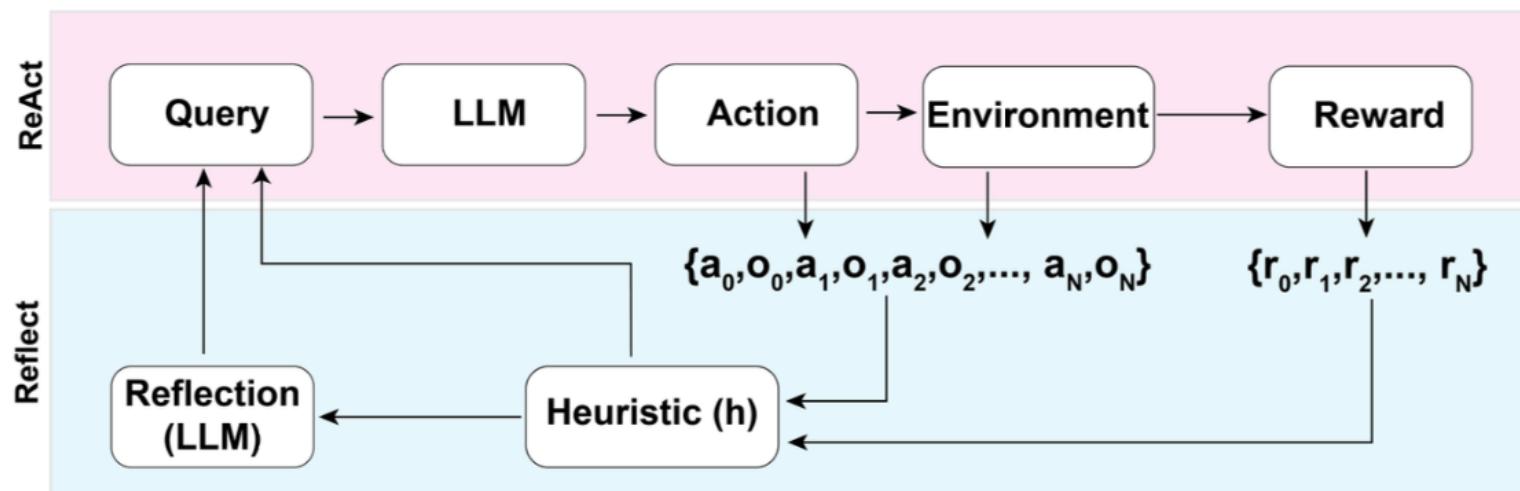
- Taxonomy for planning with LLMs
  - **Task decomposition** - figure out subgoals, do planning for subgoals if needed
  - **Multi-plan selection** - generate multiple plans and then select one
  - **External planner** - LLM used to formalized the problem which is passed to an external planner
  - **Reflection and refinement** - After obtaining a plan, the LLM future reflects on the plan and refine it to fix any issues with the original plan
  - **Memory-augmented planning** - Uses external memory to retrieve information and then determines plan based on that



# Component One: Planning



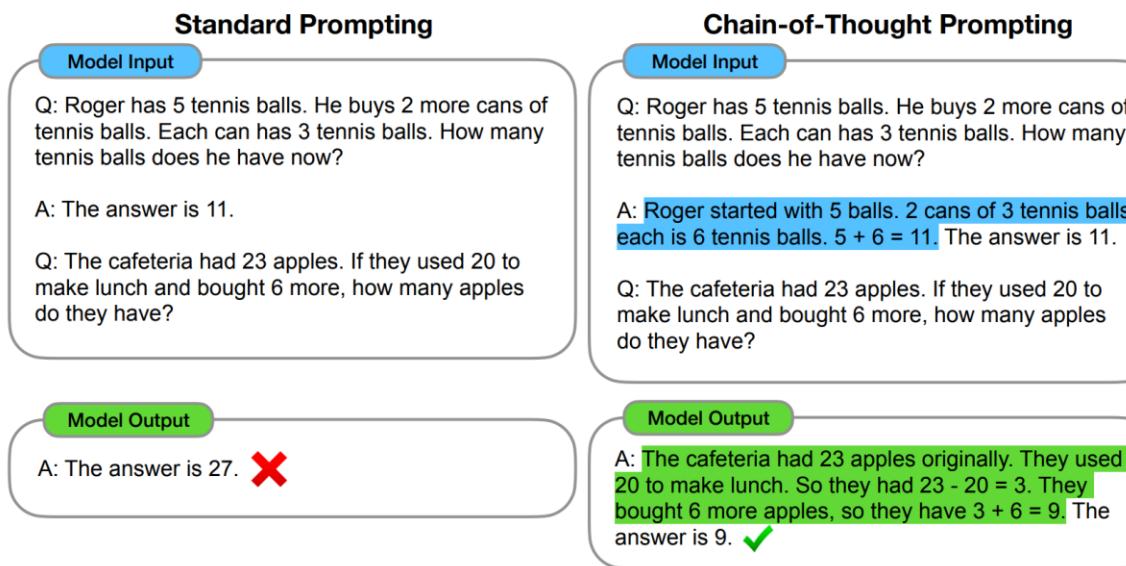
- Self-Reflection
  - Improve iteratively by refining past action decisions and correcting previous mistakes.
- ReAct: interleave thought, action, and observation
  - Prompting to combine reasoning with actions
  - Comparison with other ways of prompting



# Component One: Planning

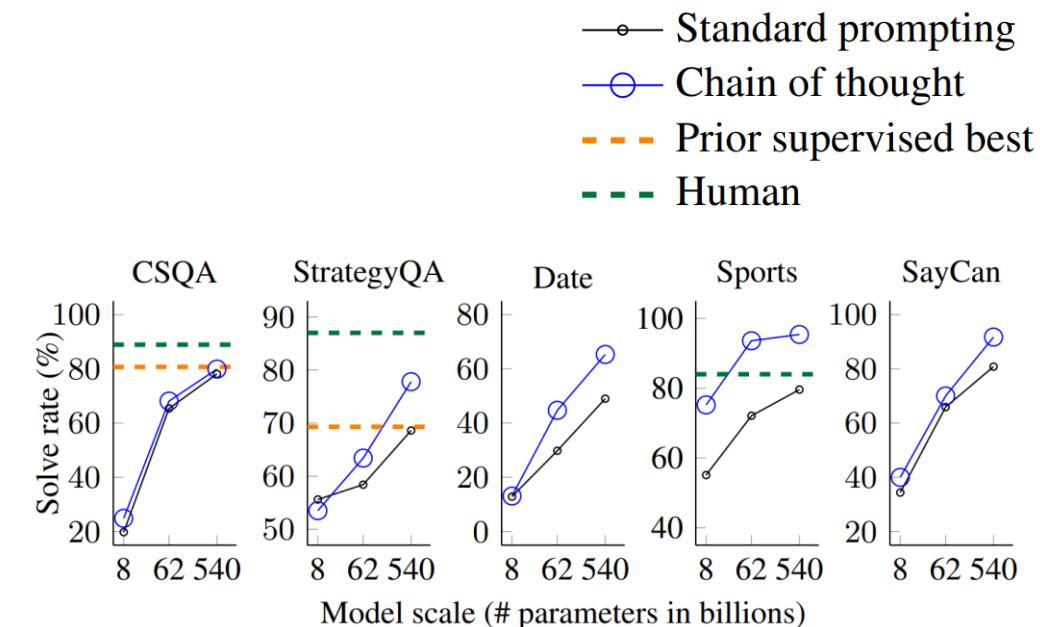


- Chain-of-thought (CoT) :
  - Elicits the reasoning capacities of large language models by guiding LLMs to generate rationales before outputting the answer.
- A few demonstrations are enough, as humans: 100x-1000x data efficient than prior supervised SOTA



<input, output>

<input, rationale, output>

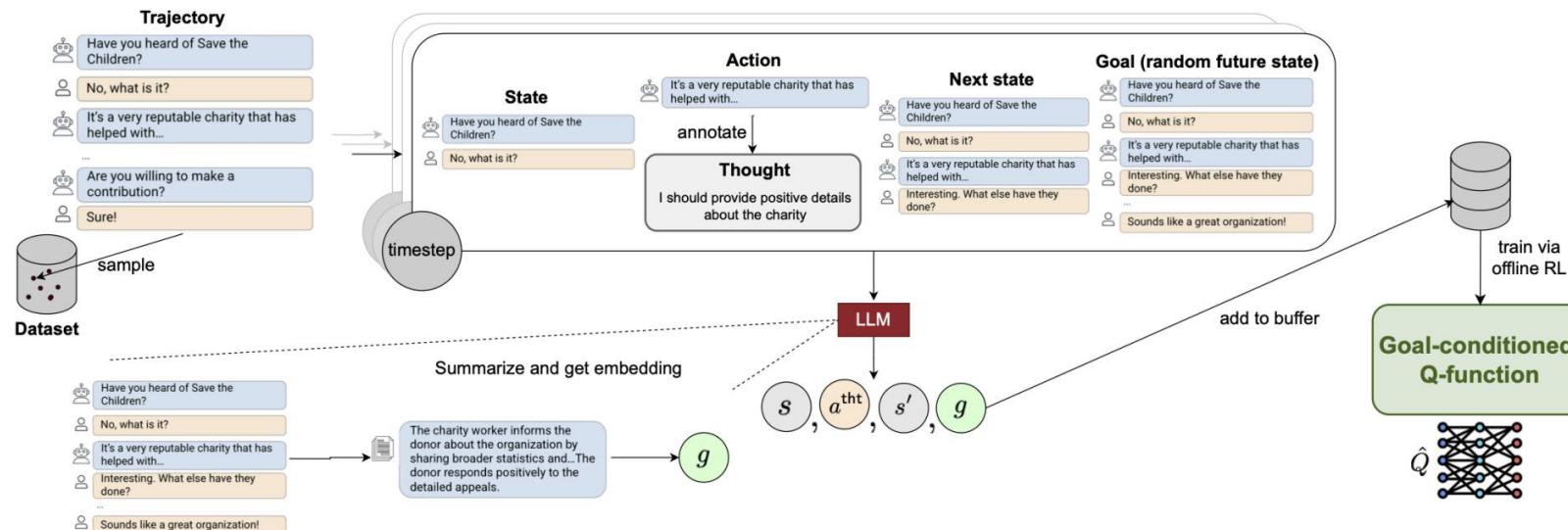


# Component One: Planning



- Beyond prompting-based methods, RL as an External Guide for Planning:
  - It is not used to directly fine-tune the generative capabilities of the LLM, but rather to train an auxiliary value or heuristic function.
  - Planning without Search leveraging offline goal-conditioned RL to learn a language-based value critic that guides LLM reasoning and planning without updating the LLM parameters.

During offline training:



[2505.18098] Planning without Search: Refining Frontier LLMs with Offline Goal-Conditioned RL

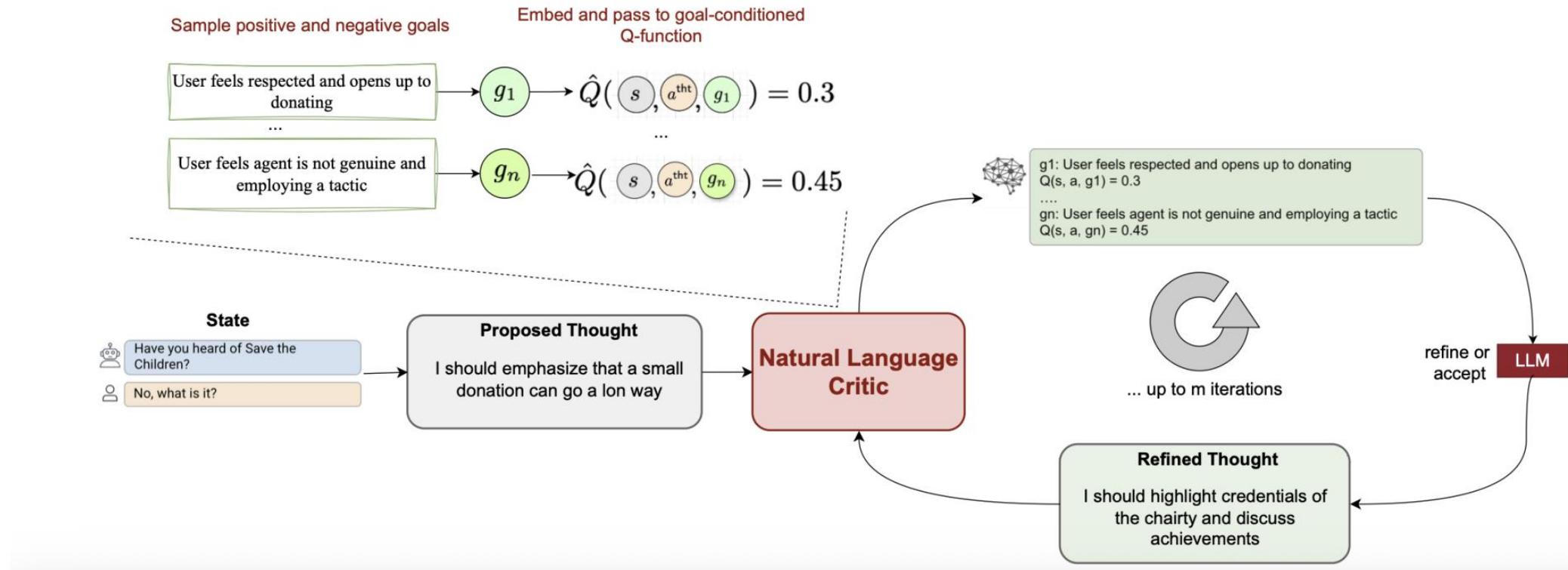
# Component One: Planning



- RL as an External Guide for Planning:

- Planning without Search

**During inference:**



# Component Two: Memory



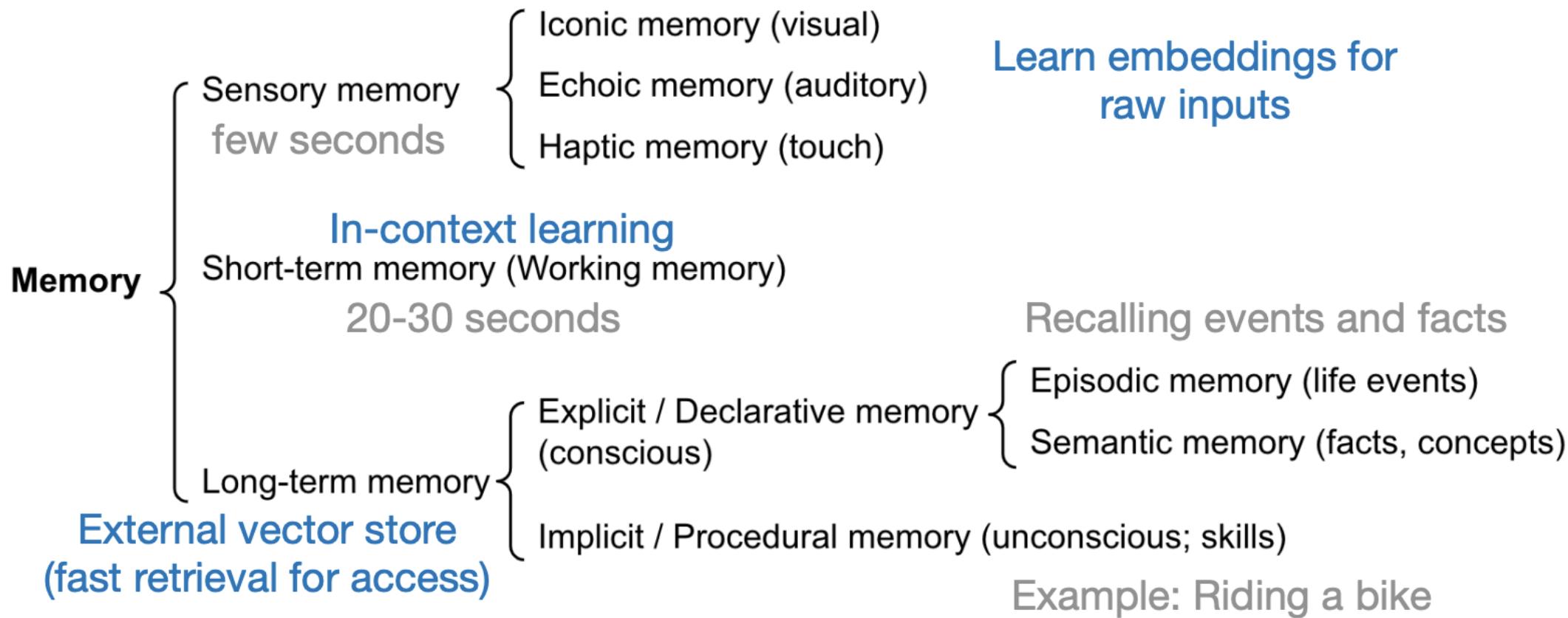
Memory can be defined as the processes used to acquire, store, retain, and later retrieve information. There are several types of memory in human brains.

- Sensory Memory
  - Sensory memory as learning embedding representations for raw inputs, including text, image or other modalities.
- Short-Term Memory
  - Short-term memory as in-context learning. It is short and finite, as it is restricted by the finite context window length of Transformer.
- Long-Term Memory
  - Long-term memory as the external vector store that the agent can attend to at query time, accessible via fast retrieval.

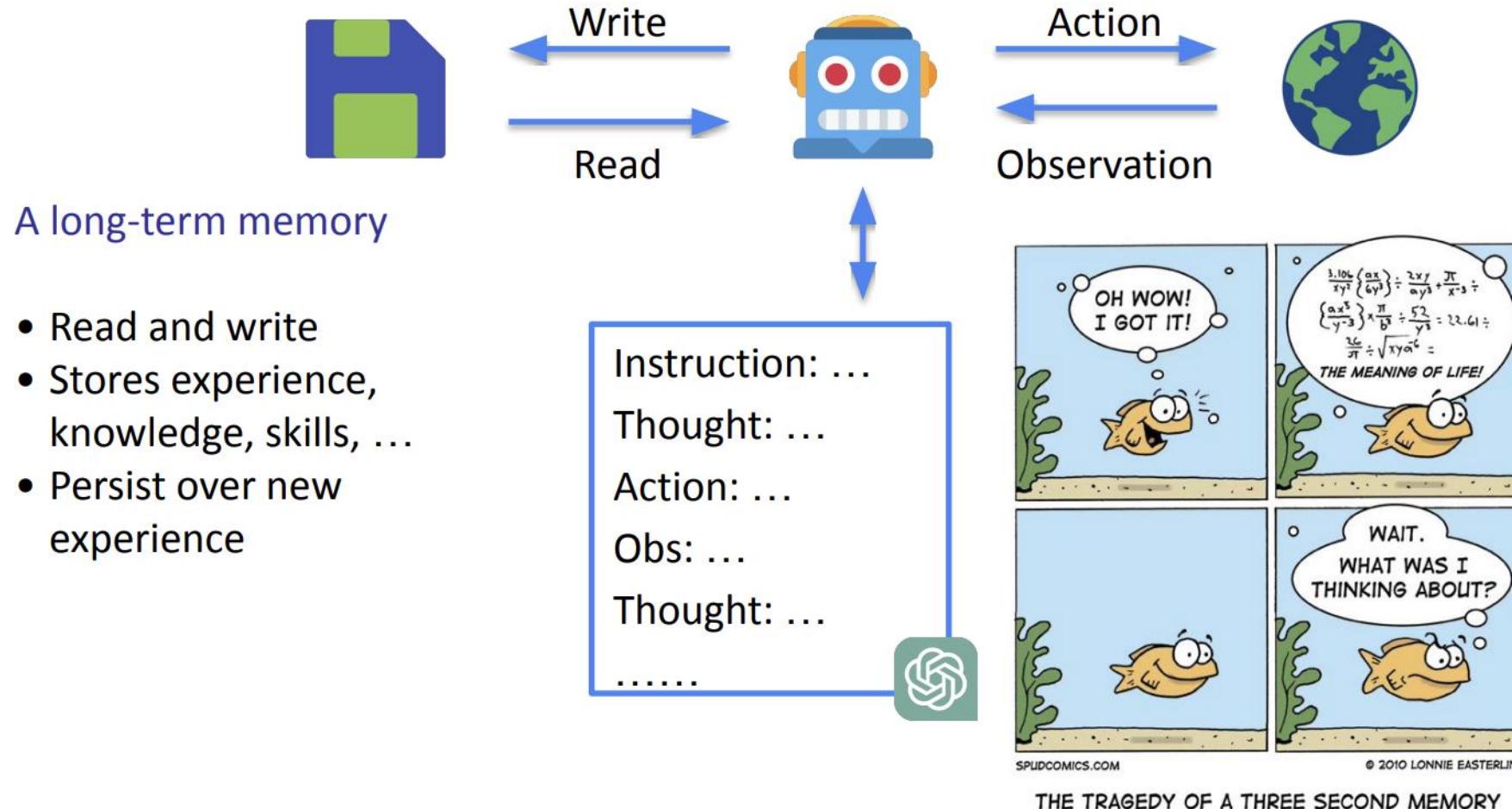
# Component Two: Memory



- Types of memory



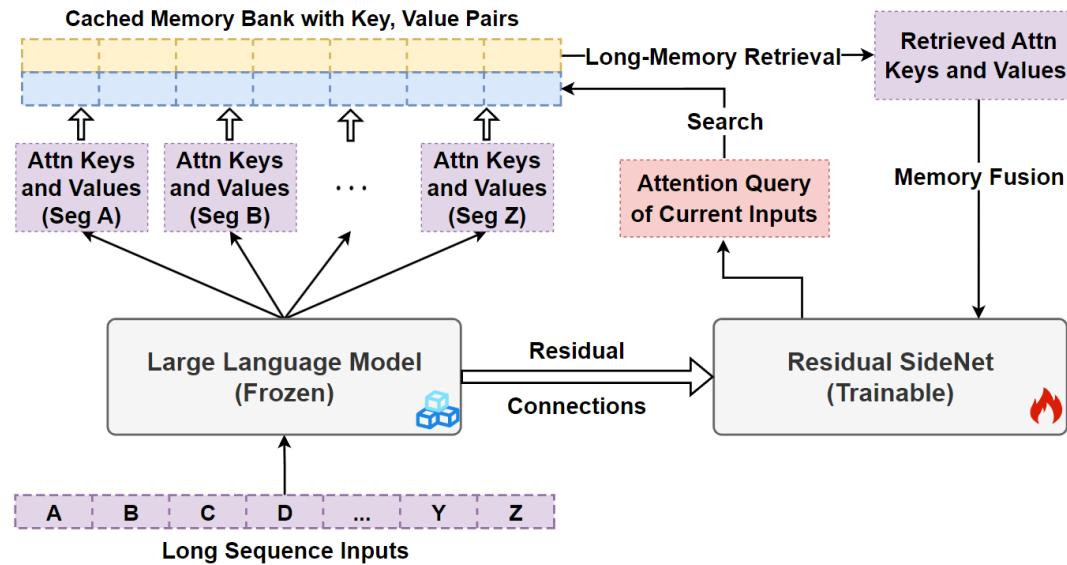
# Component Two: Memory



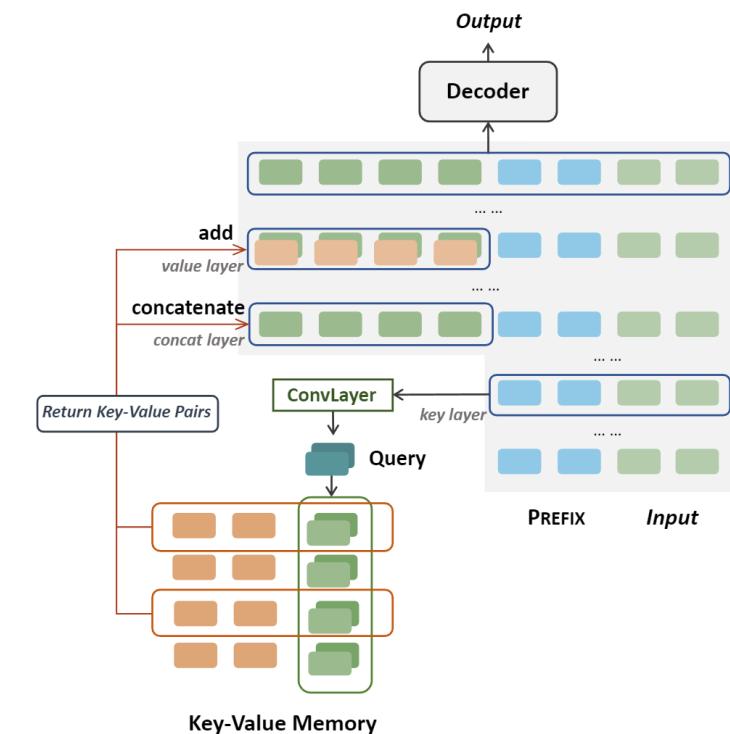
# Component Two: Memory

Compressing long-term memories with vectors

- LongMem caches long-form previous context into the non-differentiable memory bank. For future inputs, the top-k key-value pairs of long-term memory are retrieved and fused into language modeling.
- EMAT stores knowledge into a key-value memory and learns an implicit strategy to integrate multiple memory slots into the transformer.



Overview of LongMem architecture



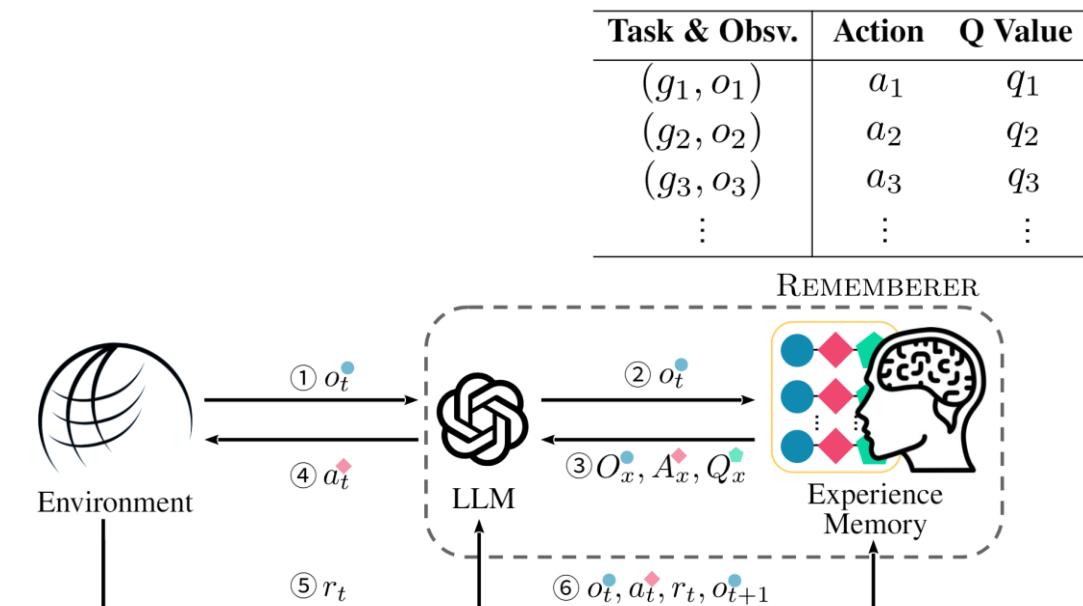
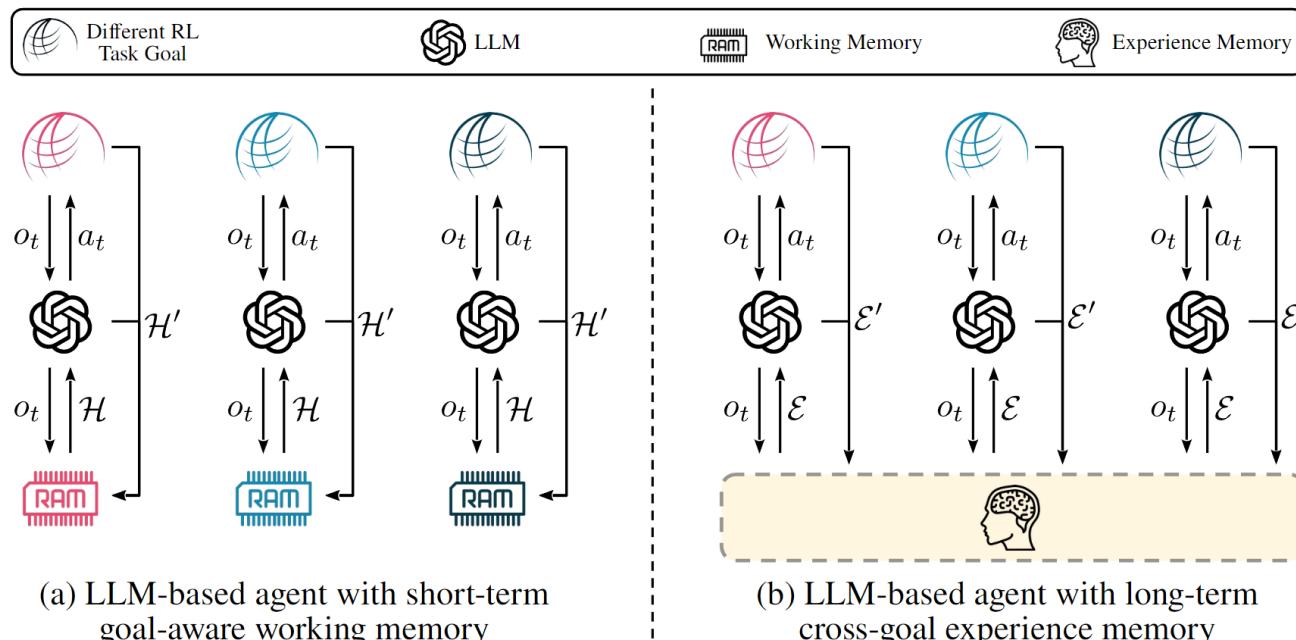
Architecture of Efficient Key-Value Memory Augmented Transformers (EMAT)

# Component Two: Memory



Storing long-term experience memories as a table

- Equipped with a long-term experience memory, REMEMBERER is capable of exploiting the experiences from the past episodes and from different related tasks.
- The experience memory is designed as a table storing the task information, observation, action, and the corresponding Q value estimation.

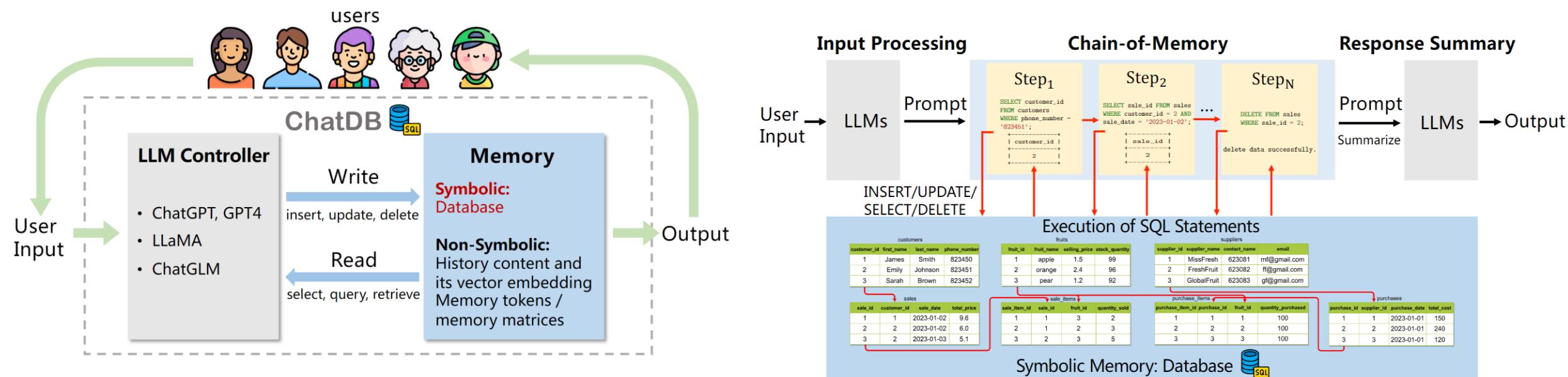


Reinforcement Learning with Experience Memory (RLEM) and architecture of REMEMBERER

# Component Two: Memory



- Long-term memories can be SQL databases
- Inspired from modern computer architectures, [ChatDB](#) augment LLMs with symbolic memory for complex multi-hop reasoning.
- Such a symbolic memory framework is instantiated as an LLM and a set of SQL databases, where the LLM generates SQL instructions to manipulate the SQL databases.

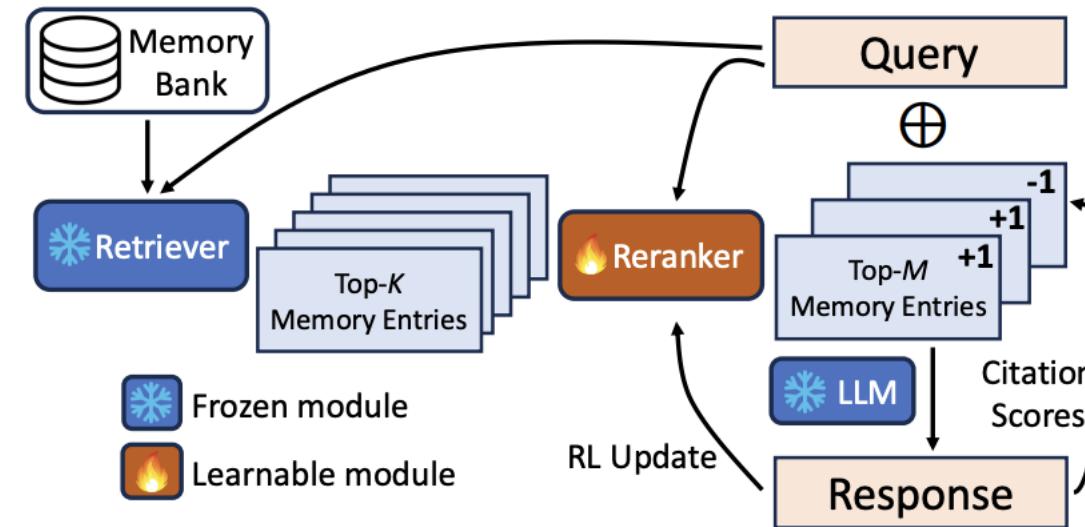
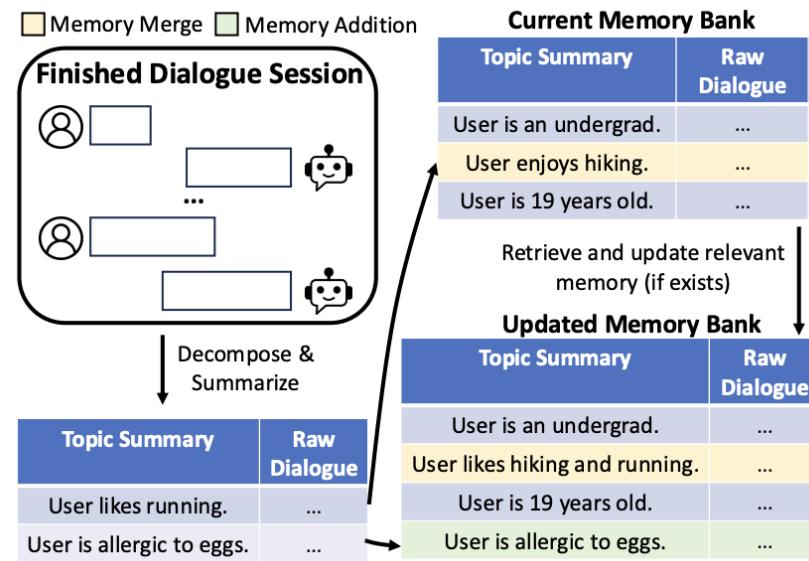


# Component Two: Memory



Refining memory retrieval with self-generated feedback

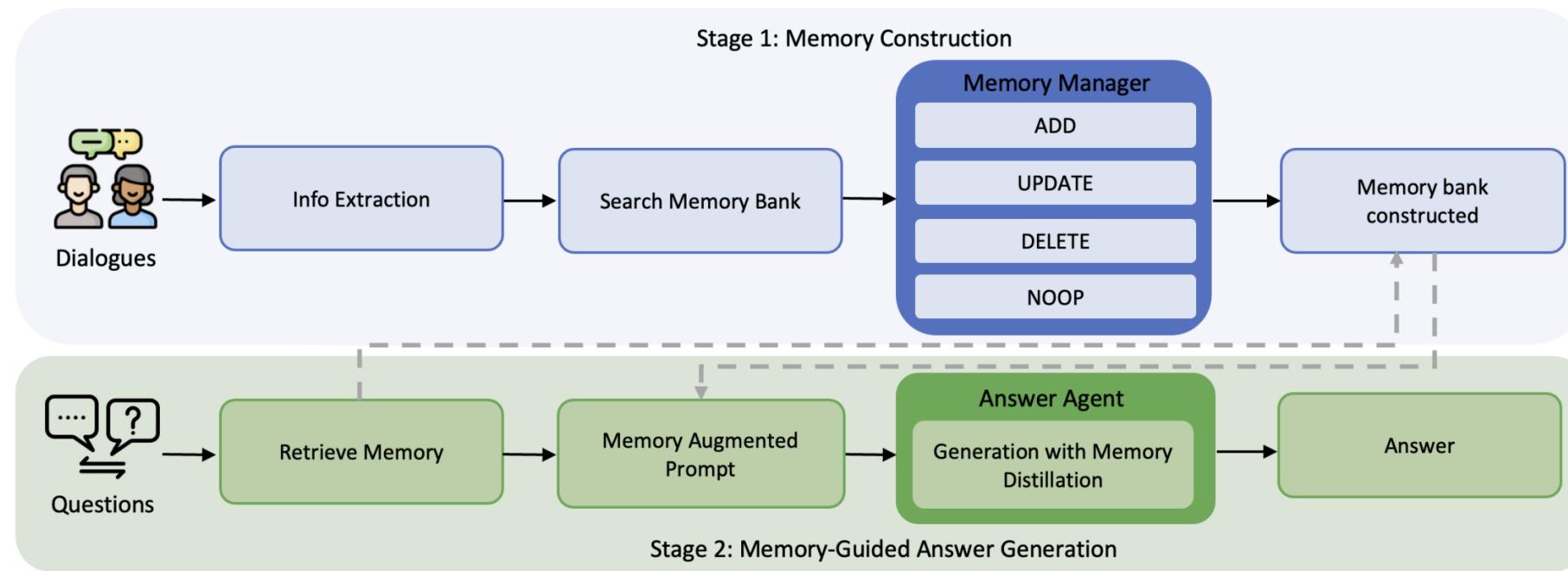
- Two challenges in long-term interactions:
  - 1) the granularity of memory storage was too rigid
  - 2) the memory retrieval mechanism was fixed and could not adapt to dynamic dialogue scenarios.
- The [RMM](#) framework optimizes memory storage quality through prospective reflection and enhances retrieval accuracy through retrospective reflection. Form a dynamic and adaptive long-term memory management system.



# Component Two: Memory

Automatic management of long-term memory through reinforcement learning

- [Memory-R1](#) introduces a RL-based memory-augmented Agent framework where a Memory Manager learns to perform structured operations (ADD/UPDATE/DELETE/NOOP) via PPO or GRPO based on downstream QA performance, while an Answer Agent employs a Memory Distillation policy over RAG-retrieved entries to reason and answer.

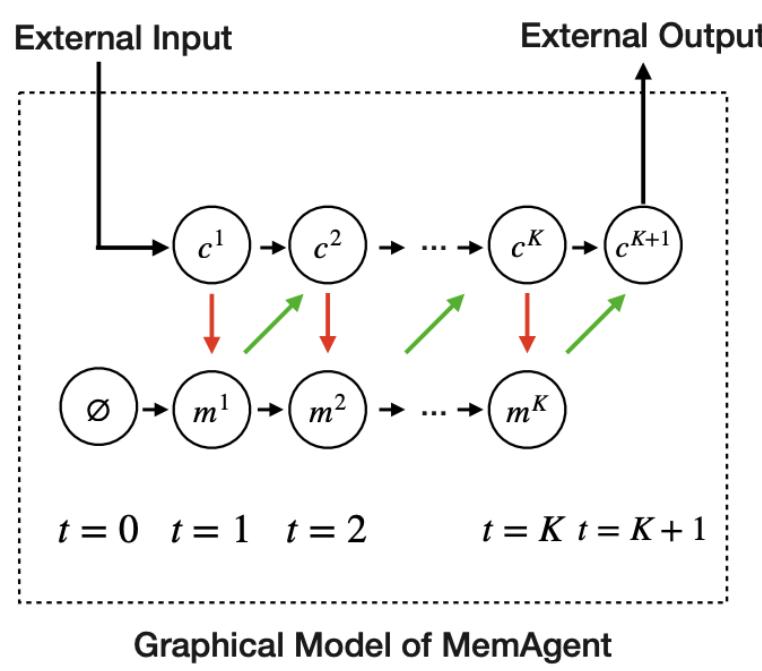
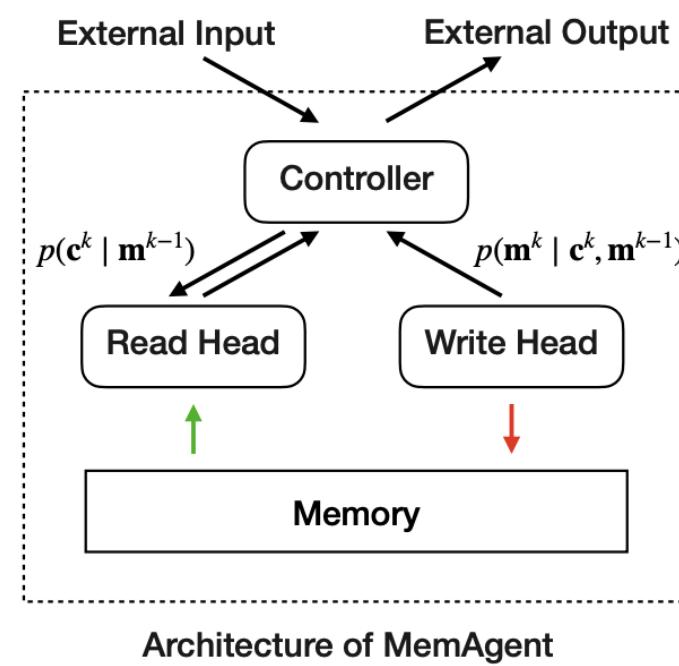


# Component Two: Memory



Compress long-term memory into token form for storage

- (I) explicit tokens (human-readable natural language): [MemAgent](#) proposes that the agent maintain a natural-language memory pool and use a reinforcement learning policy to decide, for each segment, which tokens to retain or overwrite, effectively compressing long-context inputs into concise and informative summaries.

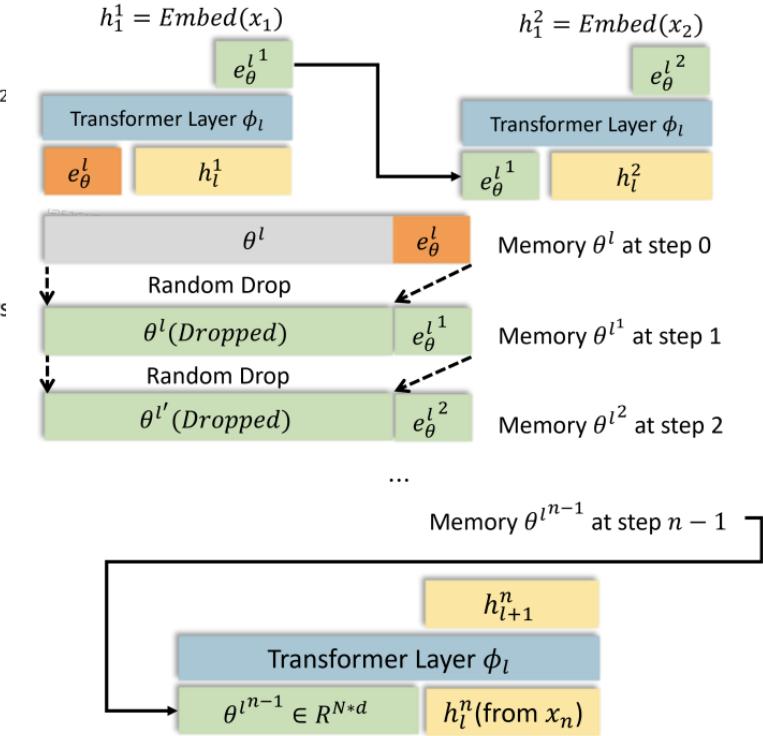
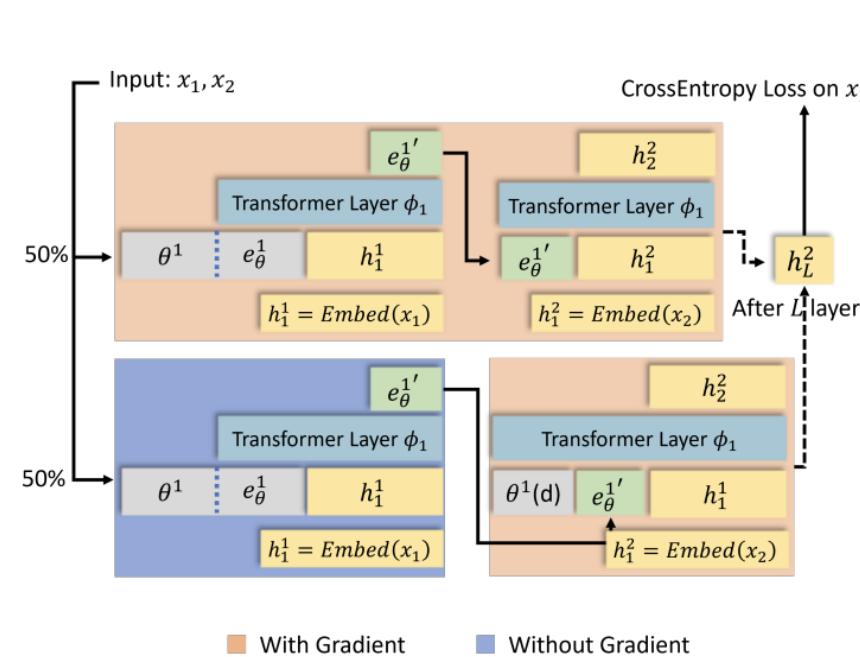
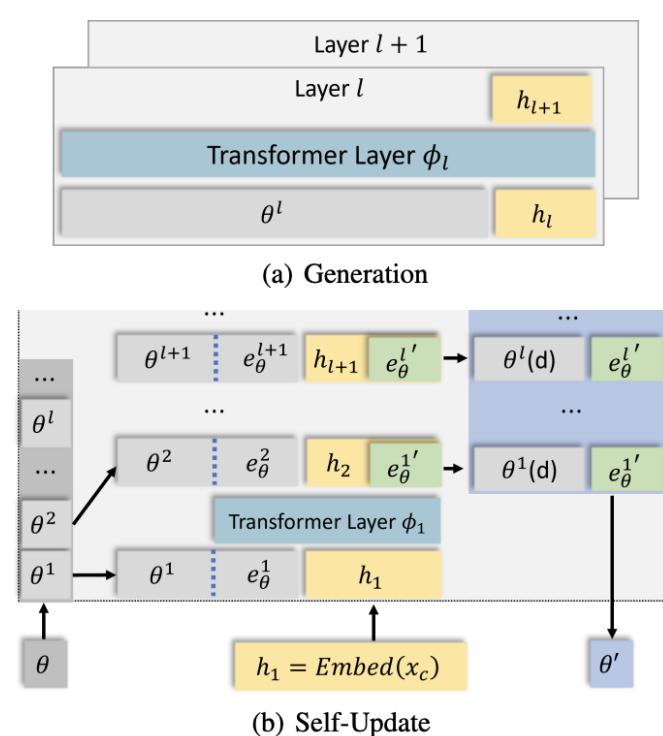


# Component Two: Memory



Compress long-term memory into token form for storage

- (2) In [MemoryLLM](#), a fixed set of latent tokens serves as “memory tokens,” which are repeatedly retrieved, integrated into the LLM’s forward computation, and updated as the context evolves

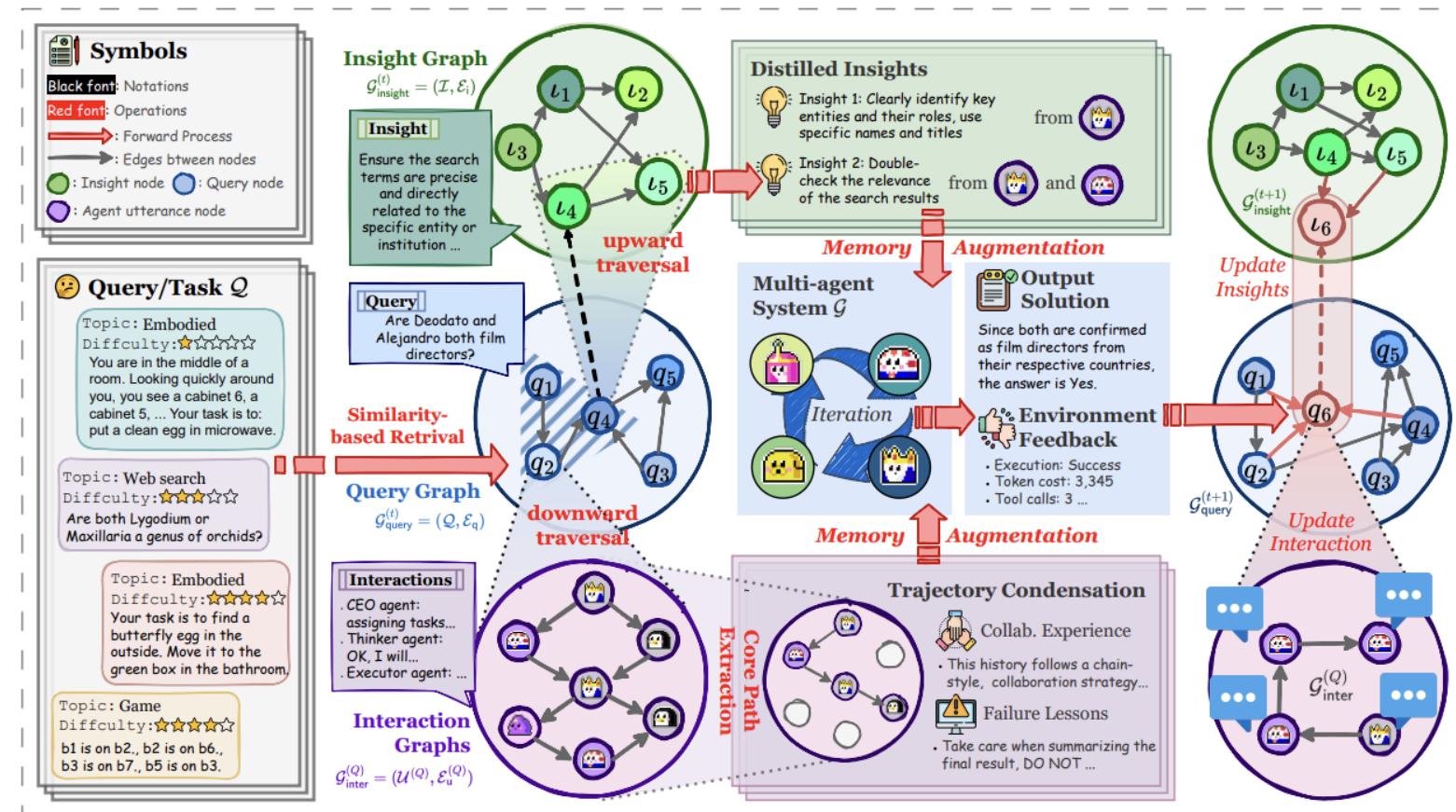


# Component Two: Memory



Compress long-term memory into a graph structure for management.

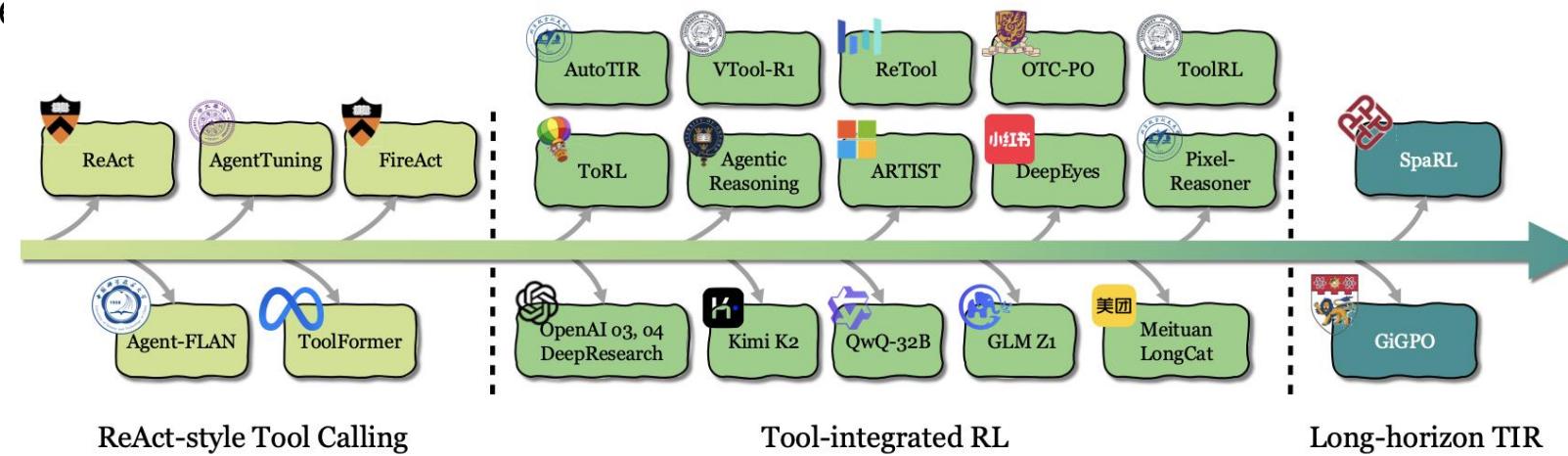
- G-Memory is a hierarchical agentic memory system that manages long-term multi-agent interactions through a three-tier graph structure, enabling cross-trial knowledge utilization and compression of collaboration trajectories, thereby enhancing MAS self-evolution and task performance.



# Component Three: Tool Use



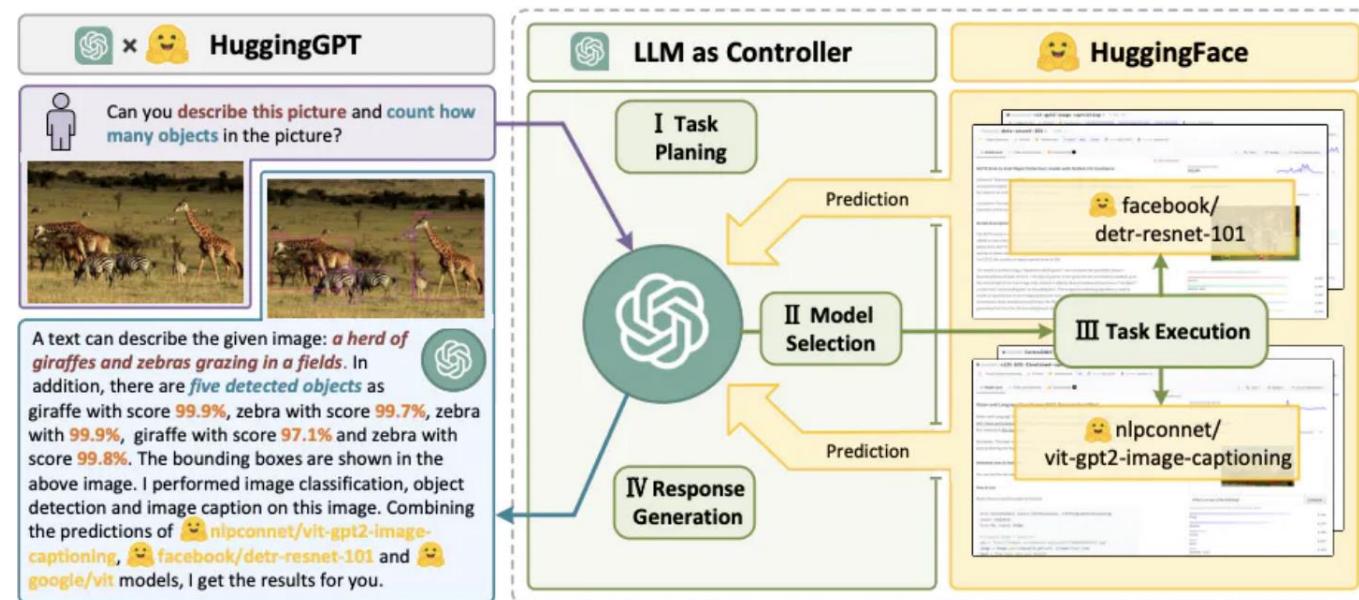
- Tool use is a remarkable and distinguishing characteristic of human beings. We create, modify and utilize external objects to do things that go beyond our physical and cognitive limits. Equipping LLMs with external tools can significantly extend the model capabilities.
  - API calls to external services (math calculator, currency converter, etc)
  - Expert models that can be called
  - Executing programming or scripting tasks
  - Accessing knowledge
  - Multimodal tools
  - .....



# Component Three: Tool Use



- ReAct-style Tool Calling
- (1) prompt engineering approach:
  - Exemplified by [ReAct](#), leveraged few-shot exemplars to guide an LLM to interleave reasoning traces and actions within a "Thought-Action-Observation" cycle, capitalizing on the model's in-context learning abilities.
  - [HuggingGPT](#): Task decomposition with model selection
    - Task planning
    - Model selection
    - Task execution
    - Response generation



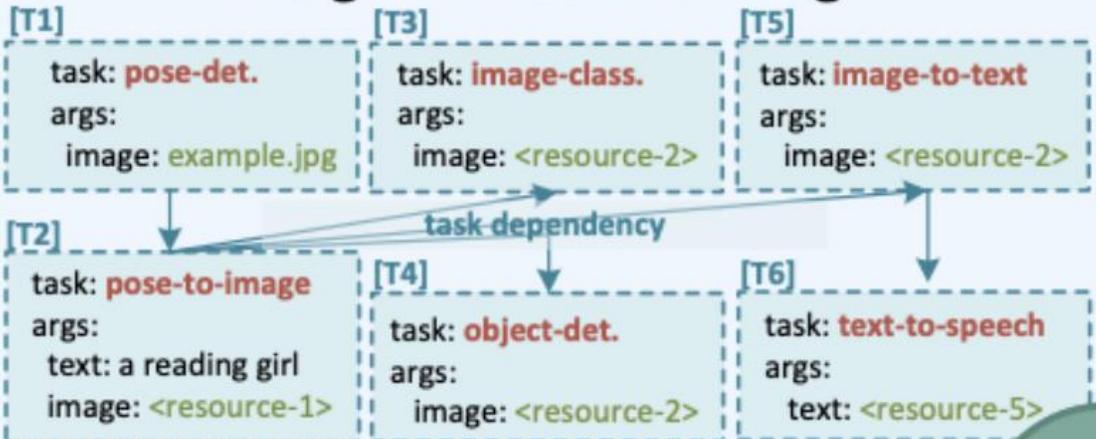
# Component Three: Tool Use



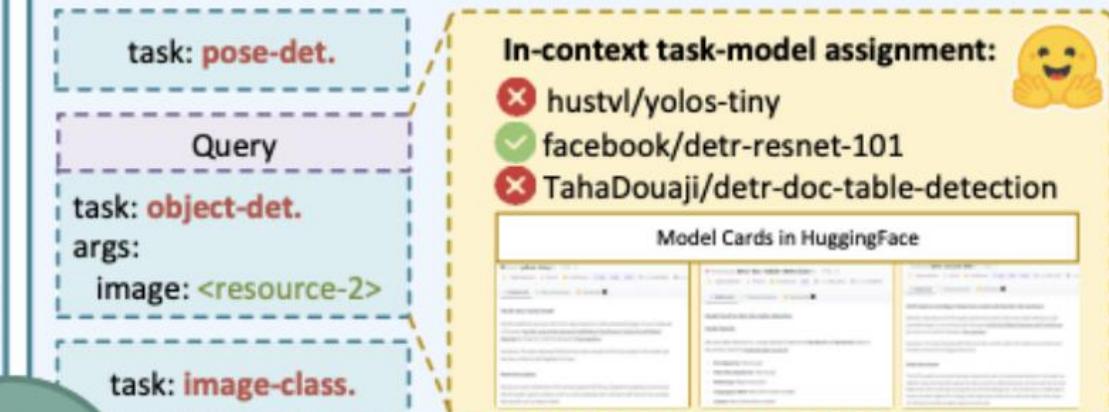
Please generate an image where a girl is reading a book, and her pose is the same as the boy in the image example.jpg, then please describe the new image with your voice.

## Request

### Stage #1: Task Planning



### Stage #2: Model Selection



# Component Three: Tool Use



- Task planning: figure out what task we want to solve, its id, dependencies, and arguments that are needed.

	Prompt
Task Planning	#1 Task Planning Stage - The AI assistant performs task parsing on user input, generating a list of tasks with the following format: <code>[{"task": task, "id": task_id, "dep": dependency_task_ids, "args": {"text": text, "image": URL, "audio": URL, "video": URL}}]</code> . The "dep" field denotes the id of the previous task which generates a new resource upon which the current task relies. The tag " <code>&lt;resource&gt;-task_id</code> " represents the generated text, image, audio, or video from the dependency task with the corresponding task_id. The task must be selected from the following options: <code>Available Task List</code> . Please note that there exists a logical connections and order between the tasks. In case the user input cannot be parsed, an empty JSON response should be provided. Here are several cases for your reference: <code>Demonstrations</code> . To assist with task planning, the chat history is available as <code>Chat Logs</code> , where you can trace the user-mentioned resources and incorporate them into the task planning stage.
	<b>Demonstrations</b>
	Can you tell me how many objects in e1.jpg? <code>[{"task": "object-detection", "id": 0, "dep": [-1], "args": {"image": "e1.jpg"}}]</code>
	In e2.jpg, what's the animal and what's it doing? <code>[{"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "e2.jpg"}}, {"task": "image-cls", "id": 1, "dep": [-1], "args": {"image": "e2.jpg"}}, {"task": "object-detection", "id": 2, "dep": [-1], "args": {"image": "e2.jpg"}}, {"task": "visual-question-answering", "id": 3, "dep": [-1], "args": {"text": "what's the animal doing?", "image": "e2.jpg"}}]</code>
	First generate a HED image of e3.jpg, then based on the HED image and a text "a girl reading a book", create a new image as a response. <code>[{"task": "pose-detection", "id": 0, "dep": [-1], "args": {"image": "e3.jpg"}}, {"task": "pose-text-to-image", "id": 1, "dep": [0], "args": {"text": "a girl reading a book", "image": "&lt;resource&gt;-0"}}]</code>

# Component Three: Tool Use



- Model selection: figure out what model to invoke

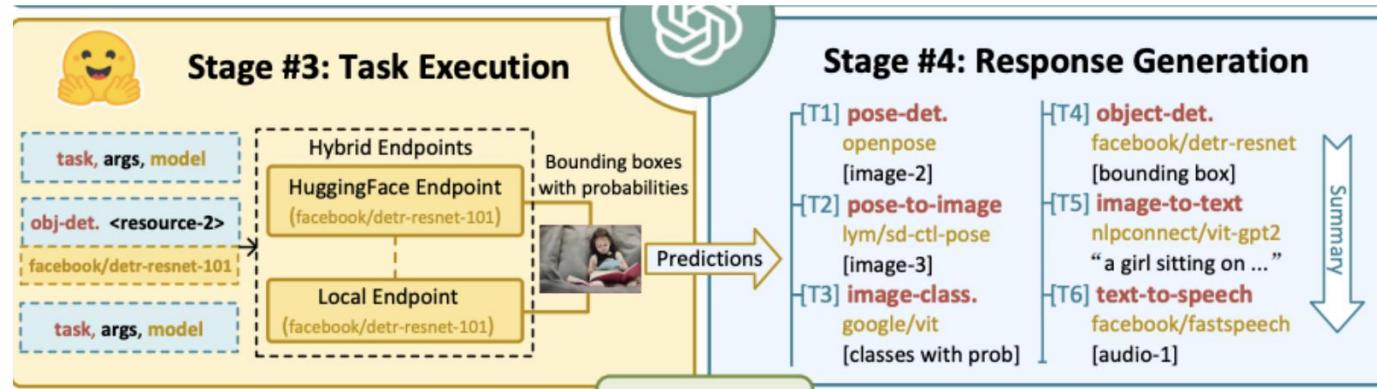
Model Selection	Prompt
	#2 Model Selection Stage - Given the user request and the call command, the AI assistant helps the user to select a suitable model from a list of models to process the user request. The AI assistant merely outputs the model id of the most appropriate model. The output must be in a strict JSON format: <code>{"id": "id", "reason": "your detail reason for the choice"}</code> . We have a list of models for you to choose from <code>{ Candidate Models }</code> . Please select one model from the list.
	Candidate Models
	<code>{"model_id": model id #1, "metadata": meta-info #1, "description": description of model #1}</code> <code>{"model_id": model id #2, "metadata": meta-info #2, "description": description of model #2}</code> ... <code>{"model_id": model id #K, "metadata": meta-info #K, "description": description of model #K}</code>

# Component Three: Tool Use



Please generate an image where a girl is reading a book, and her pose is the same as the boy in the image example.jpg, then please describe the new image with your voice.

Request



The image you gave me is of "boy". The first thing I did was use 🤖 OpenCV's openpose model to analyze the pose of the boy in the image. I fed that image, specifically located at [Image-1], into the model to get an output image of the pose, located at [Image-2]. Then I take that pose and generate a new image, located at [Image-3], using 🤖 llyasviel/sd-controlnet-openpose and the instruction "a girl is reading". Next, I ran object detection on the image generated in the previous step, to get predictions of the objects present in the image, including the box location. These predictions were run through the 🤖 facebook/detr-resnet-101 model. The box was drawn on the image, which is located at [Image-4]. Then, I used the predictions of a image classification model 🤖 google/vit-base-patch16-224 and a image caption model a 🤖 nlpconnect/vit-gpt2-image-captionin model to generate the caption for newly generated image. It generated text: "a girl sitting on a bed reading a book". Lastly, I ran the 🤖 facebook/fastspeech2-en-ljspeech model to convert the generated text into audio, located at [Audio-1].

# Component Three: Tool Use



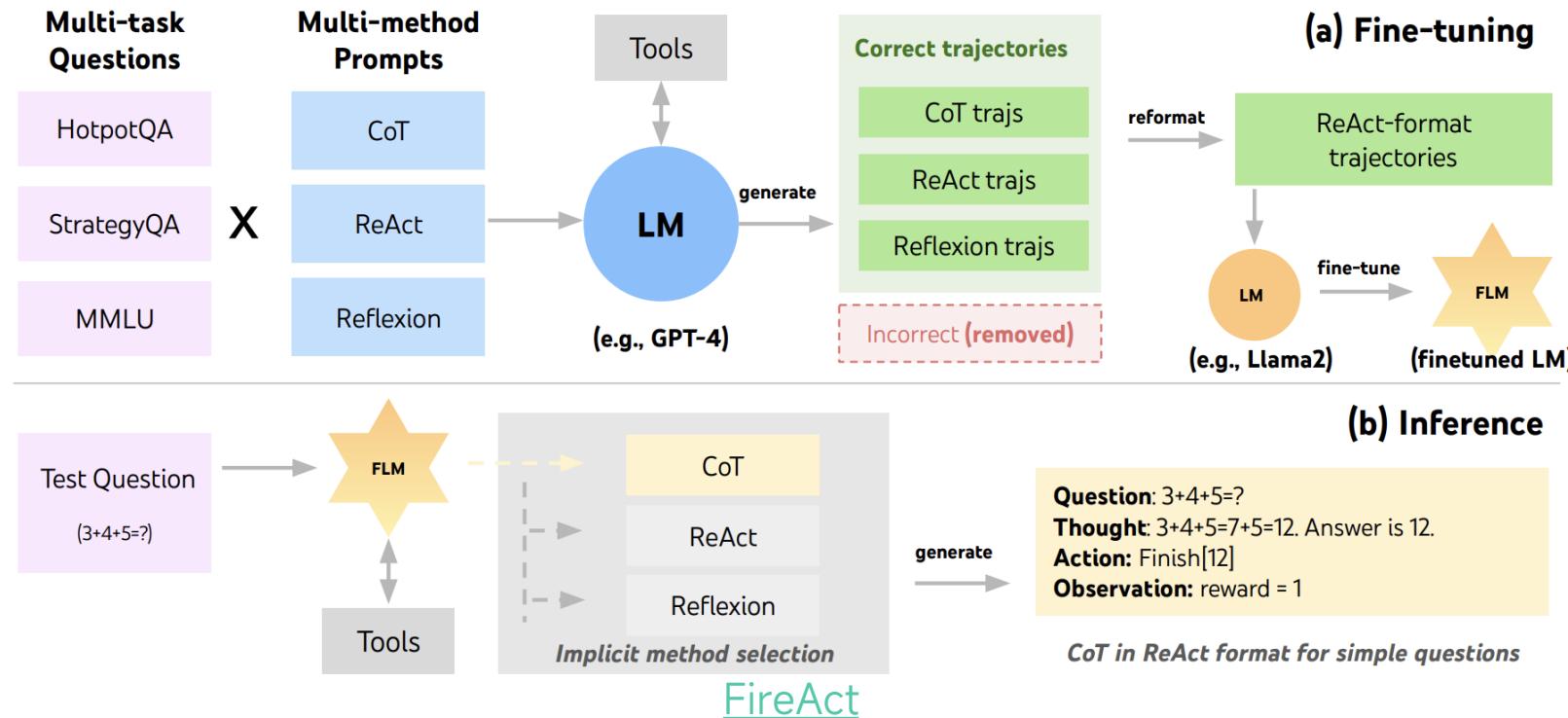
- Response generation: respond to user the process and results

Response Generation	Prompt
	#4 Response Generation Stage - With the input and the inference results, the AI assistant needs to describe the process and results. The previous stages can be formed as - User Input: {{ <i>User Input</i> }}, Task Planning: {{ <i>Tasks</i> }}, Model Selection: {{ <i>Model Assignment</i> }}, Task Execution: {{ <i>Predictions</i> }}. You must first answer the user's request in a straightforward manner. Then describe the task process and show your analysis and model inference results to the user in the first person. If inference results contain a file path, must tell the user the complete file path. If there is nothing in the results, please tell me you can't make it.

# Component Three: Tool Use



- ReAct-style Tool Calling
- (2) SFT-based methods: Frameworks like [Toolformer](#) employed a self-supervised objective to teach models where to insert API calls, while others like [FireAct](#) fine-tuned models on expert-generated or curated datasets of tool-interaction trajectories (e.g., [AgentBank](#)).



# Component Three: Tool Use

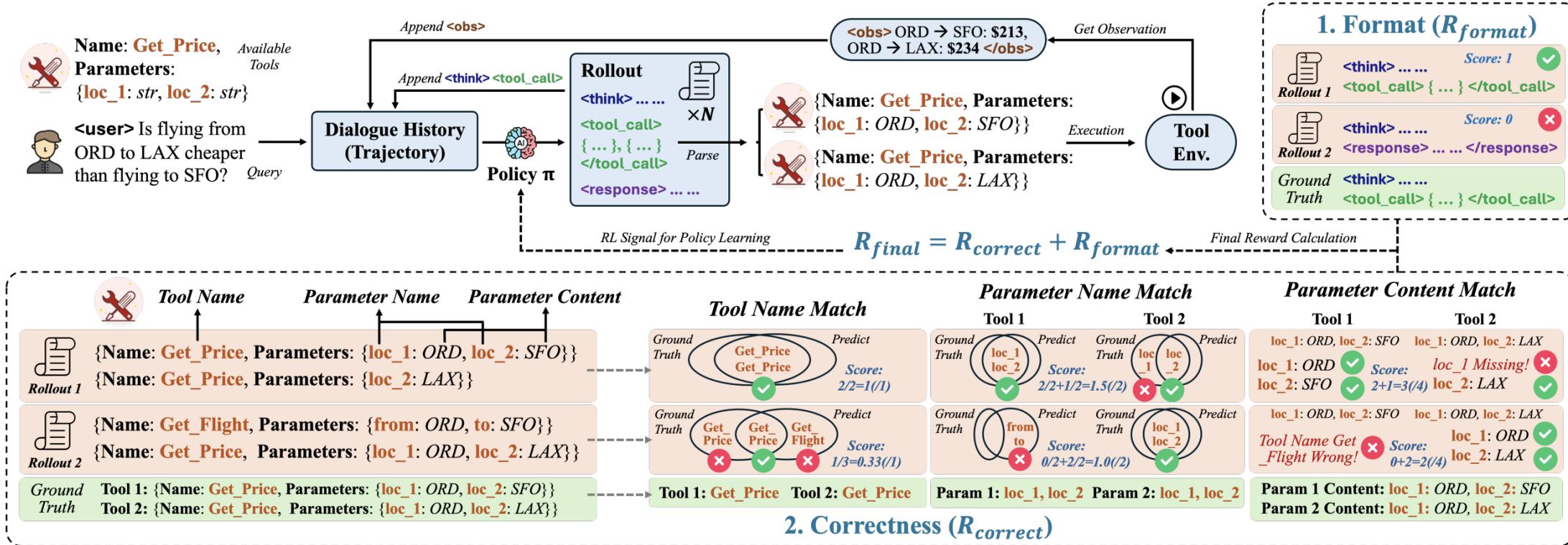


- Tool-integrated RL
- RL-based approaches for tool use shift the objective from replicating fixed patterns to optimizing overall task performance. This transition enables agents to strategically decide when, how, and in what combination to invoke tools, while dynamically adapting to novel contexts and unforeseen failures.

# Component Three: Tool Use



- Tool-integrated RL
- [ToolRL](#) demonstrates that even when starting from base models without any imitation traces, RL training can elicit emergent capabilities, such as self-correction, adaptive invocation frequency, and the composition of multiple tools for complex sub-tasks.



# Further Reading



- [LLM Powered Autonomous Agents | Lil' log](#)
- [Reflexion: Language Agents with Verbal Reinforcement Learning](#)
- [Auto-GPT for Online Decision Making: Benchmarks and Additional Opinions](#)
- [LLM agents: The ultimate guide 2025](#)
- [A Survey on Large Language Model based Autonomous Agents](#)
- [The Rise and Potential of Large Language Model Based Agents: A Survey](#)
  
- [\[2509.02547\] The Landscape of Agentic Reinforcement Learning for LLMs: A Survey](#)
- [<https://www.anthropic.com/engineering/building-effective-agents>](#)
- [\[2507.21046\] A Survey of Self-Evolving Agents: On Path to Artificial Super Intelligence](#)
- [cxxz/awesome-agent-memory](#)



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Thank you