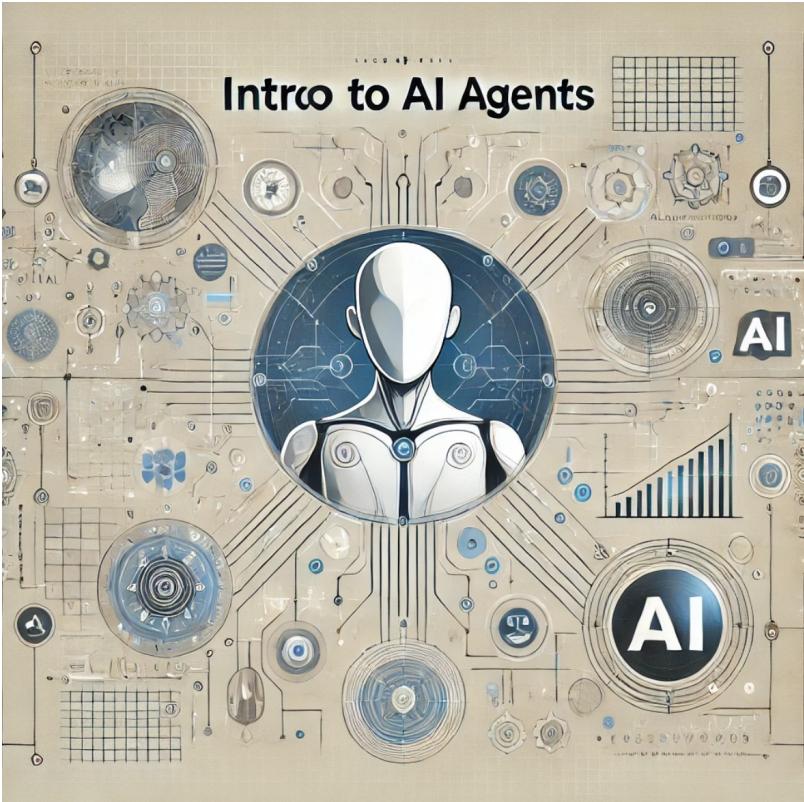


# Introduction to AI Agents

# Outline

- LLM-agent concepts
- Reasoning
- ReAct agents
- LLM-Agent evaluation



# Intro to AI Agents

# Introduction to AI Agents

**AI Agents** are autonomous entities capable of perceiving their environment, making decisions, and executing actions to achieve specific goals.

Fundamental  
models



Fundamental  
Agents

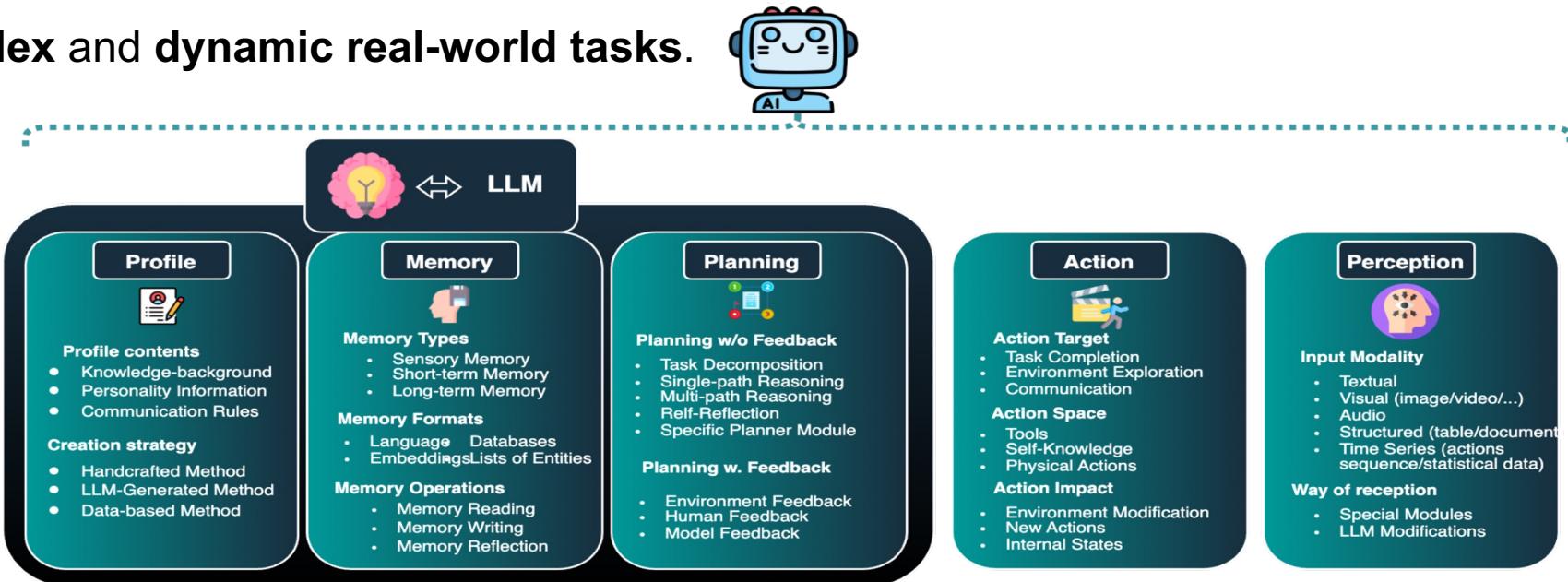
**Position: Foundation Agents as the Paradigm Shift for Decision Making**

Xiaoqian Liu, Xingzhou Lou, Jianbin Jiao, Junge Zhang

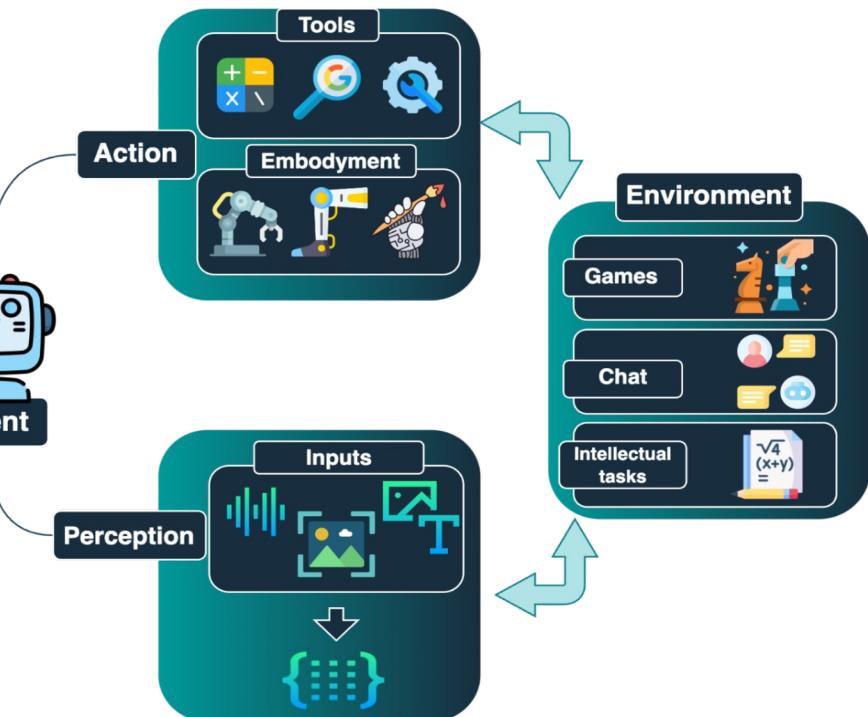
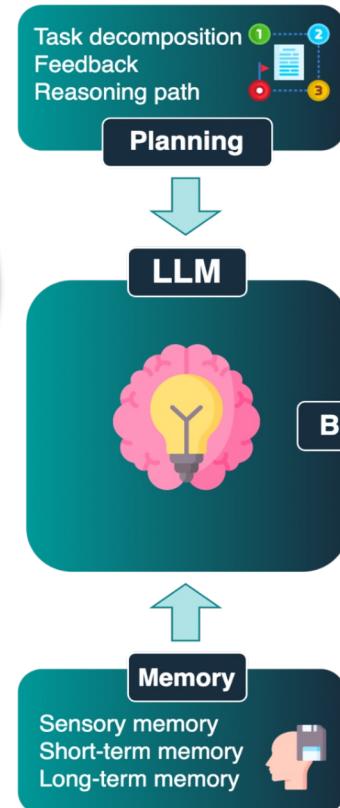
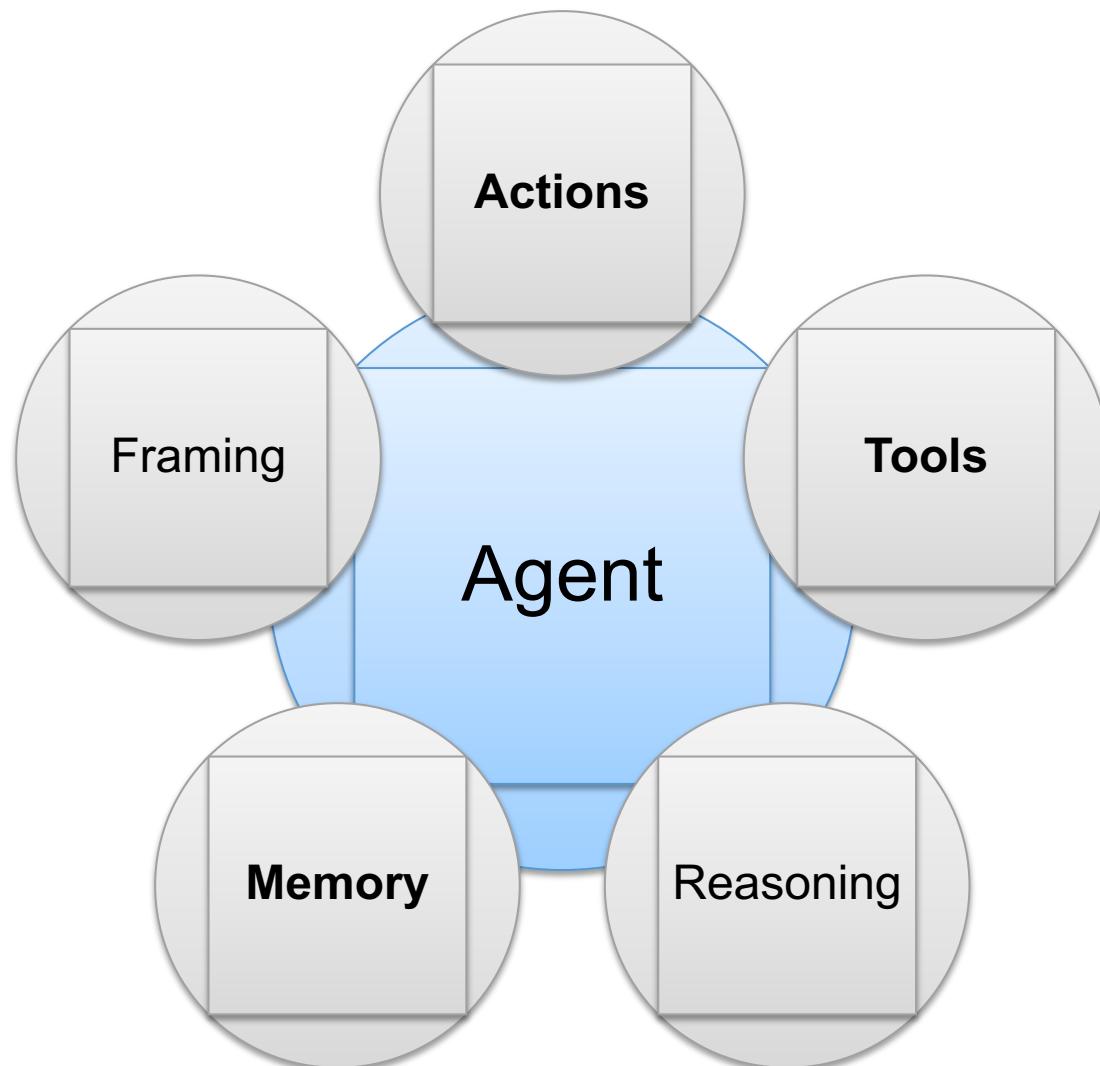
Decision making demands intricate interplay between perception, memory, and reasoning to discern optimal policies. Conventional approaches to decision making face challenges related to low sample efficiency and poor generalization. In contrast, foundation models in language and vision have showcased rapid adaptation to diverse new tasks. Therefore, we advocate for the construction of foundation agents as a transformative shift in the learning paradigm of agents. This proposal is underpinned by the formulation of foundation agents with their fundamental characteristics and challenges motivated by the success of large language models (LLMs). Moreover, we specify the roadmap of foundation agents from large interactive data collection or generation, to self-supervised pretraining and adaptation, and knowledge and value alignment with LLMs. Lastly, we pinpoint critical research questions derived from the formulation and delineate trends for foundation agents supported by real-world use cases, addressing both technical and theoretical aspects to propel the field towards a more comprehensive and impactful future.

# Introduction to AI Agents

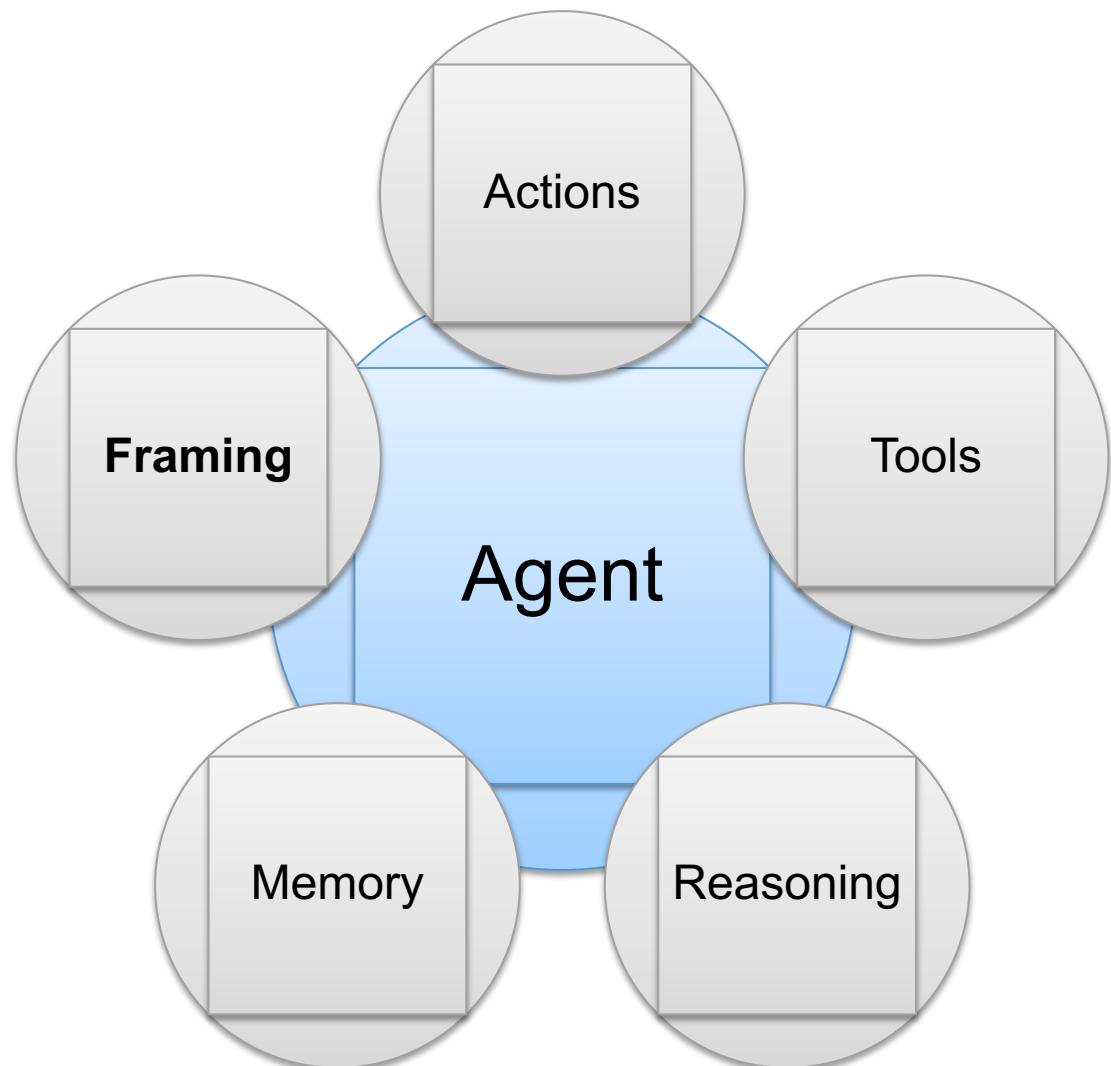
- **AI Agents** are autonomous entities capable of perceiving their environment, making decisions, and executing actions to achieve specific goals.
- AI agents have been developed to **operate in controlled environments**, such as games or simulated systems.
- Large Language Models (LLMs) has expanded the potential of AI agents, enabling them to handle **more complex** and **dynamic real-world tasks**.



# Actions and tools



# Framing



- **Framing** in LLMs is the way you set up the situation or role for the model, so it knows how to respond in a specific way.
- In simple terms: framing is about **giving the model a "frame"** — like setting the scene — so it acts the way you want.

**Through framing, you control:**

- the style and tone of the answer,
- the perspective or role the model takes,
- what aspects the model should focus on.

Example:

- "*You are a friendly teacher explaining to kids,*" the model will respond in a simple, cheerful way.
- "*You are a legal expert analyzing a contract,*" the model will answer formally and precisely.

# Framing

## Why framing matters:

LLMs don't "know" the world like humans — they rely heavily on **how the prompt is written.**

Good framing can completely change the quality, depth, and relevance of their responses.

- **Framing** in LLMs is the way you set up the situation or role for the model, so it knows how to respond in a specific way.
- In simple terms: framing is about **giving the model a "frame"** — like setting the scene — so it acts the way you want.

## Through framing, you control:

- the style and tone of the answer,
- the perspective or role the model takes,
- what aspects the model should focus on.

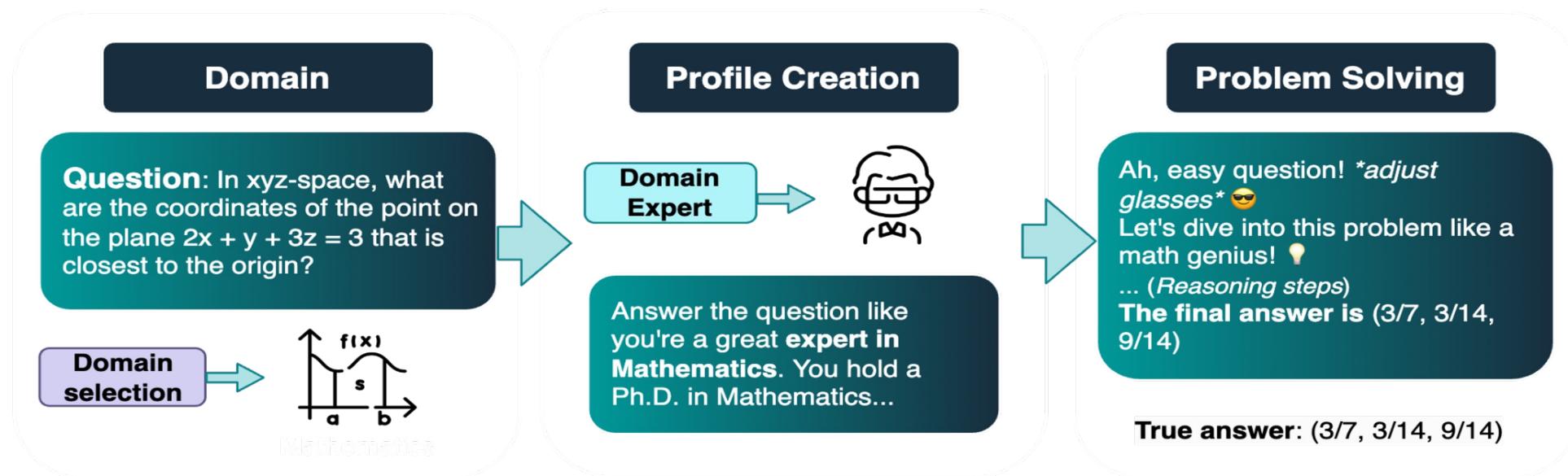
## Example:

- "*You are a friendly teacher explaining to kids,*" the model will respond in a simple, cheerful way.
- "*You are a legal expert analyzing a contract,*" the model will answer formally and precisely.

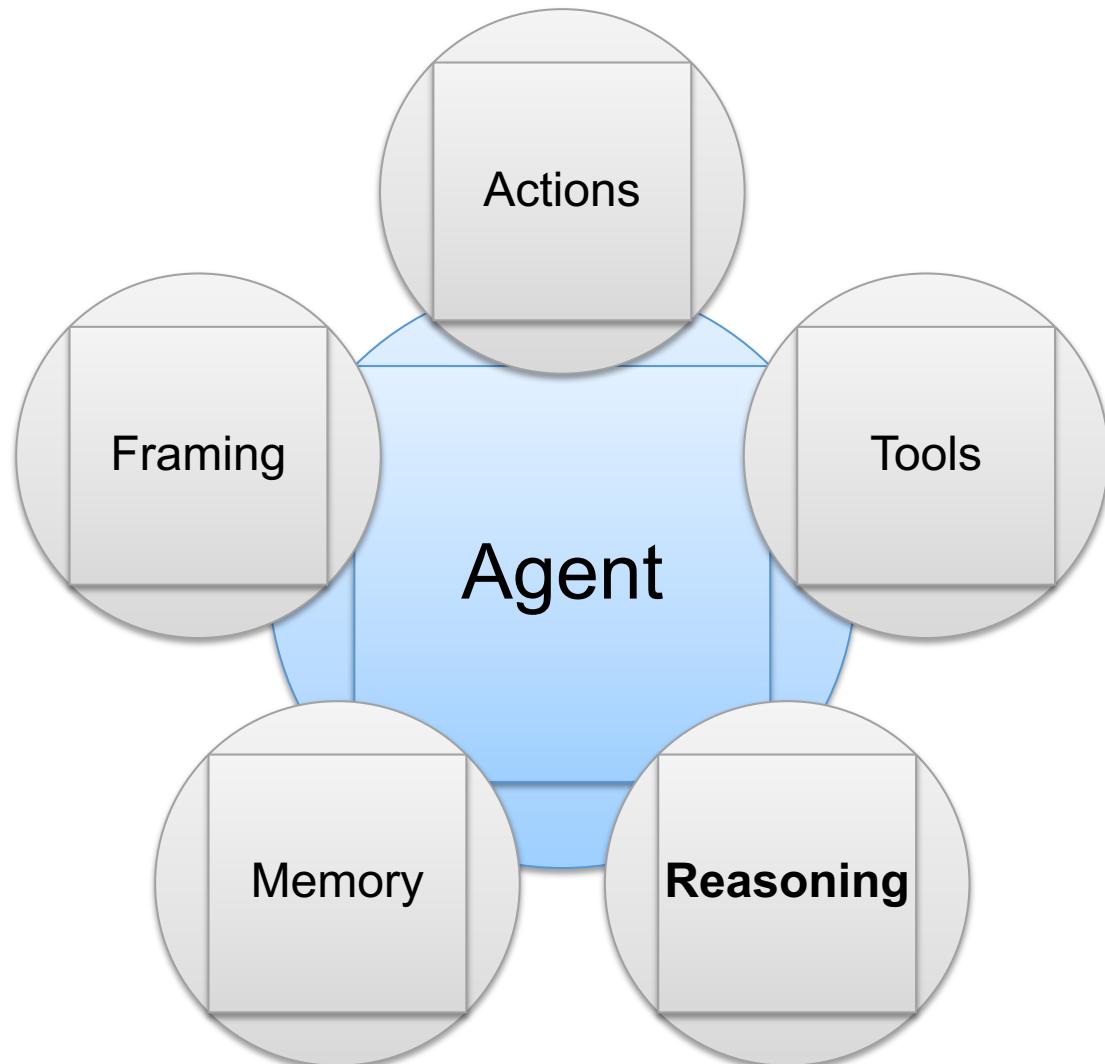
# Agent roles

**Dialogue roles for LLMs** allow them to unlock knowledge within a specific domain that previously remained in the background — their "experience baggage." This capability enables the model to:

- create more effective chains of reasoning,
- solve tasks with greater accuracy.



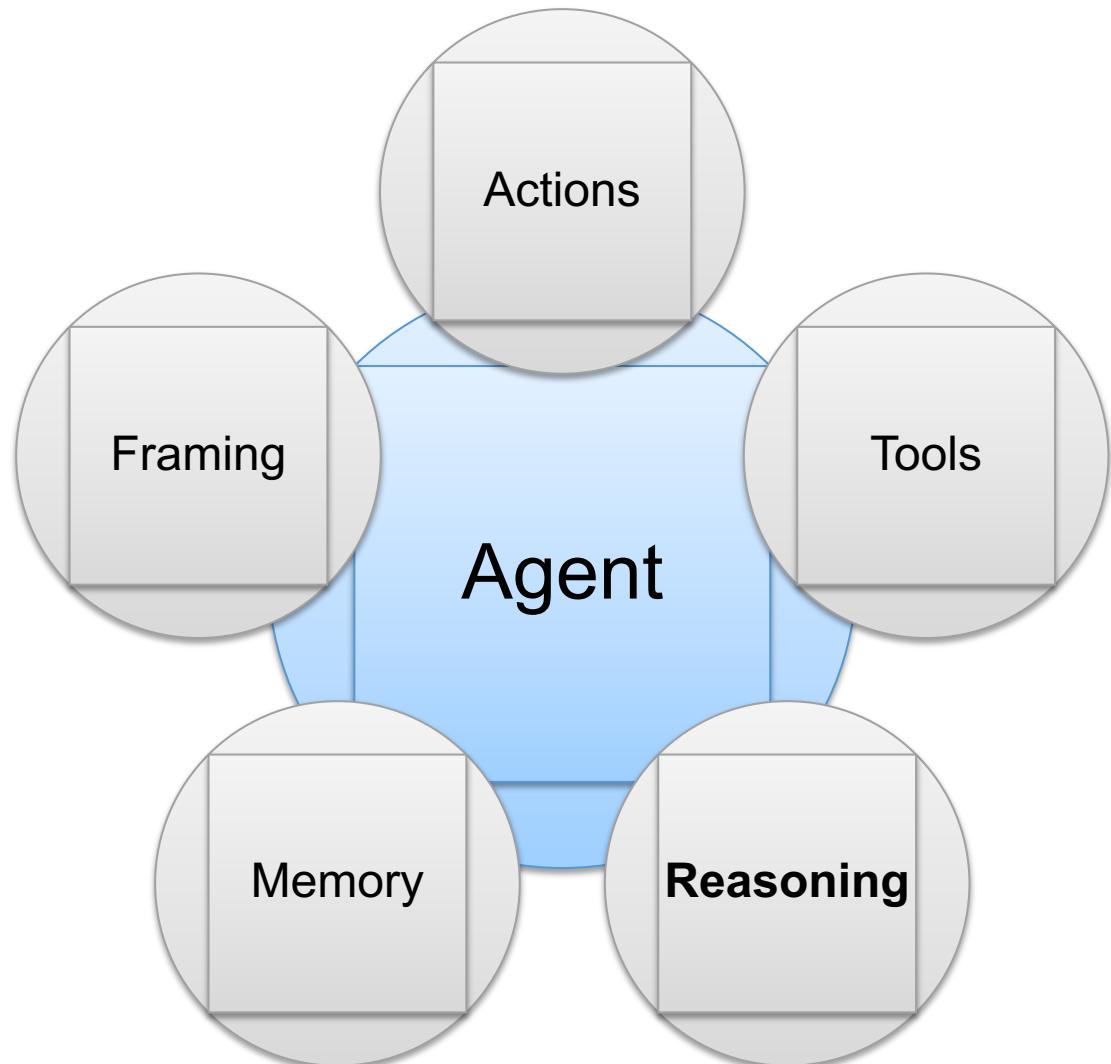
# Reasoning in LLMs



Reasoning in LLMs includes:

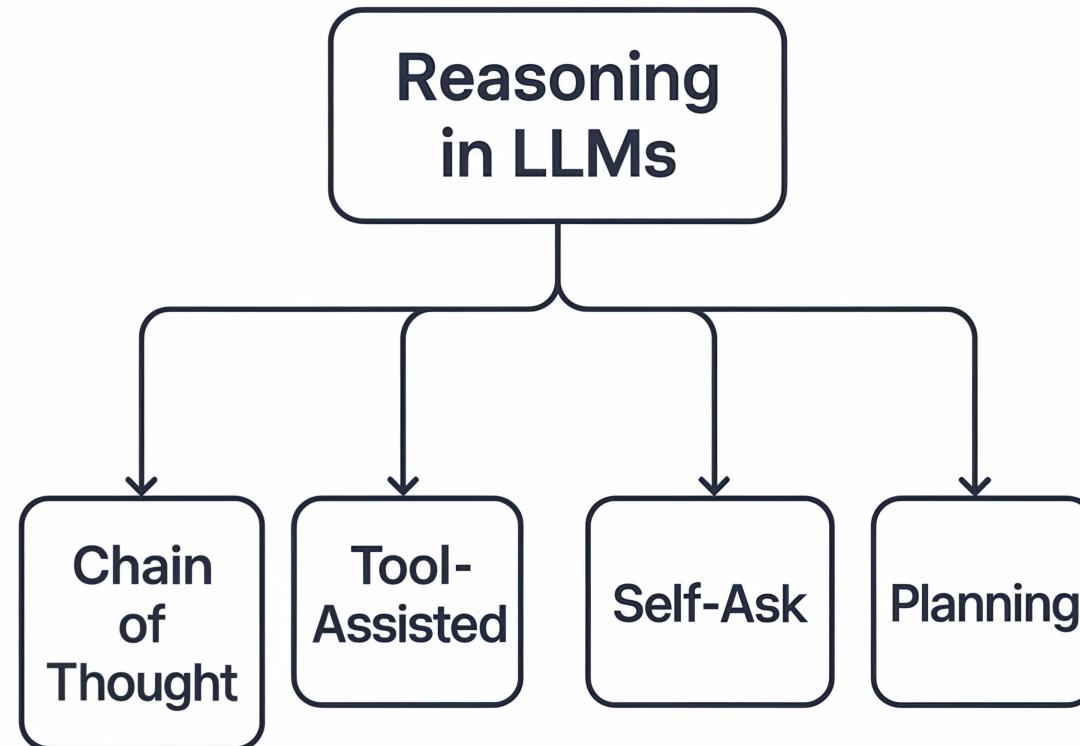
- logical thinking,
- step-by-step calculations,
- understanding cause-and-effect relationships,
- being able to generalize even if the exact answer wasn't in its training data.

# Reasoning in LLMs



- **Reasoning** in LLMs is the model's ability to make logical conclusions, think step-by-step, and come up with meaningful answers.
- **Important:** LLMs don't “think” like humans. They predict the next word, but thanks to training, it *looks* like it's actually reasoning.

# Types of reasoning



# Types of reasoning

## Chain of Thought Reasoning:

- The model **explains its steps one by one** before giving the final answer.
- **Chain of Thought** helps models solve harder problems because they don't have to "guess" the final answer at once — they think along the way.

### Example:

Question: Masha had 5 candies. She ate 2. How many are left?

Answer: Masha had 5 candies. She ate 2.  $5 - 2 = 3$ . So 3 candies are left.

## Tool-Assisted Reasoning:

- This is when the model uses external tools (e.g., calculator, a search engine, or a database) to get the answer.

### Example:

Question: Calculate  $12345 \times 6789$ .

The model realizes, "I'm not great at big multiplications," and sends the task to a calculator.

# Types of reasoning

## Chain of Thought Reasoning:

- The model **explains its steps one by one** before giving the final answer.
- **Chain of Thought** helps models solve harder problems because they don't have to "guess" the final answer at once — they think along the way.

### Example:

Question: Masha had 5 candies. She ate 2. How many are left?

Answer: Masha had 5 candies. She ate 2.  $5 - 2 = 3$ . So 3 candies are left.

## Tool-Assisted Reasoning:

- This is when the model uses external tools (e.g., calculator, a search engine, or a database) to get the answer.

### Example:

Question: Calculate  $12345 \times 6789$ .

The model realizes, "I'm not great at big multiplications," and sends the task to a calculator.

## Self-Ask Reasoning:

- Here, the model asks itself small follow-up questions to break down the bigger problem.

### Example:

Question: How many years passed between 1990 and 2020?

Model's thinking:

— What's the difference between 1990 and 2020? → 30 years.

Final answer: 30 years.

## Planning-Based Reasoning:

- The model **first makes a plan** — a list of steps — and then follows the plan to solve a complex task.

# DeepSeek aha moment

---

Question: If  $a > 1$ , then the sum of the real solutions of  $\sqrt{a - \sqrt{a+x}} = x$  is equal to

---

Response: <think>

To solve the equation  $\sqrt{a - \sqrt{a+x}} = x$ , let's start by squaring both ...

$$(\sqrt{a - \sqrt{a+x}})^2 = x^2 \implies a - \sqrt{a+x} = x^2.$$

Rearrange to isolate the inner square root term:

$$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$$

...

**Wait, wait. Wait. That's an aha moment I can flag here.**

Let's reevaluate this step-by-step to identify if the correct sum can be ...

We started with the equation:

$$\sqrt{a - \sqrt{a+x}} = x$$

First, let's square both sides:

$$a - \sqrt{a+x} = x^2 \implies \sqrt{a+x} = a - x^2$$

Next, I could square both sides again, treating the equation: ...

...

---

Table 3 | An interesting “aha moment” of an intermediate version of DeepSeek-R1-Zero. The model learns to rethink using an anthropomorphic tone. This is also an aha moment for us, allowing us to witness the power and beauty of reinforcement learning.

# ReAct agents



# ReAct as a standard for modern Agents<sup>17</sup>

**Re-Act (Reason + Act)** is a paradigm that combines **reasoning and action execution** in the operation of a language model.

Re-Act makes the model **alternate between logical reasoning and calling user-defined functions (or tools)** to interact with the external world.

Computer Science > Computation and Language

[Submitted on 6 Oct 2022 ([v1](#)), last revised 10 Mar 2023 (this version, v3)]

## ReAct: Synergizing Reasoning and Acting in Language Models

[Shunyu Yao](#), [Jeffrey Zhao](#), [Dian Yu](#), [Nan Du](#), [Izhak Shafran](#), [Karthik Narasimhan](#), [Yuan Cao](#)

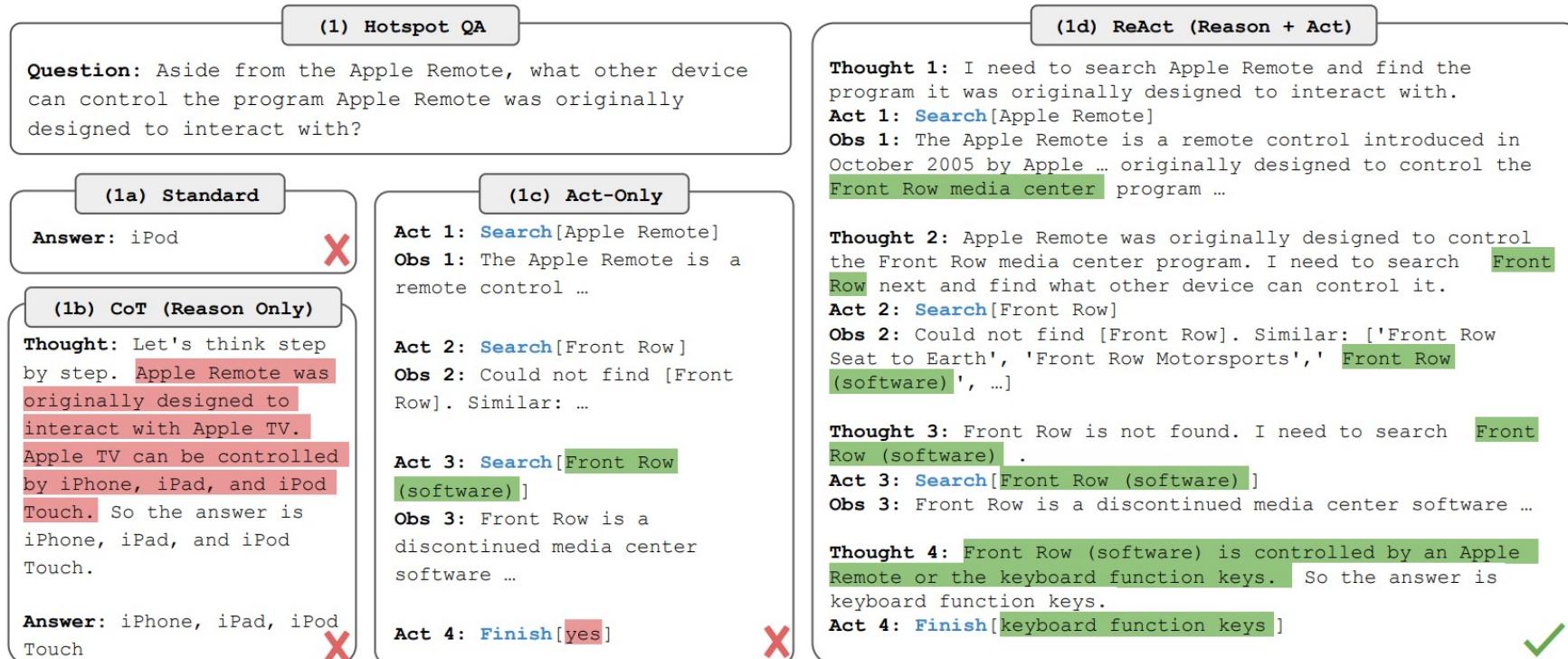
While large language models (LLMs) have demonstrated impressive capabilities across tasks in language understanding and interactive decision making, their abilities for reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. 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 external sources, such as knowledge bases or environments, to gather additional information. 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, as well as improved human interpretability and trustworthiness over methods without reasoning or acting components. Concretely, on question answering (HotpotQA) and fact verification (Fever), ReAct overcomes issues of hallucination and error propagation prevalent in chain-of-thought reasoning by interacting with a simple Wikipedia API, and generates human-like task-solving trajectories that are more interpretable than baselines without reasoning traces. 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. Project site with code: [this https URL](#)

[https://arxiv.org/pdf/2210.03629](https://arxiv.org/pdf/2210.03629.pdf)

<https://habr.com/ru/articles/882432/>

# ReAct as a standard for modern Agents

Re-Act makes the model alternate between logical reasoning and calling user-defined functions (or tools) to interact with the external world.



# ReAct agent components

- **Language Model (LLM)** is *the “brain”* of the agent that generates reasoning steps. It produces chains of thought and specifies which actions to take.
- **Tools** are *external functions* or APIs that the agent can call to retrieve information or perform operations.
- **Agent Executor** executes *the model’s chosen action* by invoking the corresponding tool and returning the result back to the model.  
This component implements the cycle: ***Thought*** → ***Act*** → ***Observe***.
- **Agent State (State)** contains *the accumulated history* of the dialogue and actions.
  - It includes the user’s query, the model’s messages (thoughts/actions), and observations (tool outputs).
  - The state is continuously updated as each step is executed.

<https://arxiv.org/pdf/2210.03629>

<https://habr.com/ru/articles/882432/>

# ReAct agent components

**Receiving the Query.** The agent receives the user's input message, which becomes the initial state.

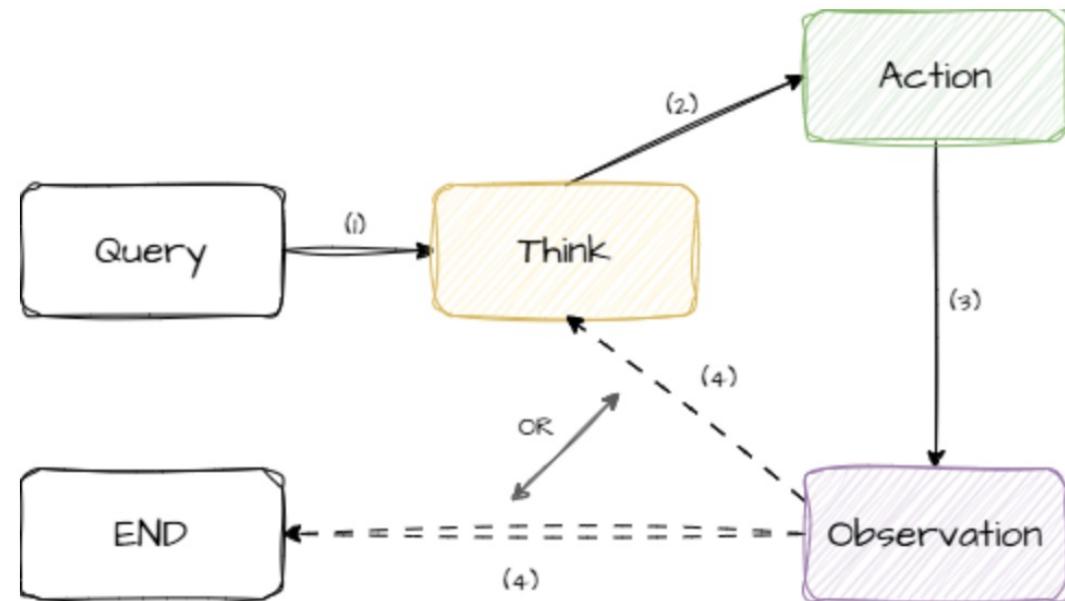
**Generating reasoning and action (Think).** Based on the current state, the language model produces an output consisting of:

- **Thought:** reasoning — the agent analyzes the task and determines the next step.
- **Action:** an instruction to perform a specific operation.

**Executing the Action.** If the generated output includes a command (e.g., Action: Search["query"]), the executor calls the corresponding tool and retrieves the result.

**Updating the state (Observation).** The tool's result is saved as an observation and added to the interaction history. The updated state is then passed back to the model for the next iteration.

**Completion (END).** If the model generates a final answer (e.g., Final Answer: ...), the loop stops, and the response is returned to the user.



Цикл работы Re-Act агента

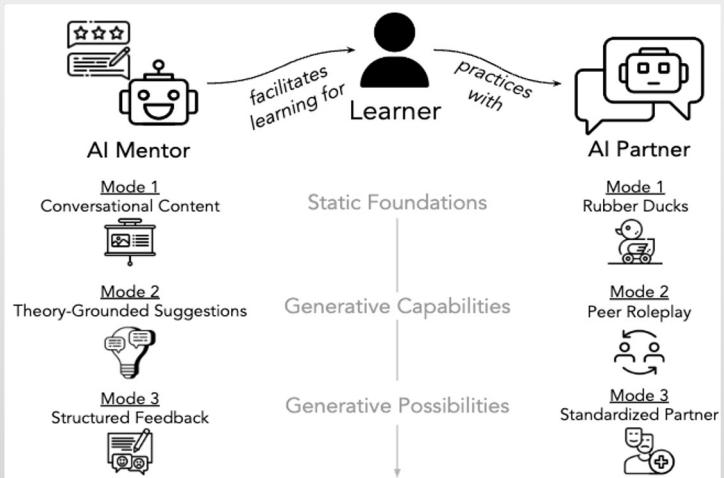
# Usage Examples

- Next-generation virtual assistants
- Automation of business processes and IT support
- Generation of complex multimodal content (text + image + video)
- Education, healthcare, and finance...



# Cases

## Task SoftSkill development



**Task**  
Based on the lease document, generate a contract in the form of a JSON structure.

# On the way to multiagent

Task difficulty

**LLM**

**Agents (LLMs + Tools)**

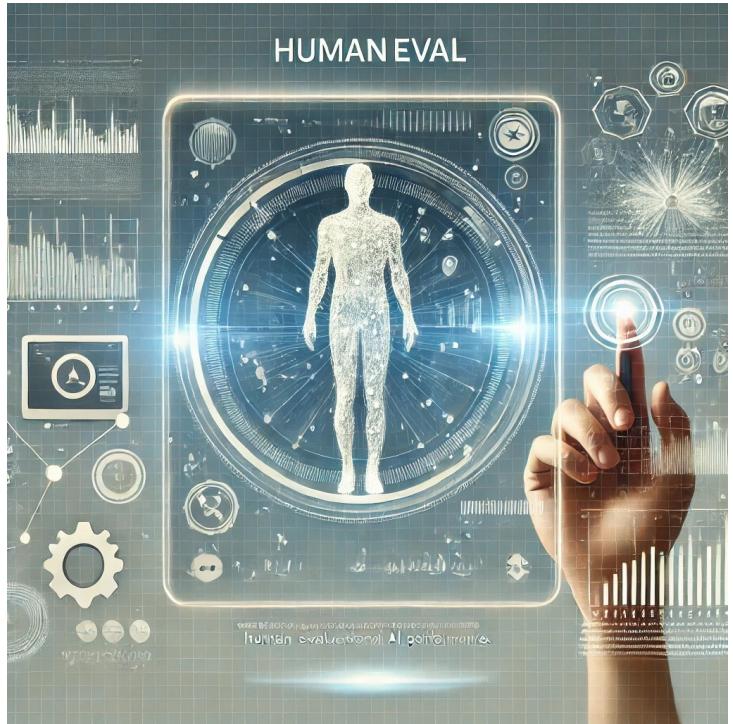
**Multiple Agents**

LLMs are instruction-following systems prone to errors, hallucinations, and challenges with multi-step reasoning in their responses.

**Integrating external tools + RAG** reduces the risk of hallucinations and enables the handling of more complex tasks, such as code execution.

It enables solving tasks in multiple directions (**roles**) and establishing specialized forms of communication between agents for information exchange (shared context).

# AI Agents Evaluation



# Agent evaluation challenges

Agent evaluation differs from LLM in fundamental ways.

Tasks for them are:

- *harder*
- *more realistic*
- *have more real-world utility*
- *not always a single correct answer*



Evaluation challenges:

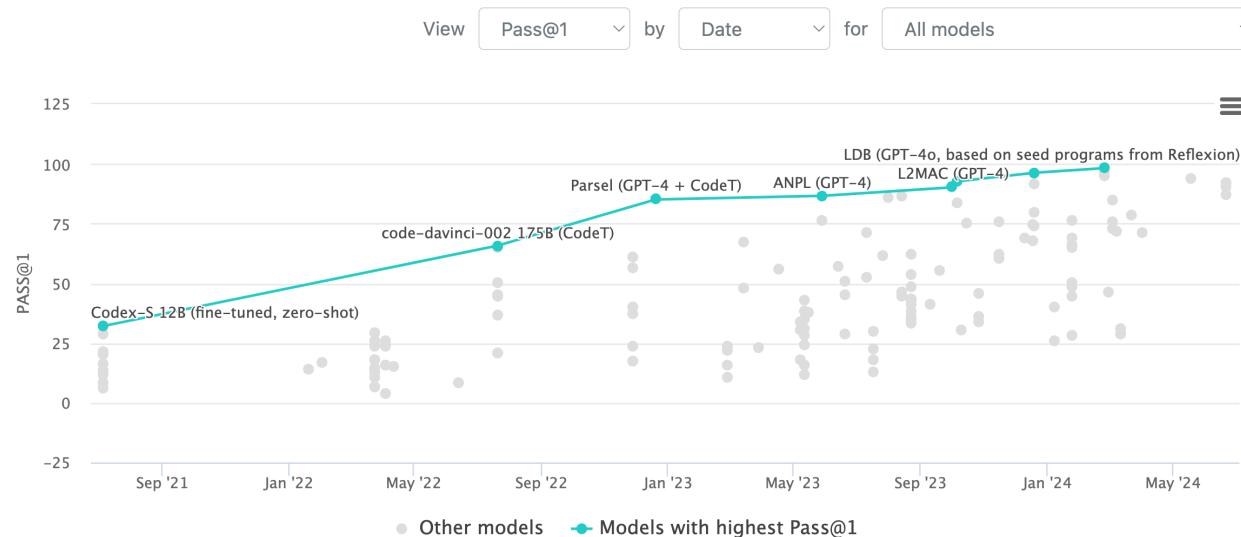
- 1. AI agent evaluations must be cost-controlled.** The language models underlying most AI agents are stochastic. This means simply calling the model multiple times can increase accuracy and cost.
- 2. Jointly optimizing accuracy and cost can yield better agent design.**
- 3. Model developers and downstream developers have distinct benchmarking needs.** Benchmarks meant for model evaluation can be misleading for downstream evaluation. Downstream evaluation should account for dollar costs.
- 4. Agent benchmarks enable shortcuts.** Without proper hold-out sets/tasks/domain, models can overfit, even unintentionally.
- 5. Agent evaluations lack standardization and reproducibility.**



# Human eval is a code classics

Used to measure functional correctness for synthesizing programs from docstrings.

Consists of 164 original programming problems, assessing language comprehension, algorithms, and simple mathematics.



```
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]

def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) =>12
    solution([3, 3, 3, 3, 3]) =>9
    solution([30, 13, 24, 321]) =>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)

def encode_cyclic(s: str):
    """
    returns encoded string by cycling groups of three characters.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group. Unless group has fewer elements than 3.
    groups = [(group[1:] + group[0]) if len(group) == 3 else group for group in groups]
    return ''.join(groups)

def decode_cyclic(s: str):
    """
    takes as input string encoded with encode_cyclic function. Returns decoded string.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group.
    groups = [(group[-1] + group[:-1]) if len(group) == 3 else group for group in groups]
    return ''.join(groups)
```

# ruHumanEval

Russian **HumanEval** (**ruHumanEval**) is the Russian analogue of the original HumanEval dataset from MERA benchmark, created to evaluate the ability of language models to generate code in the Python programming language to solve simple problems.

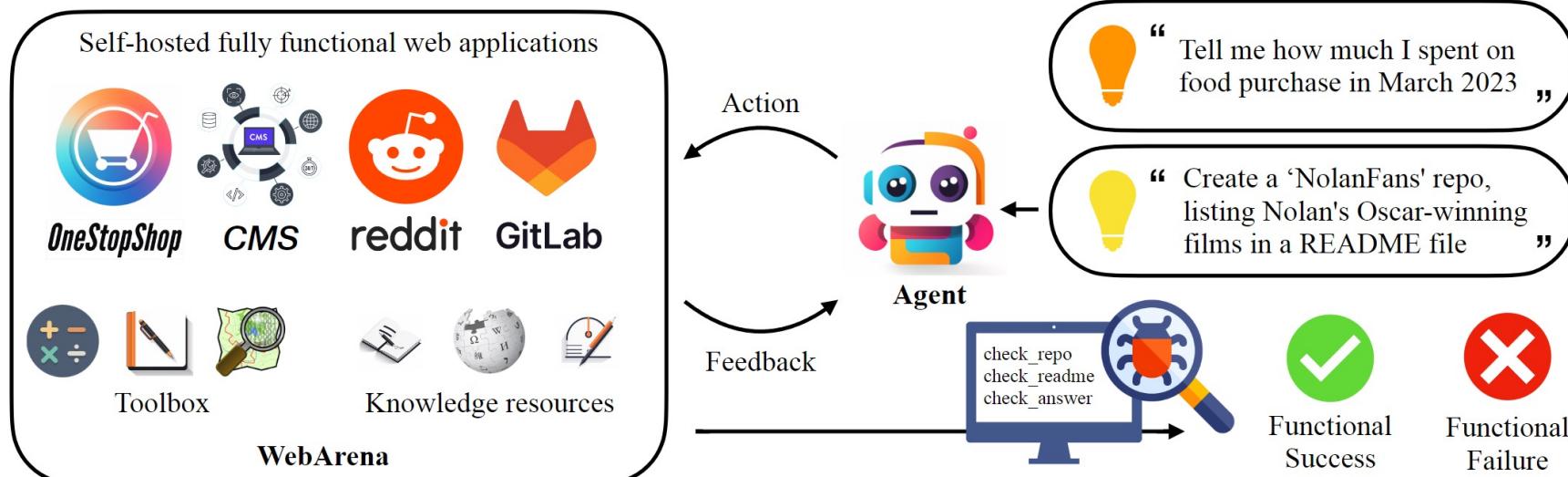
## ruHumanEval

Assessing models' abilities to generate solutions for straight-forward programming problems on Python. The solution is considered to be correct if it passes all test case.

```
"instruction": "На вход подается функция с описанием в виде строки docstring. В соответствии с описанием вам необходимо реализовать функцию на основе шаблона:\n{function}"  
"inputs": {  
    "function":  
        "def greatest_common_divisor(a: int, b: int) -> int:  
            '''Верните наибольший общий делитель двух целых чисел a и b.  
            Примеры: greatest_common_divisor(3, 5)  
            1'''",  
    "tests": "[{'a': 3, 'b': 7}, {'a': 10, 'b': 15}, {'a': 49, 'b': 14}, {'a': 144, 'b': 60}]"  
},  
"outputs": ["1", "5", "7", "12"]
```

# WebArena

- WebArena is web environment for building autonomous agents.
- WebArena creates websites from four popular categories with functionality and data **mimicking their real-world equivalents**.
- To emulate human problem-solving, WebArena also embeds tools and knowledge resources as independent websites.



# Realistic tasks on WebArena

A high-level task that can be fully executed in WebArena.

Completing such tasks requires sophisticated, long-term planning and reasoning capability.

## Examples:

*Subscribe to the newsletter on OneStopMarket*

*Cancel order 307*

*Set my Gitlab status as enjoying life*



“ Create an efficient itinerary to visit all Pittsburgh's art museums with minimal driving distance starting from CMU. Log the order in my “awesome-northeast-us-travel” repository

webarena.wikipedia.com

Wikipedia Pittsburgh museums

List of museums in Pittsburgh

This list of museums in Pittsburgh, Pennsylvania encompasses

museums defined for this context as institutions (including nonprofit organizations) that collect, care for, preserve, display, interpret, and explain artifacts, history, and culture to the public viewing. Also included are non-profit organizations that collect artifacts that exist only in cyberspace (i.e., virtual museums) are not included.

Wikimedia Commons has media related to [Museums in Pittsburgh](#).

See also: [List of museums in Pennsylvania](#)

▼ Museums

Search for museums in Pittsburgh

webarena.openstreetmap.com

OpenStreetMap Edit History Export

Carnegie Mellon University, Schenley Drive E

The Andy Warhol Museum, 117, Sandusky St

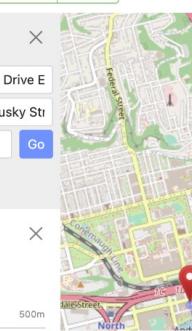
Car (OSRM) Go Reverse Directions

Directions

Distance: 6.5km. Time: 0:09.

↑ 1. Start on Forbes Avenue

500m



Search for each art museum on the Map

webarena.gitlab.com

README.md 158 B

Travel in Northeast US

Pittsburgh

- CMU

+ Miller Gallery at Carnegie Mellon U

+ American Jewish Museum

+ Carnegie Museum of Art

...

Travel in Northeast US

Pittsburgh

- CMU

+ Miller Gallery at Carnegie Mellon U

+ American Jewish Museum

+ Carnegie Museum of Art

...

Record the results to t

...

# Observation space

The observation is the URL and the content of a web page, with options to represent the content as a screenshot, HTML DOM tree, and accessibility tree.

Patio, Lawn & Garden

Shop By

Category

- Gardening & Lawn Care (168)
- Patio Furniture & Accessories (92)

Price

- \$0.00 - \$999.99 (311)
- \$1,000.00 - \$1,999.99 (8)
- \$3,000.00 and above (1)

Compare Products

You have no items to compare.

My Wish List

You have no items in your wish list.

**Outdoor Patio Folding Side Table**  
Square Metal End Table, Portable Small Bistro Coffee Table, Green  
★★★★★ 12 Reviews  
\$49.99 Add to Cart

**Shop Succulents | Assorted Collection of Live Air Plants, Hand Selected Variety Pack of Air Succulents | Collection of 6**  
\$21.96 Add to Cart

**ENEVOTX Front Door Side Window Covering Alligator and Cactus Decor for Front Door Durable Fabric Decor for Door Multi Size Door Protector for Bedroom Home Kitchen Party Decoration**  
\$38.00 Add to Cart

```
<li>
<div>
<a href="#"></a>
<div class="product-item">
<a href="#">Outdoor Patio ...
</a>
<div>
<span>Rating:</span>
<div>
<span>82%</span>
</div>
<a href="#reviews">12 ...
<span>Reviews</span></a>
</div>
</div>

```

```
[4] RootWebArea 'Patio, Lawn ..'
[1543] link 'Image'
[1547] img 'Image'
[1552] link 'Outdoor Patio..'
[1549] LayoutTable ''
[1559] StaticText 'Rating:'
[1557] generic '82%'
[1567] link '12 Reviews'
[1574] StaticText '$49.99'
[1582] button 'Add to Cart' focusable: True
[1585] button 'Wish List' focusable: ...
[1586] button 'Compare' focusable: ...
```

# Evaluation

Two evaluation approaches:

- **The correctness of performing information seeking tasks.** It compares the predicted answer with the annotated reference with three implementations.
- **If the intermediate states during the executions possess the anticipated properties specified by the intent.**

Function	ID	Intent	Eval Implementation
$r_{\text{info}}(a^*, \hat{a})$	1	Tell me the name of the customer who has the most cancellations in the history	<code>exact_match(<math>\hat{a}</math>, "Samantha Jones")</code>
	2	Find the customer name and email with phone number 8015551212	<code>must_include(<math>\hat{a}</math>, "Sean Miller")</code> <code>must_include(<math>\hat{a}</math>, "sean@gmail.com")</code>
	3	Compare walking and driving time from AMC Waterfront to Randyland	<code>fuzzy_match(<math>\hat{a}</math>, "Walking: 2h58min")</code> <code>fuzzy_match(<math>\hat{a}</math>, "Driving: 21min")</code>
$r_{\text{prog}}(s)$	4	Checkout merge requests assigned to me	<code>url = locate_last_url(s)</code> <code>exact_match(url, "gitlab.com/merge_requests?assignee_username'=byteblaze")</code>
	5	Post to ask “whether I need a car in NYC”	<code>url = locate_latest_post_url(s)</code> <code>body = locate_latest_post_body(s)</code> <code>must_include(url, "/f/nyc")</code> <code>must_include(body, "f/nyc")</code>

<https://webarena.dev/>  
<https://arxiv.org/pdf/2307.13854>

# AgentBench

- **AgentBench** is a multi-dimensional benchmark for LLMs as agent evaluation in interactive environments.
- It assesses LLMs' reasoning and decision-making abilities across eight distinct environments of three main types: Code-Grounded, Game-Grounded, and Web-grounded.



## Real-world Challenges

(On an Ubuntu bash terminal)  
Recursively set all files in the directory to read-only, except those of mine.

(Given Freebase APIs)  
What musical instruments do Minnesota-born Nobel Prize winners play?

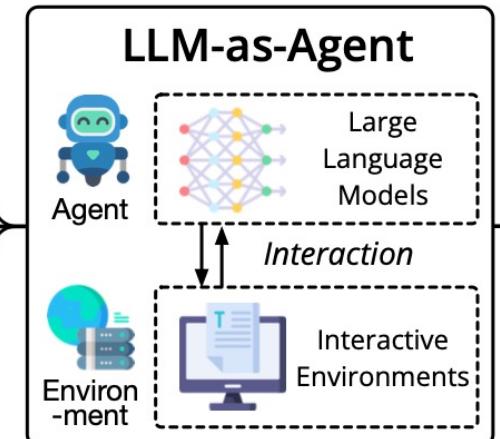
(Given MySQL APIs and existed tables)  
Grade students over 60 as PASS in the table.

(On the GUI of Aquawar)  
This is a two-player battle game, you are a player with four pet fish cards .....

A man walked into a restaurant, ordered a bowl of turtle soup, and after finishing it, he committed suicide. Why did he do that?

(In the middle of a kitchen in a simulator)  
Please put a pan on the dinning table.

(On the official website of an airline)  
Book the cheapest flight from Beijing to Los Angeles in the last week of July.

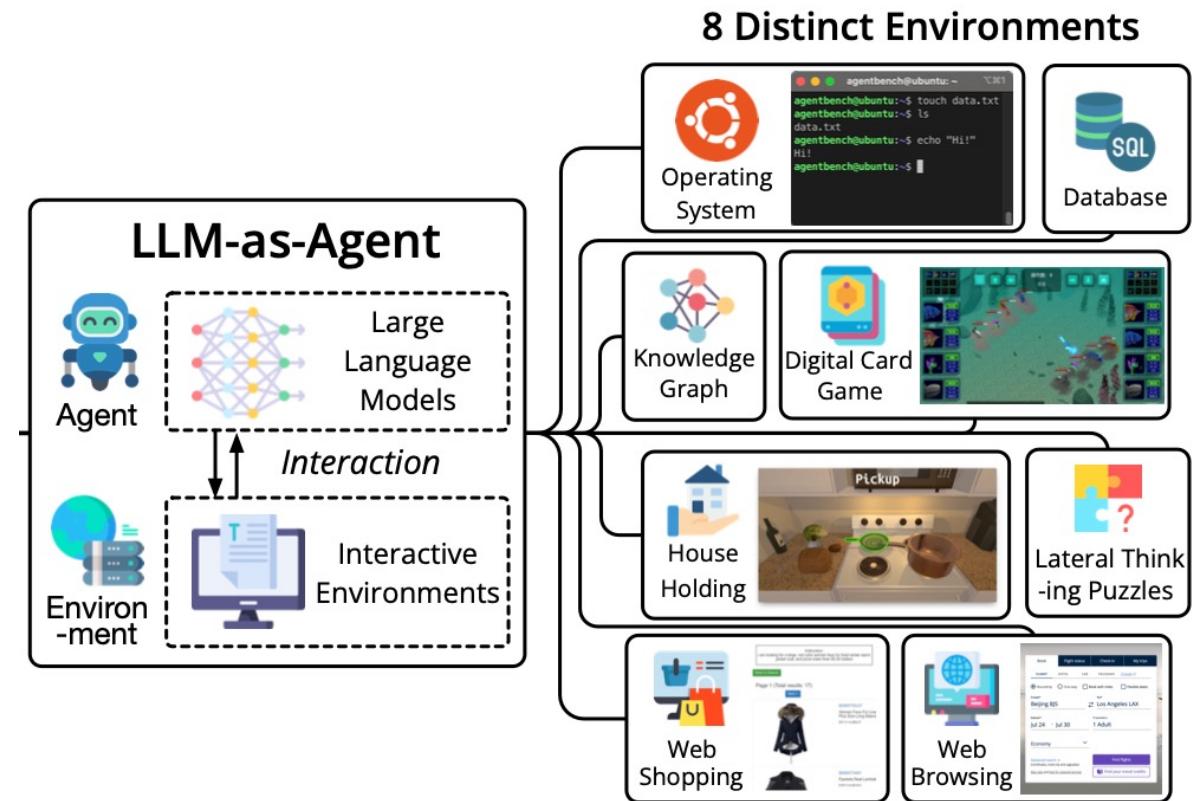


# AgentBench Environments

## Three Types of Groundings:

- **Code-Grounded Environments** coding and reasoning abilities.
- **Game-Grounded Environments** require no expertise in coding but more integral grasping of commonsense and world knowledge.
- **Web-Grounded Environments** interfaces for people to interact in the real world => **critical and valuable for following development.**

Each environment simulates real-world challenges where LLMs operate as autonomous agents.



# Evaluation methodology

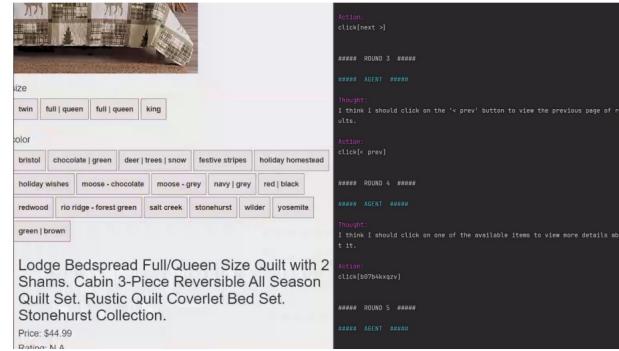
- AgentBench uses a **Chain-of-Thought (CoT)** approach, a standard reasoning strategy for LLMs.
- The evaluation is structured as an **Interactive Evaluation**.
- LLMs are assessed on their ability to follow instructions, perform coding tasks, and engage in multi-turn interactions.
- **Metrics:** Success Rate (SR), F1 score, and Game Progress, among others.
- The **overall score** is calculated by resizing each task's average score to 1 and then averaging across all tasks for each model.



(d) **Digital Card Game (DCG)**

*Task : “Compete against another player using four ‘fish’ cards in ‘Aquawar’ game.”*

*Action space: Four ‘fish’ cards and Assertion  
Observation: Battle process, status of ‘fish’*



(g) **Web Shopping (WS)**

*Task : “Looking for a queen size bedspread set in the color redwood, and price lower than 70.”*

*Action space: Search (generate keywords) and Click (choose from all clickable buttons)  
Observation: Products’ descriptions; the webpage*

# LLMArena

- **LLMArena** a dynamic evaluation benchmark specially designed for multi-agent interaction.
- Covers **seven different types** of dynamic, multi-agent game environments.
- Focusses on assessing LLMs in scenarios that require **spatial reasoning, strategic planning, communication, modeling, and collaboration.**

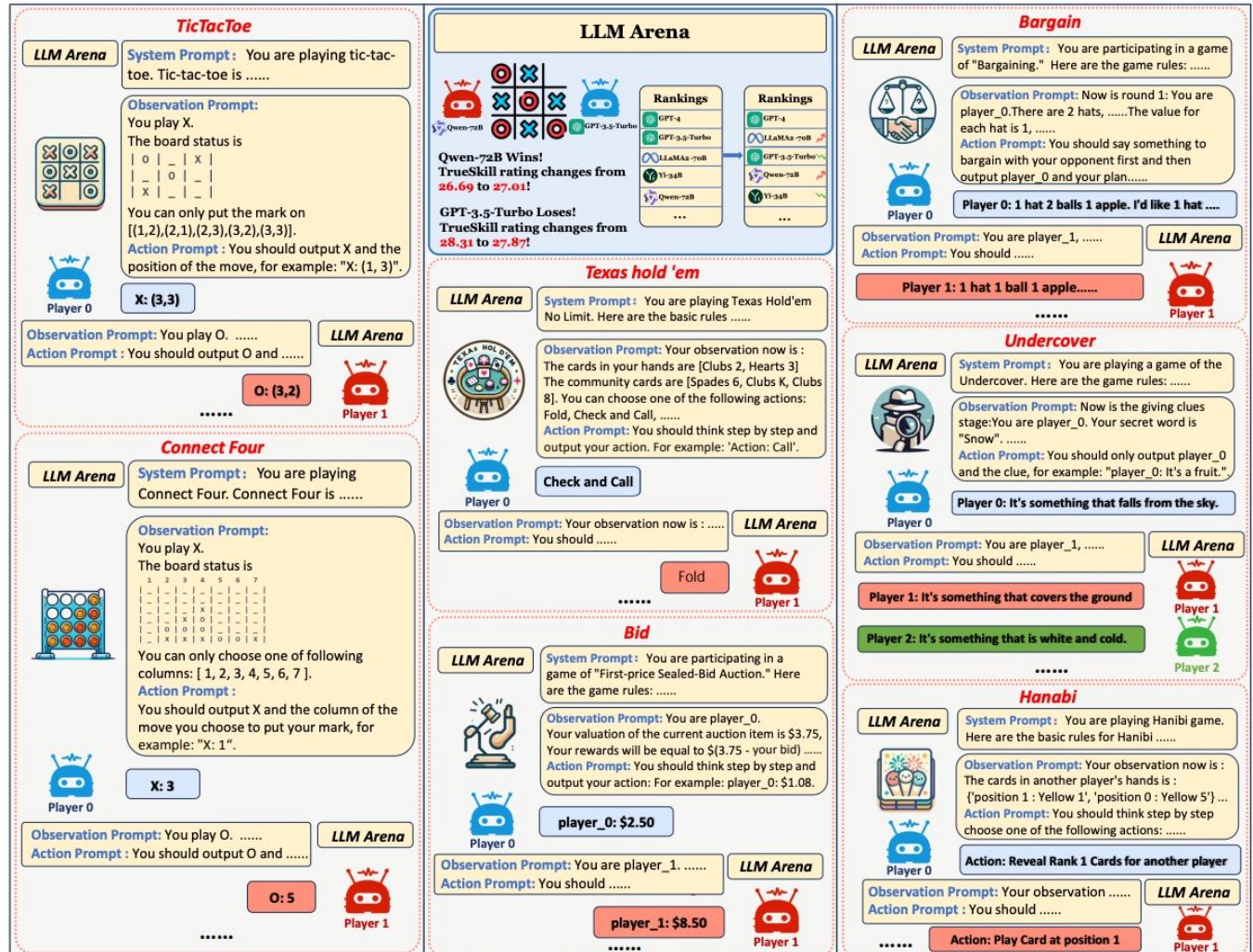


Figure 1: Examples of seven different game environments in LLMArena.

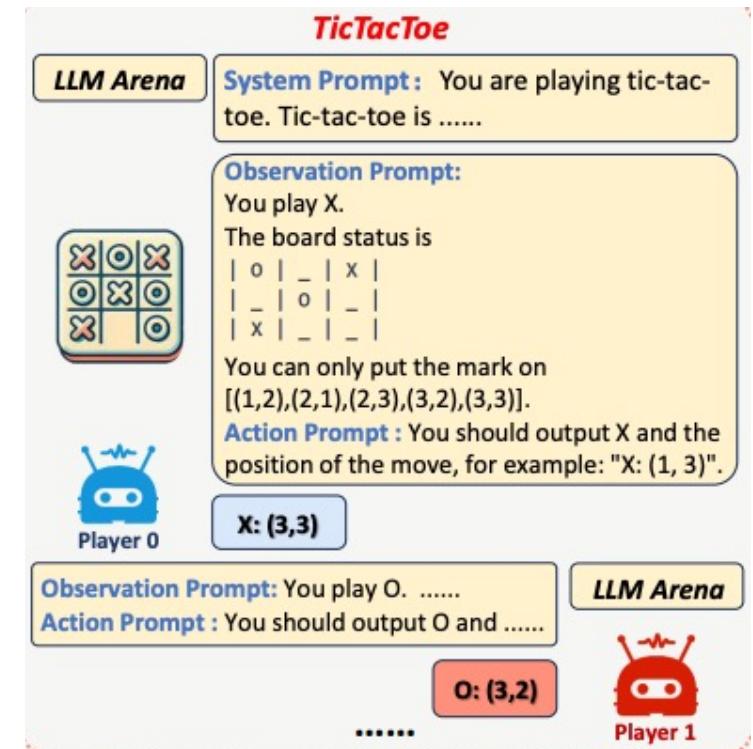
# Prompt design

- **System Prompt.** System prompts are special messages used to steer the behavior of LLM. They allow to prescribe the agent's style and task within certain bounds, making it more customizable and adaptable for various use cases.

In LLM Arena, we stipulate the role of the agent and the rules of the game in the system prompt.

- **Observation Prompt.** Observation Prompt provides the necessary state information for the agent to interact with the environment, including the opponent's actions, current game status, legal actions, etc. We use {} to frame these changing information.

- **Action Prompt.** Action Prompt guides the Agent to choose from legal actions and output regularized actions. In addition, to stimulate the agent's reasoning ability, in some environments, we also add CoT Prompt.



# Experiments and results

- As the scale of the model parameters increases, there is a noticeable improvement in the capabilities of LLMs.
- However, exceptions can arise in specific environments.
- Significant disparity between GPT-4 and other models. GPT-4 achieved SOTA across all evaluated tasks.

LLMs	TicTacToe	ConnectFour	Texas Hold'em	Bid	Bargain	Undercover	Hanabi	Average
GPT-4	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
GPT-3.5-Turbo	82.81	96.30	81.23	84.17	99.26	92.59	84.13	88.64
Qwen-72B-Chat	90.08	80.30	94.88	84.42	92.95	62.96	89.01	84.94
Llama-2-70B	82.21	95.90	98.42	94.22	88.63	85.19	80.35	89.27
Agentlm-70B	81.44	83.62	77.35	86.10	98.33	92.59	79.58	85.57
DeepSeek-67B-Chat	68.00	88.88	96.64	20.30	89.02	74.07	68.69	72.23
SUS-Chat-34B	79.75	96.30	84.38	66.17	74.53	88.89	86.33	82.34
Yi-34B-Chat	78.23	95.36	86.06	88.77	74.18	96.30	34.30	79.03
Qwen-14B-Chat	86.40	72.67	90.56	86.82	91.13	88.89	71.70	84.02
WizardLM-13B	79.14	83.35	74.26	35.28	91.28	48.15	54.23	66.53
AgentLM-13B	78.89	89.23	86.85	30.39	94.93	62.96	22.09	66.48
DeepSeek-7B-Chat	81.17	73.06	96.10	14.53	92.56	74.07	0.00	61.64
AgentLM-7B	68.29	90.35	85.14	0.04	83.53	66.67	98.20	70.32
Vicuna-7B	64.95	80.12	90.46	68.94	86.33	62.96	87.37	77.30
Average	80.10	87.53	88.74	61.44	89.76	78.31	68.28	79.16

Table 1: Normalized scores of 14 different LLMs in 7 environments.



# AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents\*

Autonomous agents that address **day-to-day digital tasks** (e.g., ordering groceries for a household), must not only operate **multiple apps** (e.g., notes, messaging, shopping app) via APIs, but also **generate rich code with complex control flow** in an iterative manner **based on their interaction with the environment**.

## AppWorld

=

**AppWorld Engine** is a high-quality **execution environment** of 9 day-to-day apps operable via 457 APIs and populated with realistic digital activities simulating the lives of ~100 fictitious users.

+

**AppWorld Benchmark** is a suite of 750 autonomous agent tasks requiring rich and interactive code generation.

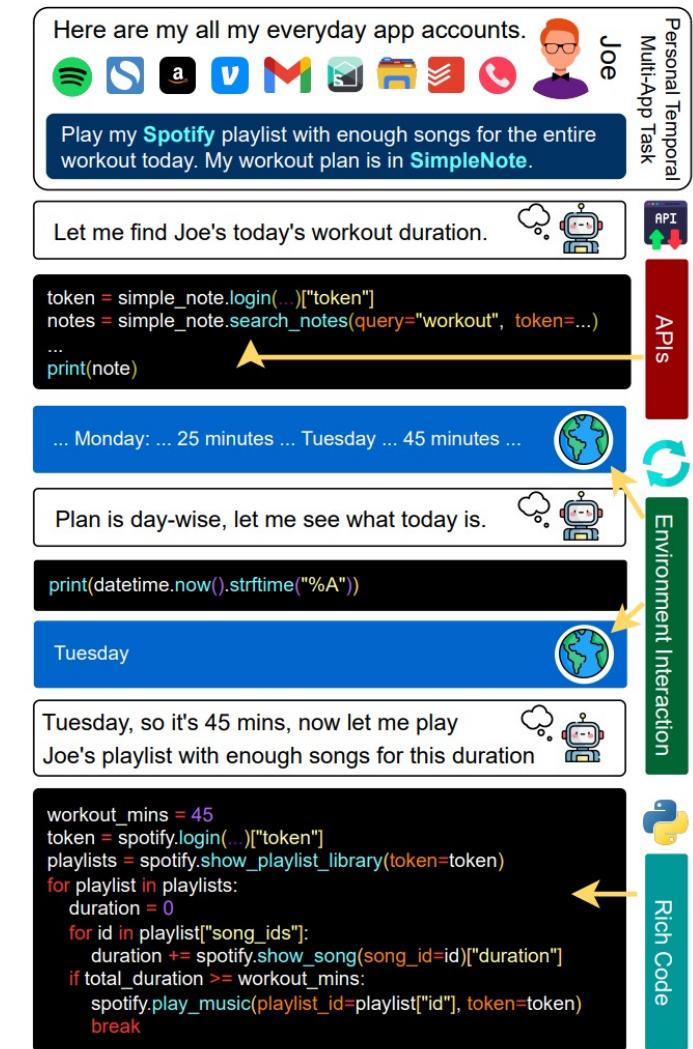


Figure 1: A (shortened) day-to-day task from AppWorld Benchmark requiring **rich code** with **environment interaction** and **API calls**.<sup>2</sup>

# AppWorld =

**AppWorld Engine** is a high-quality simulator of real-world apps and people using these apps, controlled via APIs:

*a fully controllable, stable, and reproducible execution environment of simulated apps and people, where agents can operate apps via APIs without any real-world consequences (e.g., spamming emails) or resource usage (e.g., money for Venmo).*

+

**AppWorld Benchmark** provides complex tasks and their evaluation suites, built using AppWorld:

*750 complex tasks covering everyday scenarios using the apps in the AppWorld Engine. The tasks are natural, challenging, diverse, and carefully designed with distractors and hurdles so as to require thorough reasoning. Tasks cover multiple apps and rely on using many APIs in an intricate flow.*

+

## Programmatic and Robust Evaluation :

AppWorld Engine allows fine-grained control of the database underlying the apps

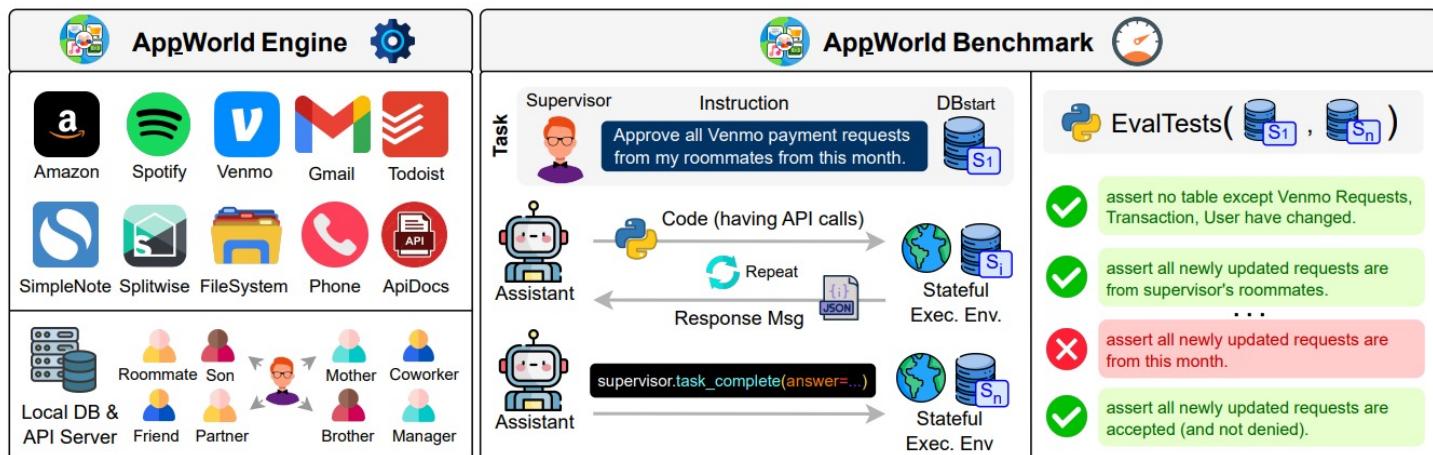


Figure 2: (Left) AppWorld Engine is a feature-rich API-based app simulator based on 9 real-world day-to-day apps populated with data simulating the digital activities of fictitious people living in this world. (Right) AppWorld Benchmark is a collection of complex everyday tasks requiring interactive coding with API calls, which come with reliable programmatic state-based evaluation suites robustly capturing goal completion criteria.

# Experiments and results

Base LLM	Method	Test-N	Test-C
		TGC   SGC	TGC   SGC
GPT4 <sup>O</sup>	ReAct	<b>48.8   32.1</b>	<b>30.2   13.0</b>
	PlanExec	44.6   23.2	19.7   7.9
	FullCodeRefl	33.9   26.8	19.2   12.2
	IPFunCall	32.1   16.1	18.0   10.1
GPT4 <sup>Trb</sup>	ReAct	26.8   12.5	<b>17.5   5.8</b>
	PlanExec	<b>32.7   16.1</b>	11.0   3.6
	FullCodeRefl	25.6   19.6	12.5   7.2
	IPFunCall	30.4   <b>21.4</b>	14.6   <b>9.3</b>
LLaMA3	ReAct	20.8   8.9	3.4   0.0
	PlanExec	8.9   1.8	2.4   0.7
	FullCodeRefl	<b>24.4   17.9</b>	<b>7.0   4.3</b>
DeepSeek	ReAct	7.1   1.8	2.9   0.7
	PlanExec	1.8   0.0	0.7   0.0
	FullCodeRefl	<b>13.1   8.9</b>	<b>5.8   2.9</b>
Mistral-7B	CodeAct	0.0   0.0	0.0   0.0
LLaMA	ToolLLaMA	0.0   0.0	0.0   0.0

Table 3: Main Results: Task and scenario goal completion scores (TGC | SGC), demonstrating that AppWorld Benchmark is highly challenging for current models.

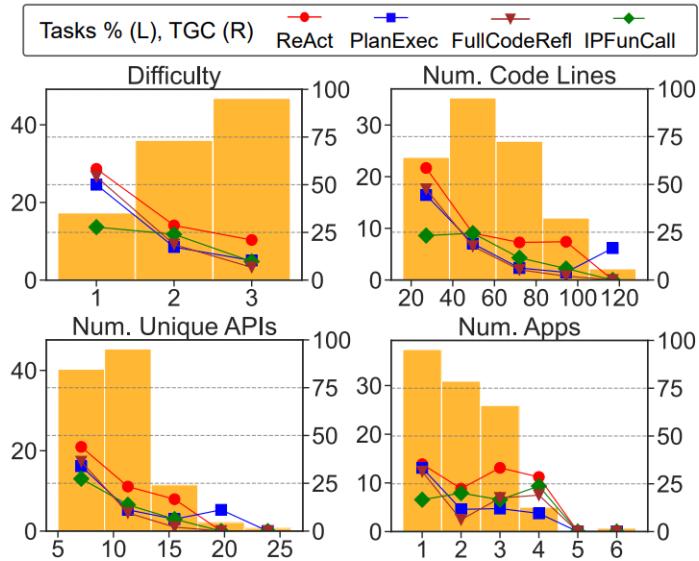
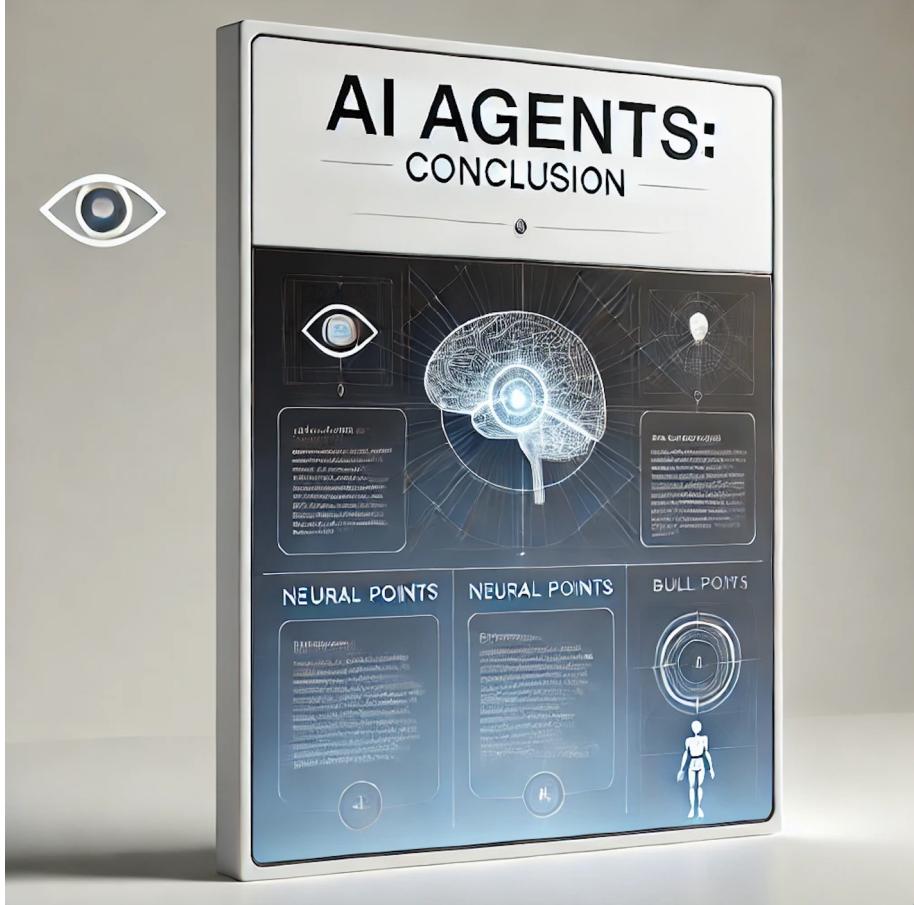


Figure 4: Percentage of Test-C tasks for hardness indicators (left axis) and TGC scores (right axis) for GPT4<sup>O</sup>. All methods perform worse as difficulty increases.

All methods perform substantially worse as the difficulty increases.



# AI agents: conclusion

# Conclusion

- New era of AI agents based on LLMs.
- A promising approach for multi-component, complex systems and tasks that a single model cannot handle
- Possibility of real-time actualization of knowledge without additional training.
- Tools and reasoning abilities for complex problems.
- ReAct as a standard LLM-agent paradigm.