



香港中文大學(深圳)  
The Chinese University of Hong Kong, Shenzhen

# **CSC6052/5051/4100/DDA6307/ MDS5110 Natural Language Processing**

Lecture 7-1: LLM Agents

Spring 2025  
Benyou Wang  
School of Data Science

# Manus



<https://manus.im/>

## Everyone should be get used to interact with Agents

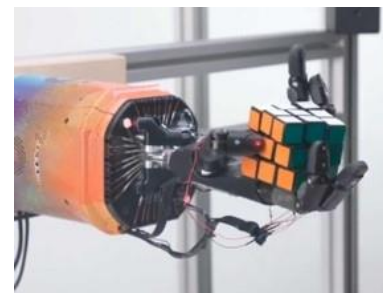
- In silicon valley, most companies have up to 30 employees, because AI improves coding efficiency.
- In silicon valley, they tend to hire new graduate than ~~experienced software engineers~~, since the formers are happy to use APIs
- A new guy is not qualified if he/she cannot solve problems with the help of AI (like cursor). Everyone could be full-stack engineers.
- work with AI and spend more time to define the applications with larger vision that benefits users.

# Contents

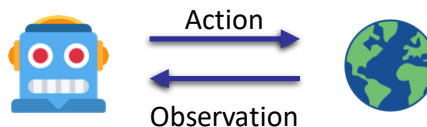
- **Overall framework of agents**
- **Four elements**
  - Planning
  - Tools
  - Memory
  - Action
- **LLM agent categories**

# **The framework of agents**

# What is “Agent”?



# 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!
  - Exercise question: how would you define **“intelligent”**?

# What is LLM Agents



## Explorations of Visual-Language Model on Autonomous Driving

### Sec. 2.1 Traffic signs understanding

#### Prompt:

This is a front-camera view of a driving car. Please identify all the traffic signs in the image, and explain each of them.



#### GPT-4V:

There are three traffic signs in the image:

A sign under **the overhead structure displaying "4.5m"**: This sign indicates the maximum height of vehicles that can safely pass underneath the structure. Vehicles taller than 4.5 meters should not proceed.

**A triangular sign with a curved arrow pointing left**: This is a warning sign that indicates there is a left bend or curve ahead. Drivers should be cautious and possibly reduce their speed.

**A rectangular sign with the word "SLOW"**: This sign instructs drivers to reduce their speed, typically because of potential hazards or a condition ahead that requires slower driving.

#### Prompt:

This is a front-camera view of a driving car. Please identify all the traffic signs in the image, and explain each of them.



#### GPT-4V:

There are three traffic signs displayed:

**60**: This is a speed limit sign, indicating that the maximum allowed speed on this road or section of the road is 60 kilometers per hour (km/h).

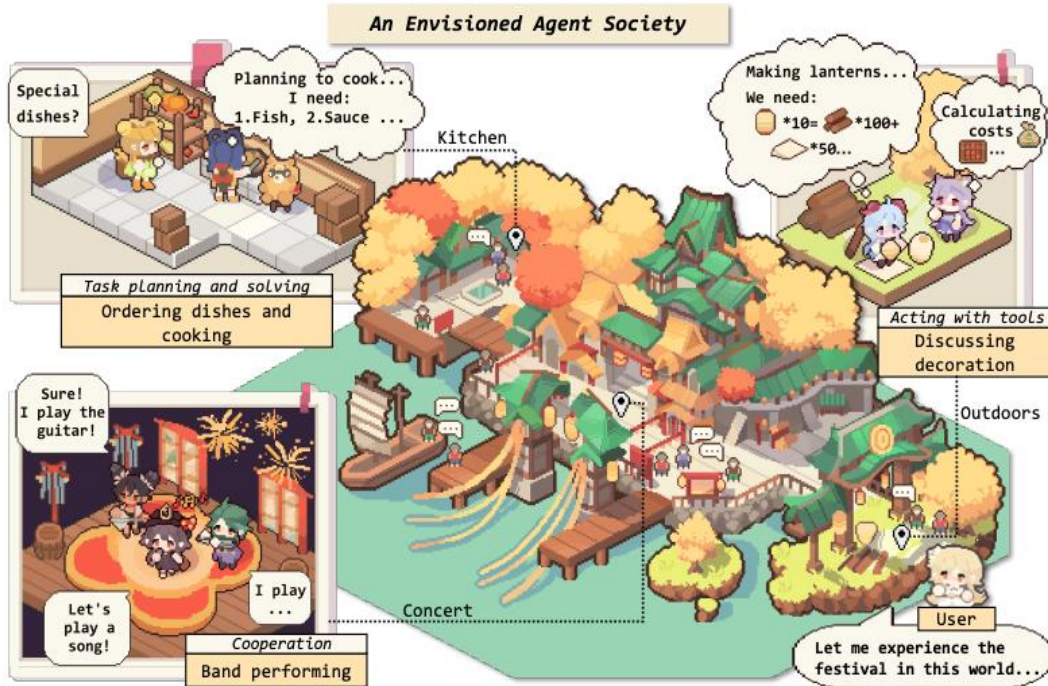
**Truck with a slash through it**: This is a no trucks or no heavy vehicles sign, indicating that trucks or heavy vehicles are not permitted on this road or section of the road.

**Car with a horn and a slash through it**: This signifies a no honking or horn prohibited zone. Drivers are not allowed to use their vehicle's horn in the area where this sign is displayed.

Figure 6: Results on the ability to understand the traffic signs. **Green** highlights the right answer in understanding, **Red** highlights the wrong answer in understanding. Check Section 2.1 for detailed discussions.



# What is LLM Agents



Scenario of an envisioned society composed of AI agents

In the **kitchen**, one agent **orders dishes**, while another agent is responsible for **planning and solving the cooking task**.

At the **concert**, three agents are collaborating to **perform in a band**.

**Outdoors**, two agents are **discussing lantern-making, planning the required materials, and finances by selecting and using tools**. Users can participate in any of these stages of this social activity

# What is LLM Agents

```
> python scripts/main.py
Welcome back! Would you like me to return to being BlogAI?
Continue with the last settings?
Name: BlogAI
Role: an Ai designed to autonomously create a blog post in Ge
Goals: ['research the topic thoroughly', 'write an article tha
st practice examples', "make the blog post interesting by link
he image", 'save article in file as markdown']
Continue (y/n): n
Welcome to Auto-GPT! Enter the name of your AI and its role b
Name your AI: For example, 'Entrepreneur-GPT'
AI Name: NewsAI
NewsAI here! I am at your service.
Describe your AI's role: For example, 'an AI designed to auto
NewsAI is: an AI designed to write news articles
Enter up to 5 goals for your AI: For example: Increase net wo
Enter nothing to load defaults, enter nothing when finished.
Goal 1: find an interesting news topic that includes "AI" and
Goal 2: find an interesting and unusual angle to the topic
Goal 3: write an article from that unusual angle
Goal 4: save the article in a file, in markdown format
Goal 5:
Using memory of type: LocalCache
█ Thinking...
```

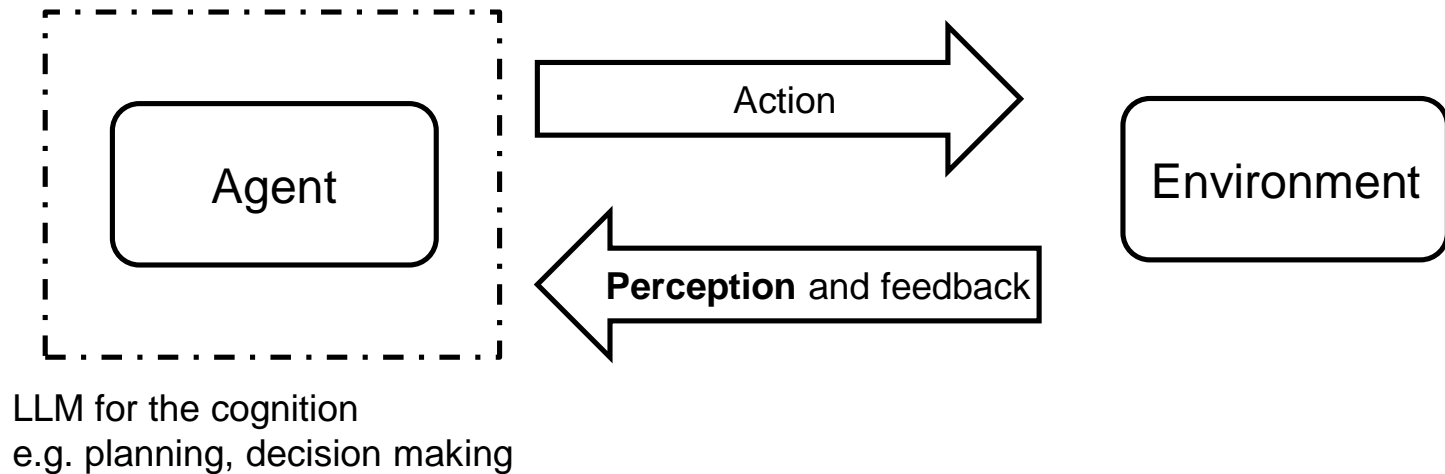
Let an LLM **decide what to do over and over**, while feeding the results of its actions back into the prompt. This allows the program to iteratively and incrementally work towards its objective.

In most cases, prompting engineering is enough for agent solution. (**not training needed**)

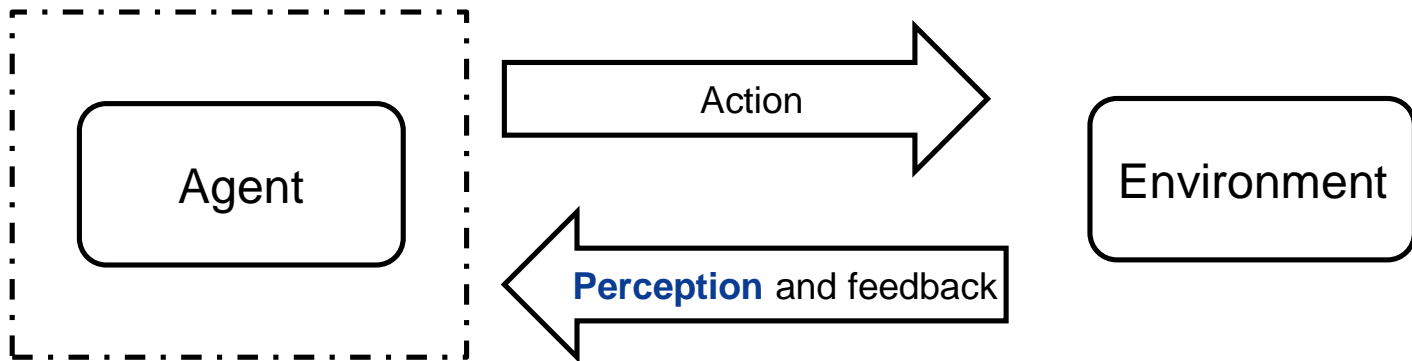
Complete Guide To Setup AutoGPT

<https://docs.agpt.co/>

# A high-level picture



# A high-level picture

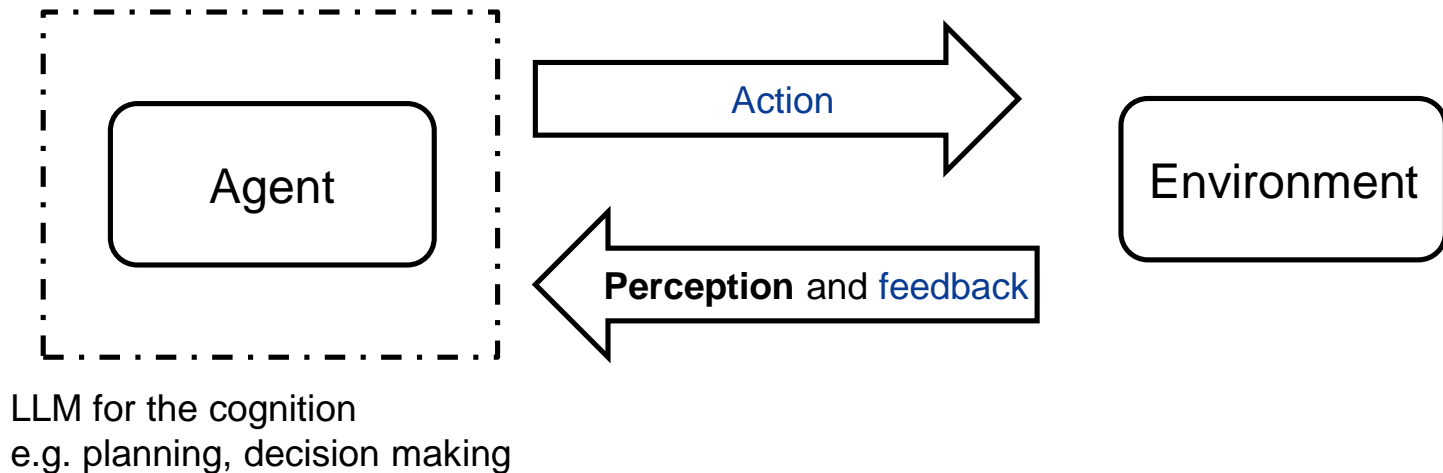


LLM for the cognition  
e.g. planning, decision making



**Perception**

# A high-level picture

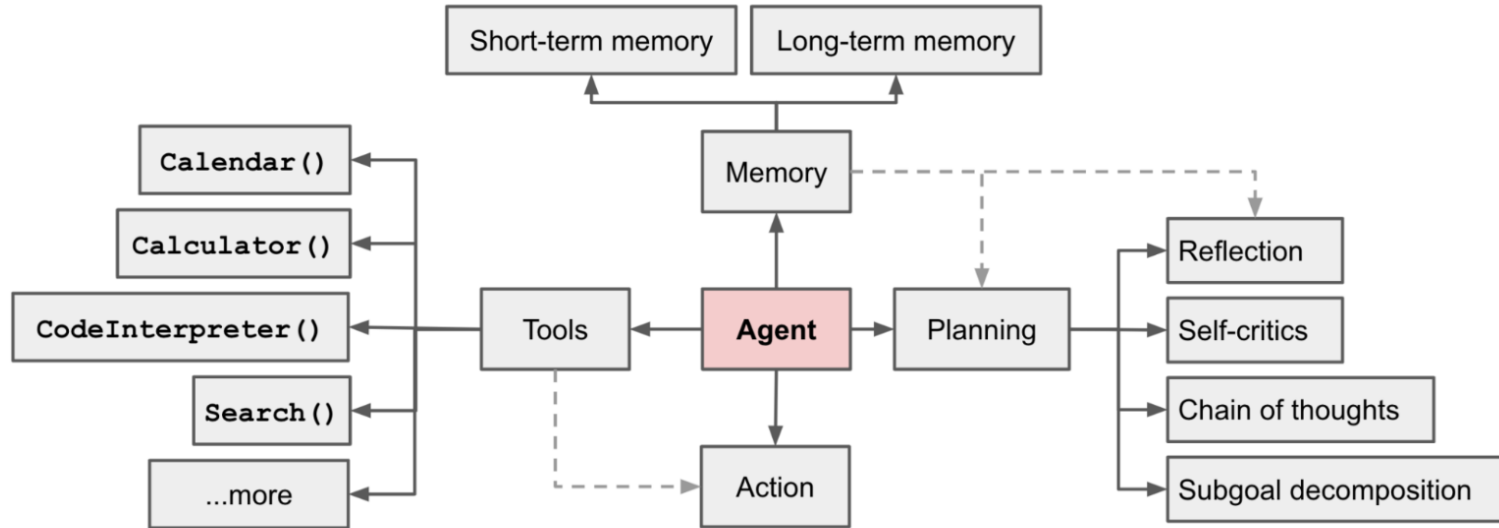


**Action and feedback helps evolution of LLM agents**

# **The four elements of LLM agents**

# LLM Agents

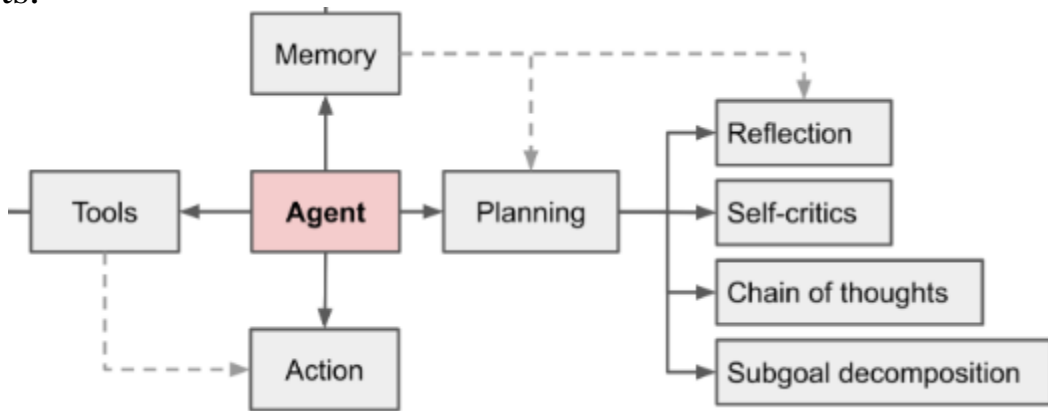
AI entities crafted around the **fundamental component** of large language models. Their prowess lies in understanding and creating human-like language, enabling a multitude of applications.



# What is LLM Agents

## Planning (multi-steps decision making):

- **Subgoal decomposition:** The agent breaks down large tasks into smaller, manageable subgoals, enabling efficient handling of complex tasks.
- **Reflection and refinement:** The agent can do self-criticism and self-reflection over past actions, learn from mistakes and refine them for future steps, thereby improving the quality of final results.

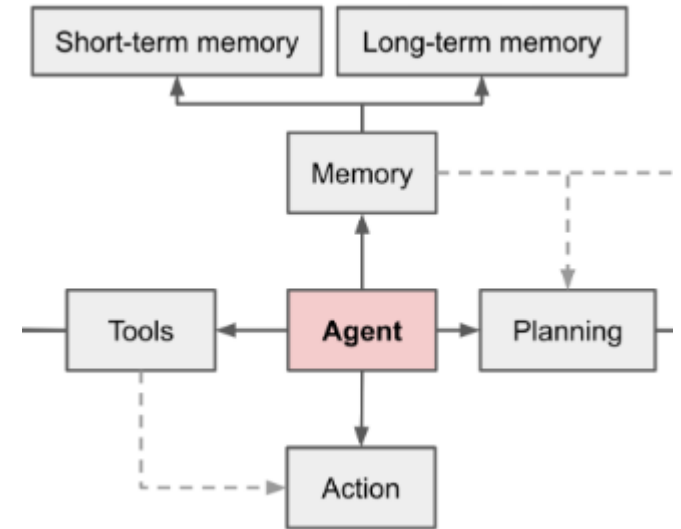




# What is LLM Agents

## Memory:

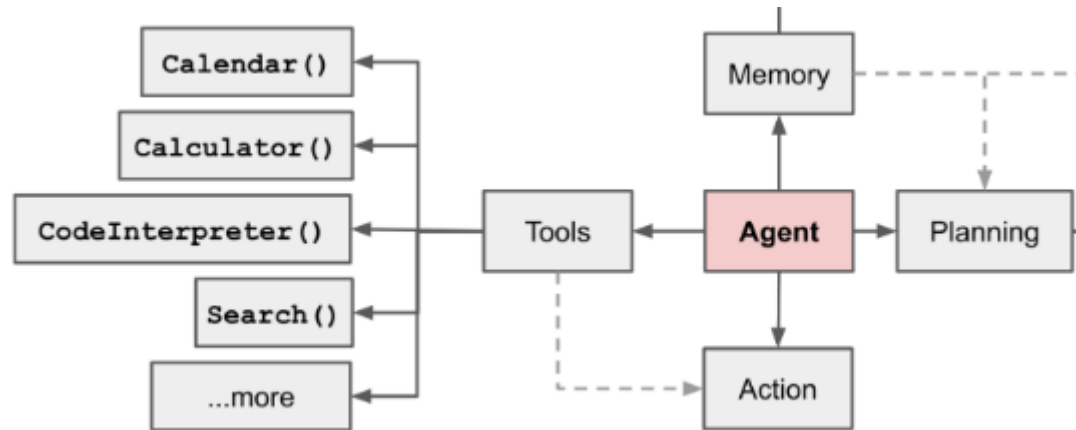
- **Short-term memory:** all the **in-context learning** is utilizing short-term memory of the model to learn.
- **Long-term memory:** this provides the agent with the capability to retain and recall (infinite) information over extended periods, often by leveraging an **external vector store** and fast retrieval.



# What is LLM Agents

## Tool use:

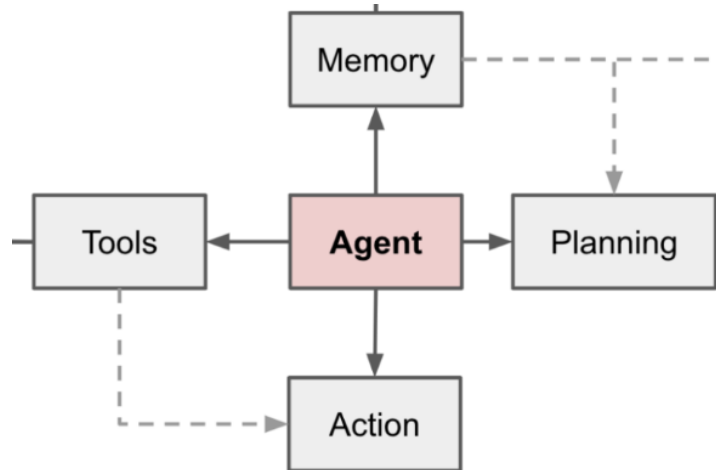
- The agent learns to call **external APIs** for extra information that is missing from the model weights (often hard to change after pre-training), including current information, code execution capability, access to proprietary information sources and more.



# What is LLM Agents

## Action:

- The agent's ability to execute actions in the real or virtual world is crucial. This can range from performing tasks in a digital environment to controlling physical robots or devices. The execution phase relies on the agent's planning, memory, and tool use to carry out tasks effectively and adaptively.



# Why LLM Agents stand out?

- **Language Mastery:** Their inherent capability to both comprehend and produce language ensures seamless user interaction.
- **Decision-making:** LLMs are equipped to reason and decide, making them adept at solving intricate issues.
- **Flexibility:** Their adaptability ensures they can be molded for diverse applications.
- **Collaborative Interactions:** They can collaborate with humans or other agents, paving the way for multifaceted interactions.



	LLM
	LLM + Planning
	LLM + Planning + Memory
	Agents (LLM + Planning + Memory + Tools)

# **Element 1: Planning**

# What is planning

## How to solve a complicated task **sequentially**?

### **One-step** task: translate an paragraph

- simple
- usually without interaction



You

translate "I think therefore I am"



ChatGPT

"I think therefore I am" translates to "我思故我在" in Chinese.

### **Multi-step** task: How to put an elephant into a fridge?

- complicated
- it involves multiple steps
- it could use **external tools or do real-world actions (e.g., operate the fridge)**



You

How to put an elephant into a fridge?



ChatGPT

The phrase "How to put an elephant into a fridge?" is often used as a humorous or rhetorical question, as it's a task that's obviously impractical and absurd in real life. This question is typically found in jokes or as a form of light-hearted riddle.

The usual joke answer to this is a three-step process:

1. Open the fridge.
2. Put the elephant in.
3. Close the fridge.

This kind of joke plays on the absurdity and simplicity of the solution to an apparently complex problem. It's a classic example of lateral thinking and humor in riddles.

# Two simple examples

- GSM8K (math word problem)

**Problem:** Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?

**Solution:** Beth bakes 4 2 dozen batches of cookies for a total of  $4*2 = \ll 4*2=8 \gg 8$  dozen cookies

There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of  $12*8 = \ll 12*8=96 \gg 96$  cookies

She splits the 96 cookies equally amongst 16 people so they each eat  $96/16 = \ll 96/16=6 \gg 6$  cookies

**Final Answer:** 6

- GAME24

Input: 4 4 6 8

Steps:

$4 + 8 = 12$  (left: 4 6 12)

$6 - 4 = 2$  (left: 2 12)

$2 * 12 = 24$  (left: 24)

Answer:  $(6 - 4) * (4 + 8) = 24$

They are both multi-step problems!

# Examples of Planning

**Task Decomposition**

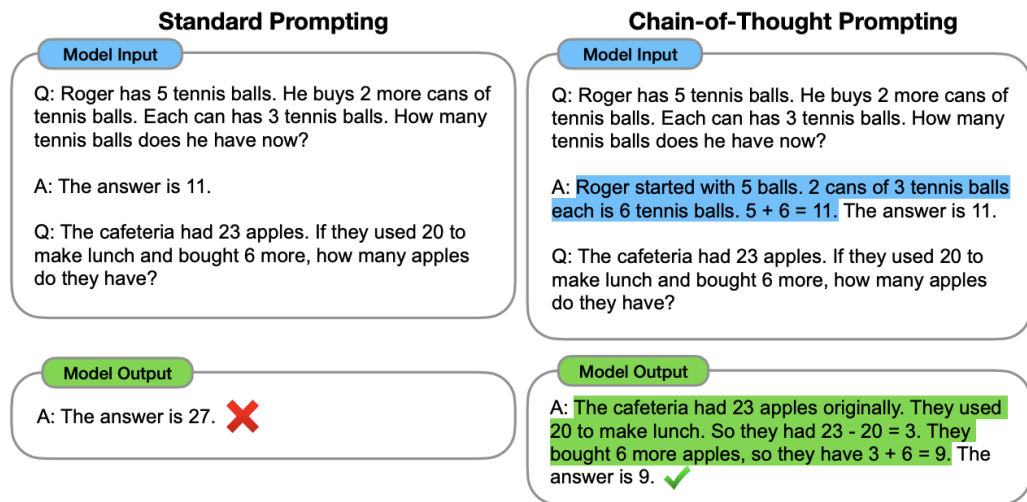
**Self-Reflection/self-refinement**



# Planning with Task Decomposition

# Task Decomposition: Chain of thought

**Chain of Thought (CoT)** has become a standard prompting technique for enhancing model performance on complex tasks. The model is instructed to “*think step by step*” to utilize more test-time computation to **decompose hard tasks into smaller and simpler steps**. CoT transforms big tasks into multiple manageable tasks and shed lights into an interpretation of the model’s thinking process.



[Chain-of-Thought Prompting Elicits Reasoning in Large Language Models](#)

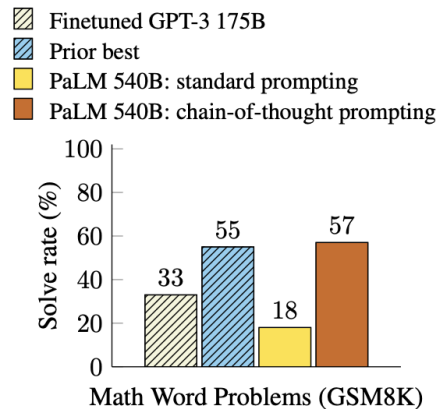


Figure 2: PaLM 540B uses chain-of-thought prompting to achieve new state-of-the-art performance on the GSM8K benchmark of math word problems. Finetuned GPT-3 and prior best are from Cobbe et al. (2021).

# Task Decomposition: Least-to-most prompting

Explicitly decompose into subquestions

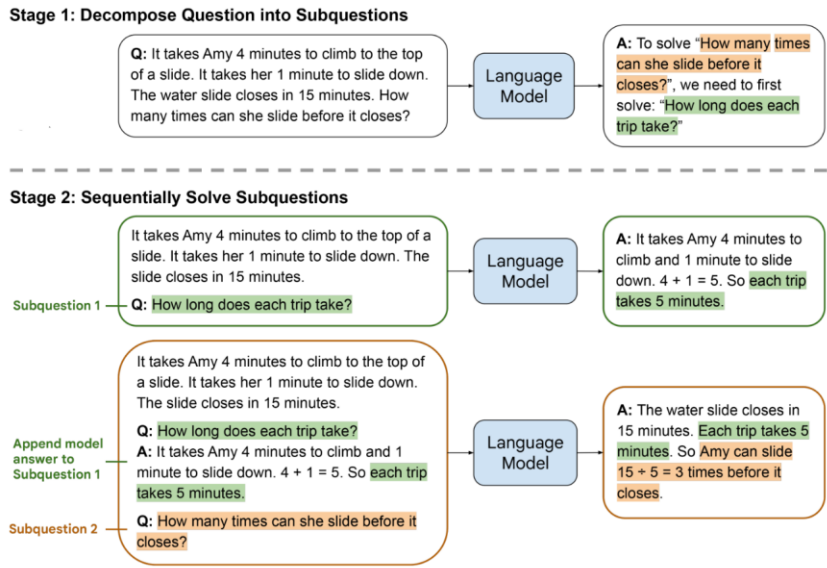
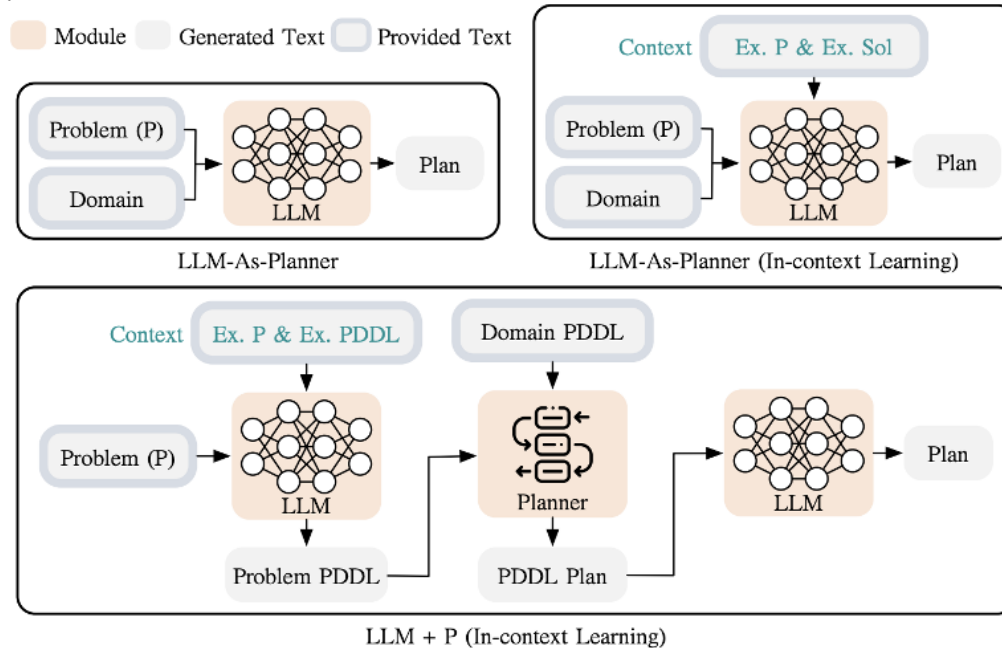


Figure 1: Least-to-most prompting solving a math word problem in two stages: (1) query the language model to decompose the problem into subproblems; (2) query the language model to sequentially solve the subproblems. The answer to the second subproblem is built on the answer to the first subproblem. The demonstration examples for each stage's prompt are omitted in this illustration.

# Task Decomposition: LLM+P

LLM+P involves relying on an external classical planner to do long-horizon planning. This approach utilizes the **Planning Domain Definition Language (PDDL)** as an intermediate interface to describe the planning problem.



# Task Decomposition: LLM+P

In the PDDL process, LLM

- 1) **translates** the problem into “Problem PDDL”;
- 2) requests a **classical planner** to generate a PDDL **plan** based on an existing “Domain PDDL”;
- 3) **translates** the PDDL plan **back** into natural language.

Essentially, the planning step is outsourced to an external tool, assuming the availability of domain-specific PDDL and a suitable planner.

An Example PDDL Problem File Written by GPT-4 with Context

**Context (shaded):**

```
An example planning problem is: You have 5 blocks.
b2 is on top of b5. b5 is on top of b1. b1 is on top of
b4. b3 is on top of b2. b4 is on the table. b3 is clear.
Your arm is empty. Your goal is to move the blocks.
b4 should be on top of b3. The problem PDDL is:
(:objects b1 b2 b3 b4 b5)
(:init (arm-empty) (on b1 b4) (on
b2 b5) (on b3 b2)
      (on-table b4) (on b5 b1)
(clear b3))
(:goal (and (on b4 b3))))
```

**Prompt:** Description of Problem (P1) + Context  
+ Provide me with the problem PDDL file that describes the planning problem directly without further explanations.

**GPT-4 (the generated problem PDDL):**

```
(:objects b1 b2 b3 b4 b5 )
(:init (arm-empty) (on-table b1)
(on b2 b1) (on b3 b4)
      (on b4 b2) (on b5 b3) (clear
b5)))
(:goal (and (on b1 b2) (on b3 b5)
(on b4 b1)))
```

**Planner:**

```
(unstack b5 b3) (putdown b5)
(unstack b3 b4) (stack b3 b5)
(unstack b4 b2) (putdown b4)
(unstack b2 b1) (putdown b2)
(pickup b1) (stack b1 b2) (pickup
b4) (stack b4 b1)
```

# Planning with Self-Reflection

吾日三省吾身



## Self-Reflection (反思)

**Self-reflection** is a vital aspect that allows autonomous agents to **improve iteratively by refining past action decisions and correcting previous mistakes**. It plays a crucial role in real-world tasks where trial and error are inevitable.

### Reflective Thinking



## Self-Reflection: **ReACT**

**ReACT** integrates reasoning and acting within LLM by extending the action space to be a combination of task-specific discrete actions and the language space. The former enables LLM to interact with the environment (e.g. use Wikipedia search API), while the latter prompting LLM to generate reasoning traces in natural language.

The ReAct prompt template incorporates explicit steps for LLM to think, roughly formatted as:

```
Thought: ...  
Action: ...  
Observation: ...  
... (Repeated many times)
```



# Self-Reflection: ReACT

## HotpotQA

### (1d) ReAct (Reason + Act)

**Thought 1:** I need to search Apple Remote and find the program it was originally designed to interact with.

**Act 1:** Search[Apple Remote]

**Obs 1:** The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the Front Row media center program ...

**Thought 2:** Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.

**Act 2:** Search[Front Row]

**Obs 2:** Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]

**Thought 3:** Front Row is not found. I need to search Front Row (software) .

**Act 3:** Search[Front Row (software)]

**Obs 3:** Front Row is a discontinued media center software ...

**Thought 4:** Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.

**Act 4:** Finish[keyboard function keys]



## AlfWorld Env

### (2b) ReAct (Reason + Act)

**Act 1:** Think[First I need to find a pepper 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 go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

**Act 6:** Go to countertop 3

**Obs 6:** On the countertop 3, you see a apple 1, a bread 1, ..... a pepper shaker 1, and a vase 3.

**Act 7:** Take pepper shaker 1 from countertop 3

**Obs 7:** You pick up the pepper shaker 1 from the countertop 3.

**Act 8:** Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1. ]

**Act 9:** Go to drawer 1

**Obs 9:** Drawer 1 is closed.

**Act 10:** Open drawer 1

**Obs 10:** You open Drawer 1 ...

**Act 11:** Put pepper shaker 1 in/on drawer 1

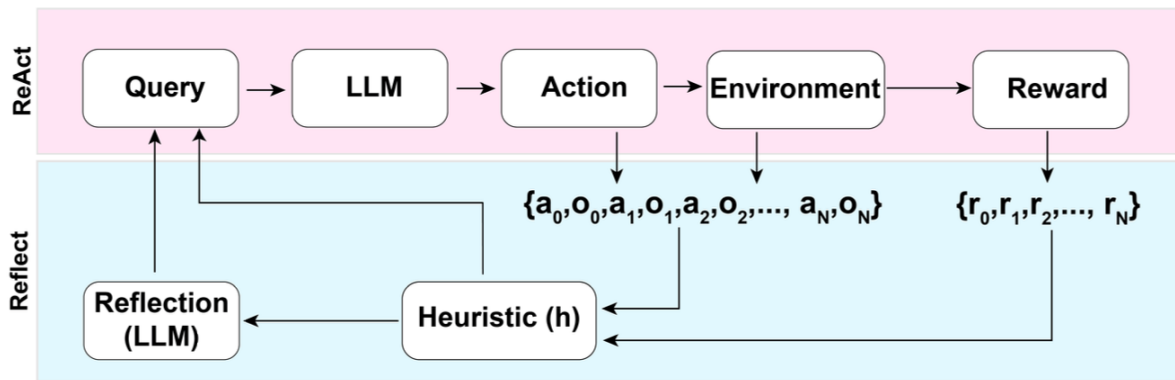
**Obs 11:** You put pepper shaker 1 in/on the drawer 1.



In both experiments on knowledge-intensive tasks and decision-making tasks, *ReAct* works better than the *Act*-only baseline where *Thought*: ... step is removed.

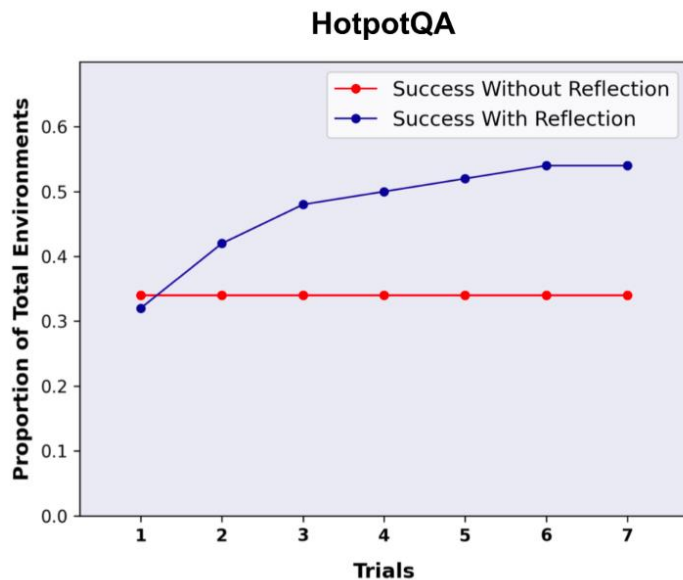
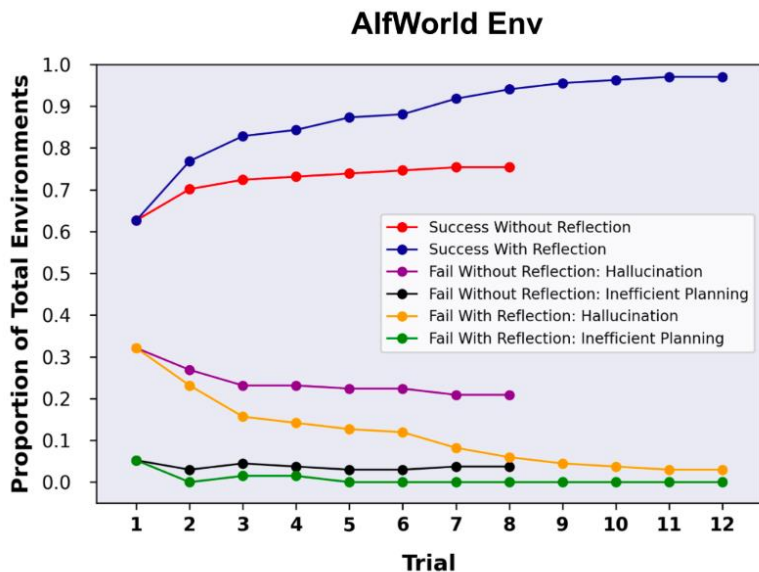
## Self-Reflection: Reflexion

**Reflexion** is a framework to equip agents with dynamic memory and self-reflection capabilities to improve reasoning skills. Reflexion has a standard RL setup, in which the **reward model provides a simple binary reward** and the action space follows the setup in ReAct where the task-specific action space is augmented with language to enable complex reasoning steps. After each action  $at$ , the agent computes a heuristic  $ht$  and optionally may decide to reset the environment to start a new trial depending on the self-reflection results.



## Self-Reflection: Reflexion

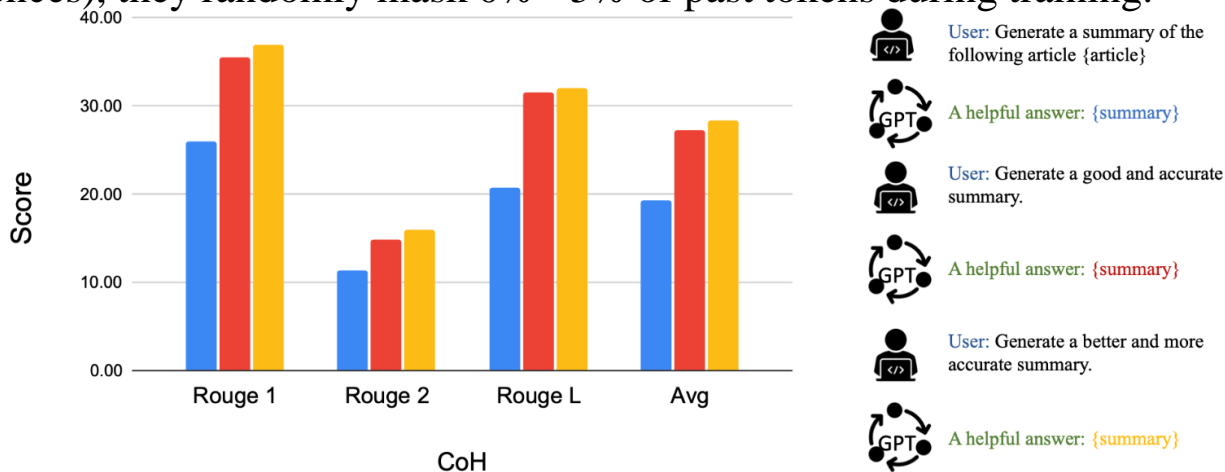
Self-reflection is created by showing two-shot examples to LLM and each example is a pair of (failed trajectory, ideal reflection for guiding future changes in the plan). Then reflections are added into the agent's working memory, up to three, to be used as context for querying LLM.



# Self-Reflection: Chain of Hindsight

**Chain of Hindsight (CoH)** encourages the model to improve on its own outputs by explicitly presenting it with a sequence of past outputs, each annotated with feedback.

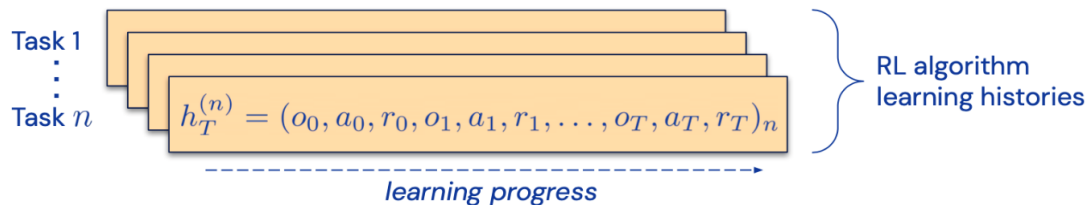
To avoid overfitting, CoH adds a regularization term to maximize the log-likelihood of the pre-training dataset. To avoid shortcutting and copying (because there are many common words in feedback sequences), they randomly mask 0% - 5% of past tokens during training.



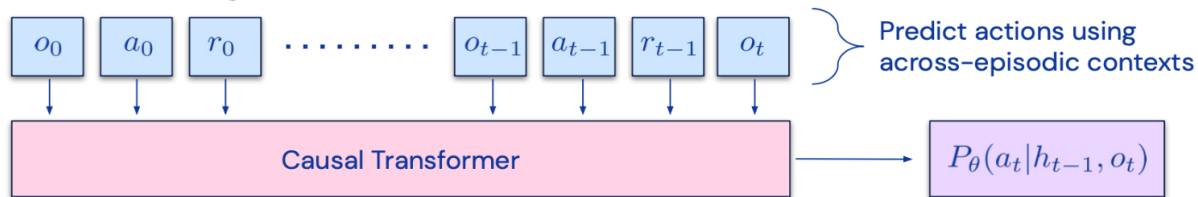
## Self-Reflection: Chain of Hindsight

The idea of CoH is to present a history of sequentially improved outputs in context and train the model to take on the trend to produce better outputs. [Algorithm Distillation](#) applies the same idea to cross-episode trajectories in reinforcement learning tasks, where an algorithm is encapsulated in a long history-conditioned policy. The goal is to learn the process of RL instead of training a task-specific policy itself.

### Data Generation



### Model Training



# **Element 2: tools**

Introduction to tools in LLMs

# Human + tool use: motivations

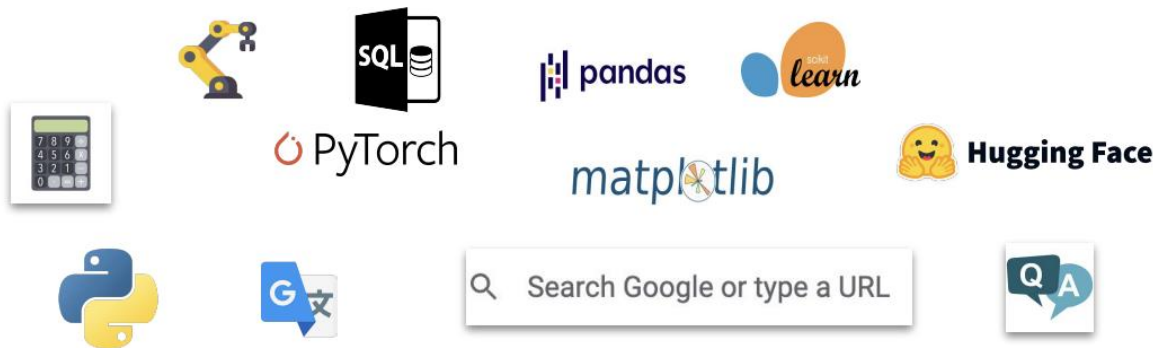
As humans, we have limited time and memory, feel tired, and have emotions.

- Human + tool use
  - Enhanced scalability
  - Improved consistency
  - Greater interpretability
  - Higher capacity and productivity



# LLMs + tool use: motivations

- Just like human, LLMs also benefits from tools
- LLMs + tool use
  - Enhanced scalability
  - Improved consistency
  - Greater interpretability
  - Higher capacity and productivity

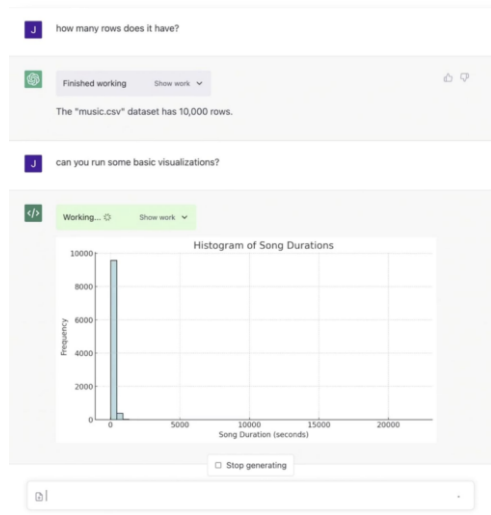




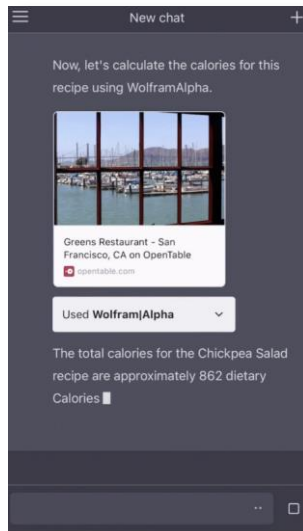
# LLMs + tool use in perspective of **executable language grounding**

Ground language models into **executable actions**

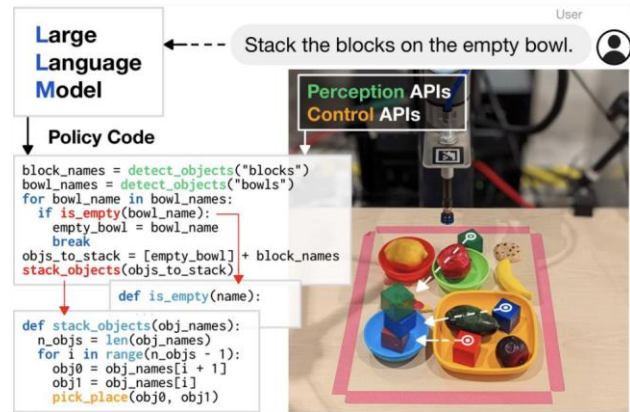
- Mapping natural language instructions into code or actions executable within various environments such as databases, web applications, and robotic physical world.
- LM (planning and reasoning) + **actions**



Data analysis



Web/Apps



Robotic physical world

<https://openai.com/blog/chatgpt-plugins>

<https://code-as-policies.github.io/>

# LLMs + tool use in perspective of **executable language grounding**

LLMs + tool use in executable language grounding tasks

## Inputs

- **Language:** user question/request
- **Toolkit:** code, APIs to search engines, self-defined functions, expert models...
- **Environment:** databases, IDE, web/apps, visual and robotic physical world...

## Outputs

- Grounded reasoning code/action seq that can be executed in the corresponding environment
  - What tools to select, when and how to use the selected tools

**Element 2: tools**  
Representative Works

**LLMs + tool** use in perspective of **executable language grounding**

## **Binding Language Models in Symbolic Languages**

Binder is a training-free neural-symbolic framework that maps the task input to an **executable Binder program** that

- (1) allows binding API calls to GPT-3 Codex into SQL/Python,
- (2) is executed with SQL/Python Interpreter + GPT-3 Codex to derive the answer.

# Binding Language Models in Symbolic Languages

Zhoujun Cheng<sup>\*1,2</sup>, Tianbao Xie<sup>\*1</sup>, Peng Shi<sup>5</sup>, Chengzu Li<sup>1</sup>, Rahul Nadkarni<sup>3</sup>, Yushi Hu<sup>3</sup>, Caiming Xiong<sup>6</sup>,  
Dragomir Radev<sup>7</sup>, Mari Ostendorf<sup>3</sup>, Luke Zettlemoyer<sup>3,8</sup>, Noah A. Smith<sup>3,4</sup>, Tao Yu<sup>1,3</sup>

<sup>1</sup>The University of Hong Kong, <sup>2</sup>Shanghai Jiao Tong University, <sup>3</sup>University of Washington,

<sup>4</sup>Allen Institute for AI, <sup>5</sup>University of Waterloo, <sup>6</sup>Salesforce Research, <sup>7</sup>Yale University, <sup>8</sup>Meta AI

> Paper

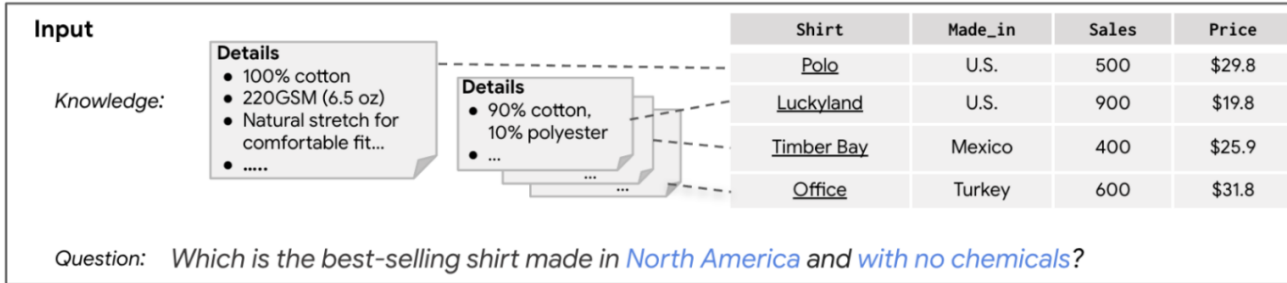
Code

Demo

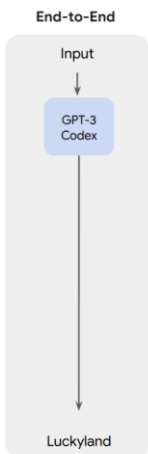
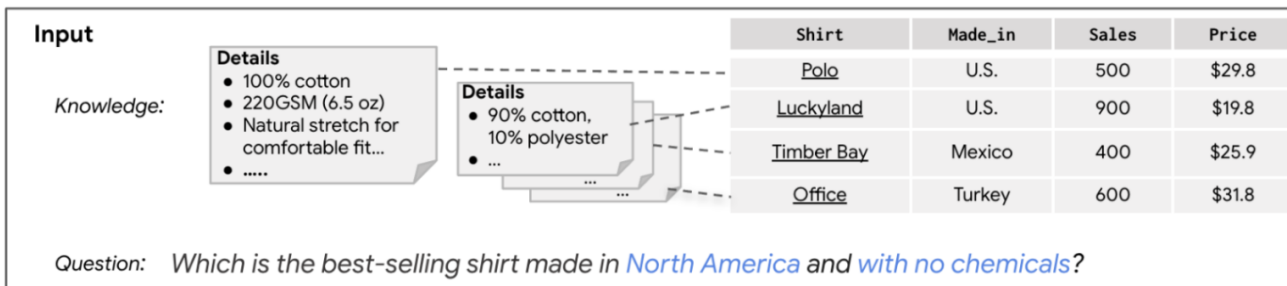
Twitter

Video

# LLMs + tool: Binding Language Models in Symbolic Languages



# LLMs + tool: Binding Language Models in Symbolic Languages



**Pros**

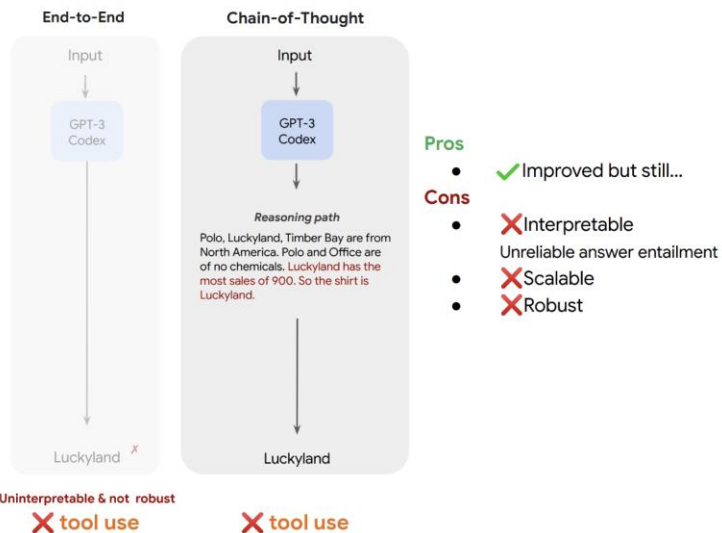
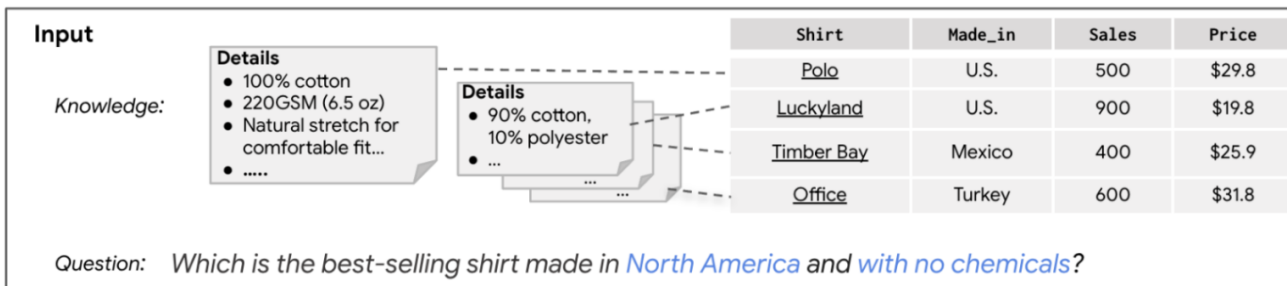
- ✓ General

**Cons**

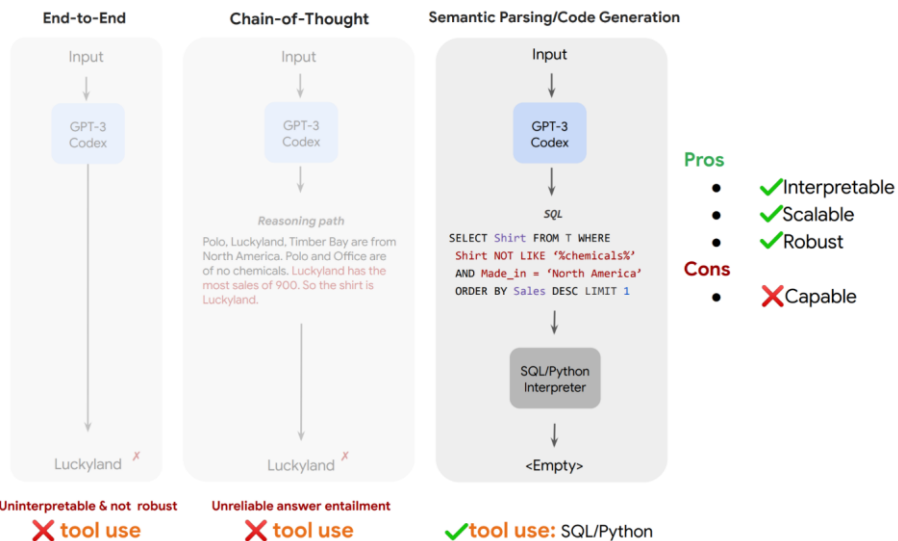
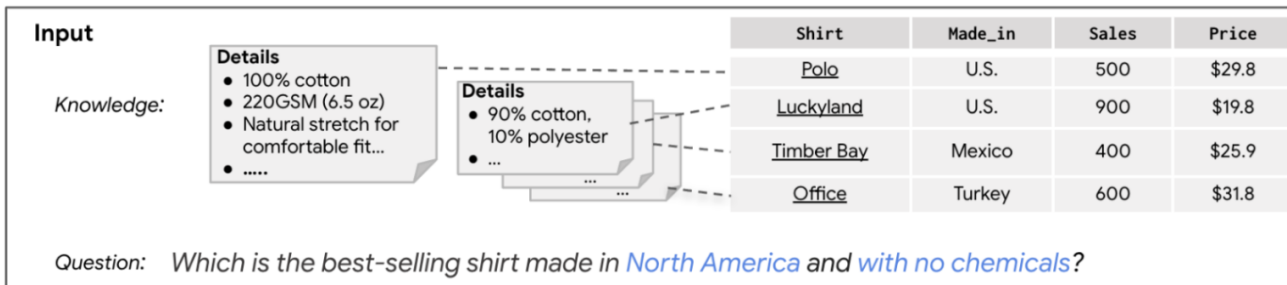
- ✗ Interpretable
- ✗ Scalable
- ✗ Robust

✗ tool use

# LLMs + tool: Binding Language Models in Symbolic Languages



# LLMs + tool: Binding Language Models in Symbolic Languages





# LLMs + tool: Binding Language Models in Symbolic Languages

**Input**

**Knowledge:**

**Details**

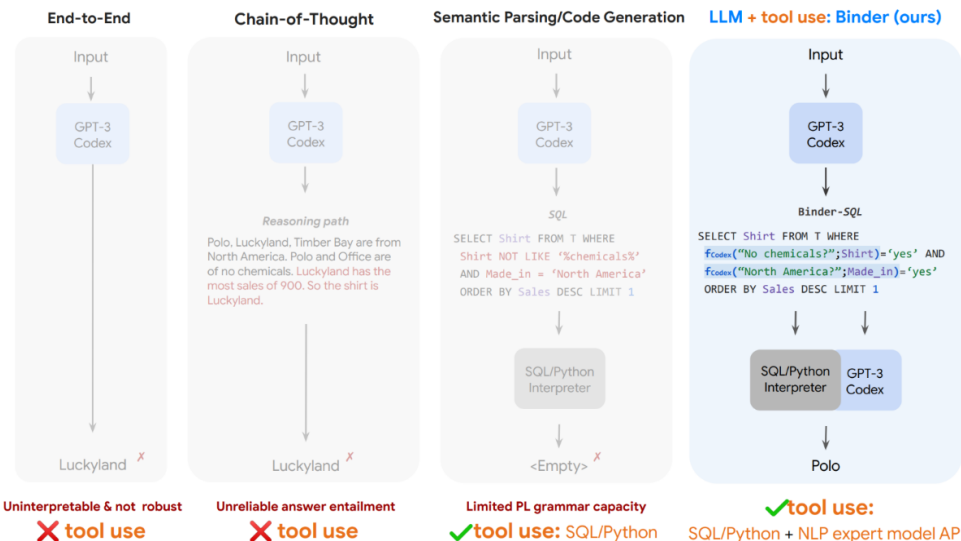
- 100% cotton
- 220GSM (6.5 oz)
- Natural stretch for comfortable fit...
- .....

**Details**

- 90% cotton, 10% polyester
- ...

Shirt	Made_in	Sales	Price
Polo	U.S.	500	\$29.8
<u>Luckyland</u>	U.S.	900	\$19.8
<u>Timber Bay</u>	Mexico	400	\$25.9
<u>Office</u>	Turkey	600	\$31.8

**Question:** Which is the best-selling shirt made in *North America* and *with no chemicals*?



## Why LLM + tool use?

Pros

- ✓ Interpretable
- ✓ Scalable
- ✓ Robust
- ✓ Capable

# LLMs + tool: PAL, PoT

## Chain-of-Thought (Wei et al., 2022)

Input

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

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

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold  $93 + 39 = 132$  loaves. The grocery store returned 6 loaves. So they had  $200 - 132 - 6 = 62$  loaves left.

The answer is 62.



## Program-aided Language models (this work)

Input

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

A: Roger started with 5 tennis balls.

tennis\_balls = 5

2 cans of 3 tennis balls each is

bought\_balls = 2 \* 3

tennis balls. The answer is

answer = tennis\_balls + bought\_balls

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves

loaves\_baked = 200

They sold 93 in the morning and 39 in the afternoon

loaves\_sold\_morning = 93

loaves\_sold\_afternoon = 39

The grocery store returned 6 loaves.

loaves\_returned = 6

The answer is

answer = loaves\_baked - loaves\_sold\_morning  
- loaves\_sold\_afternoon + loaves\_returned

```
>>> print(answer)
```

```
74
```



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

```
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

32,432,268,459



python

12,586,269,025



Question: Ketty saves 20000 dollars to the bank. After three years, the sum with compound interest rate is 1000 dollars more than the sum with simple interest rate. What is the interest rate of the bank?

Assuming the interest rate is  $x$ . The sum after two years with simple interest rate is  $20000 + x * 20000 * 3 = 20000 + 60000x$ . The sum after two years with compound interest rate is  $20000 * (1 + x)^3 = 200000 + 60000 * x + 60000x^2 + 20000x^3$ . The difference can be written as  $60000x^2 + 20000x^3 = 1000$ . In order to solve  $x$ , we can use the quadratic formula.  $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ , ...,  $x = \frac{-20000 \pm \sqrt{61600}}{120000}$ ,  $x = -0.051333$ .

CoT

```
interest_rate = Symbol('x')
sum_in_two_years_with_simple_interest= 20000 +
interest_rate * 20000 * 3
sum_in_two_years_with_compound_interest = 20000 * (1 +
interest_rate)**3
# Since compound interest is 1000 more than simple interest.
ans = solve(sum_after_in_yeras_with_compound_interest -
sum_after_two_years_in_compound_interest - 1000,
interest_rate)
```

PoT

-0.051333



python



$x = 0.24814$



[PAL: Program-aided Language Models](#)

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

# LLMs + webs/apps or personalized functions: ChatGPT-Plugins

## Function calling example

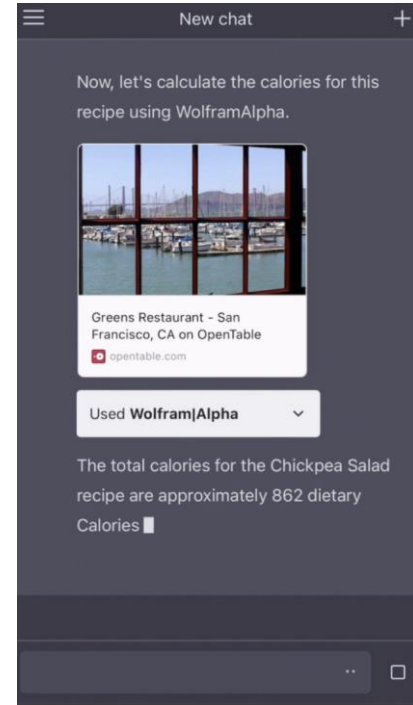
What's the weather like in Boston right now?

Step 1 - OpenAI API

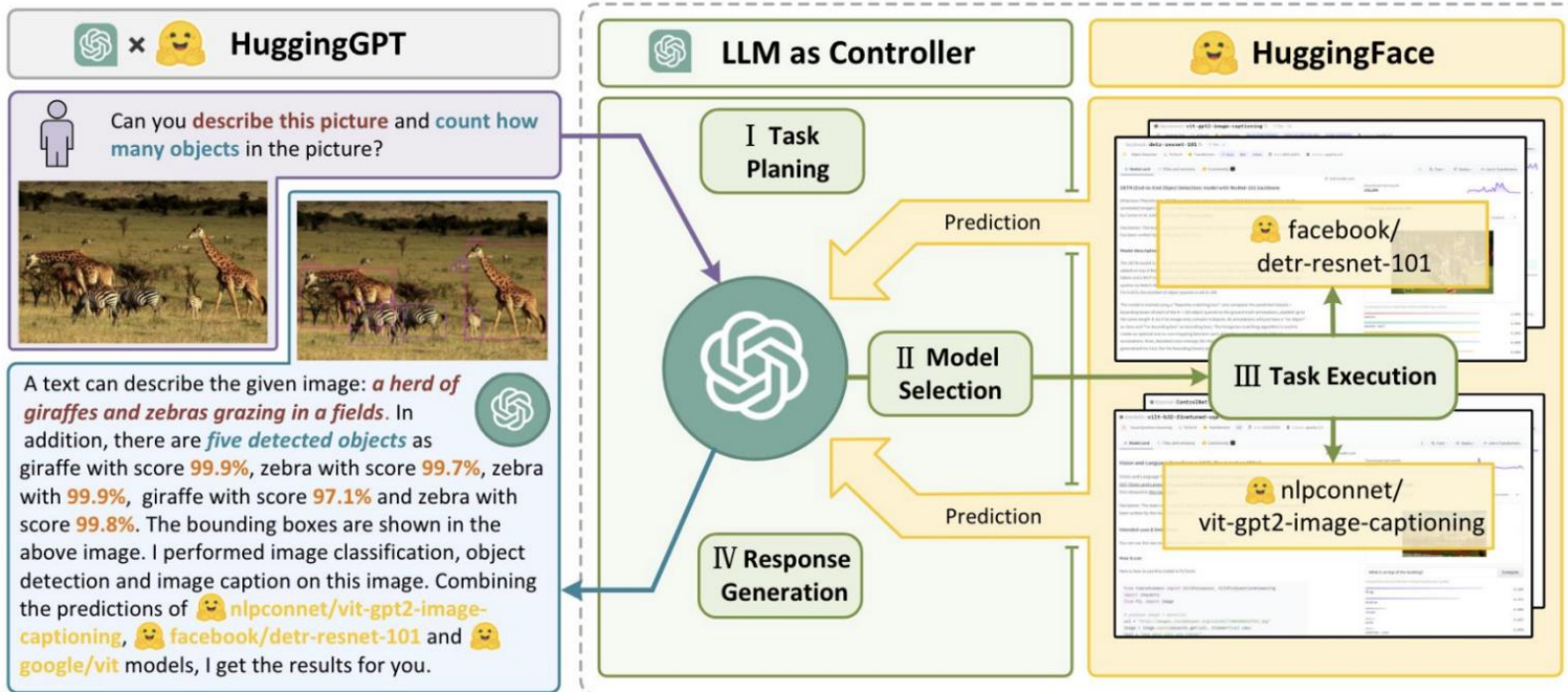
Call the model with functions and the user's input

Request Response

```
curl https://api.openai.com/v1/chat/completions -u :$OPENAI_API_KEY -H 'Content-Type: application/json' -d '{
  "model": "gpt-3.5-turbo-0613",
  "messages": [
    {"role": "user", "content": "What is the weather like in Boston?"}
  ],
  "functions": [
    {
      "name": "get_current_weather",
      "description": "Get the current weather in a given location",
      "parameters": {
        "type": "object",
        "properties": {
          "location": {
            "type": "string",
            "description": "The city and state, e.g. San Francisco, CA"
          },
          "unit": {
            "type": "string",
            "enum": ["celsius", "fahrenheit"]
          }
        }
      },
      "required": ["location"]
    }
  ]
}'
```

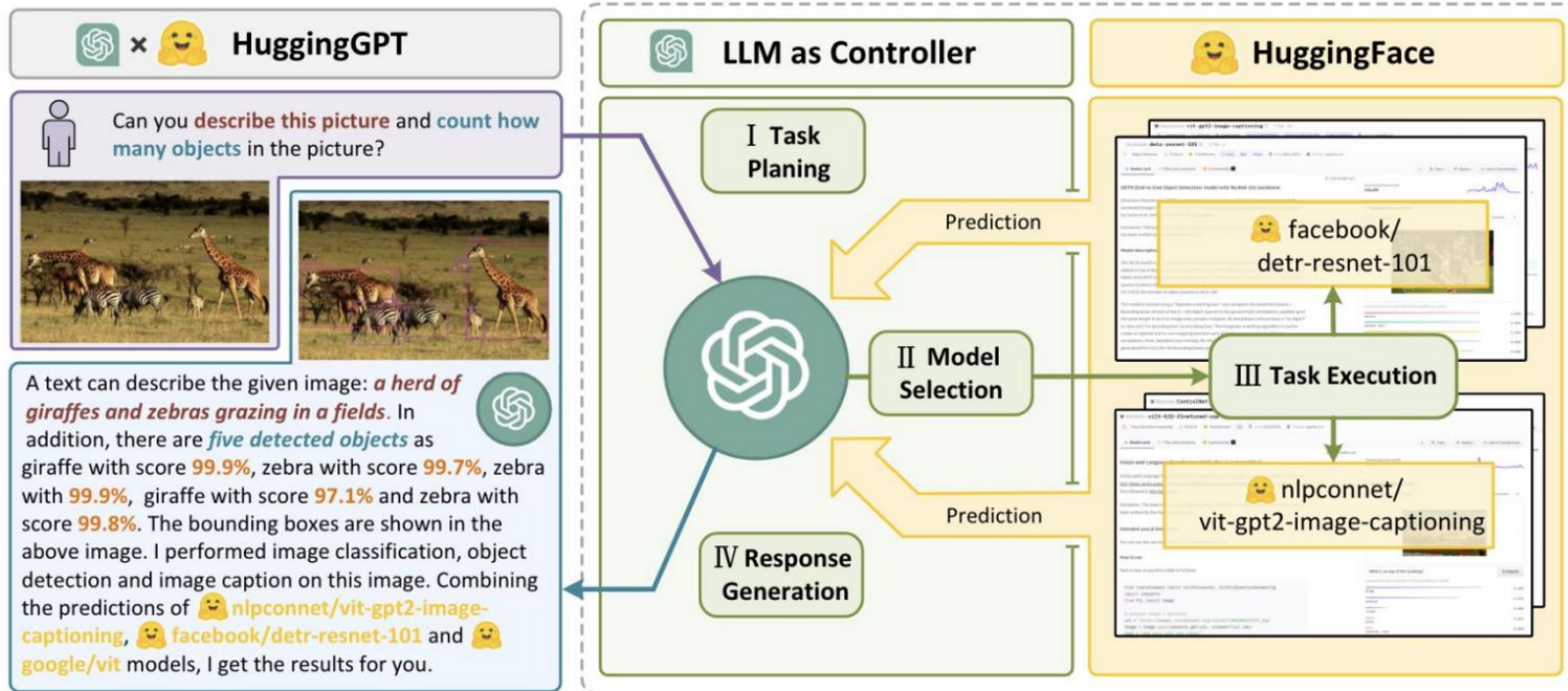


# LLMs + APIs to expert models: HuggingGPT

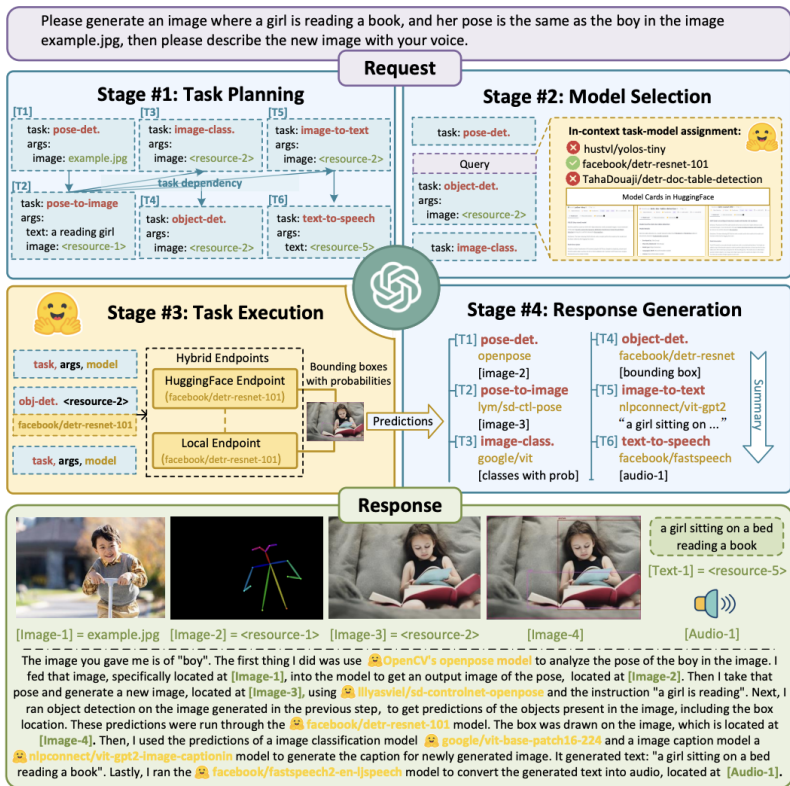


# LLMs + APIs to expert models: HuggingGPT

Lots of AI models are available in different fields and modalities, but cannot handle complex artificial intelligence tasks.



# LLMs + APIs to expert models: HuggingGPT



The system comprises of 4 stages:

- **Task Planning:** LLM analyze the user's requests, breaking them down into solvable tasks through prompts.
- **Model Selection:** LLM is presented with a list of models to choose from and distributes the tasks to expert models. LLM.
- **Task Execution:** Expert models execute on the specific tasks and log results.
- **Response Generation:** LLM receives the execution results and provides summarized results to users.

# LLMs + APIs to expert models: HuggingGPT

## Evaluation for task planning abilities:

- Single Task: The user request involves only one task.
- Sequential Task: The user's request needs to be broken down into a sequence of multiple subtasks.
- Graph Task: The user's request needs to be decomposed into a directed acyclic graph.



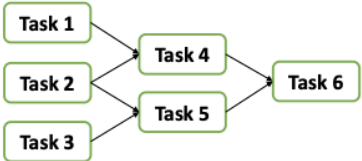
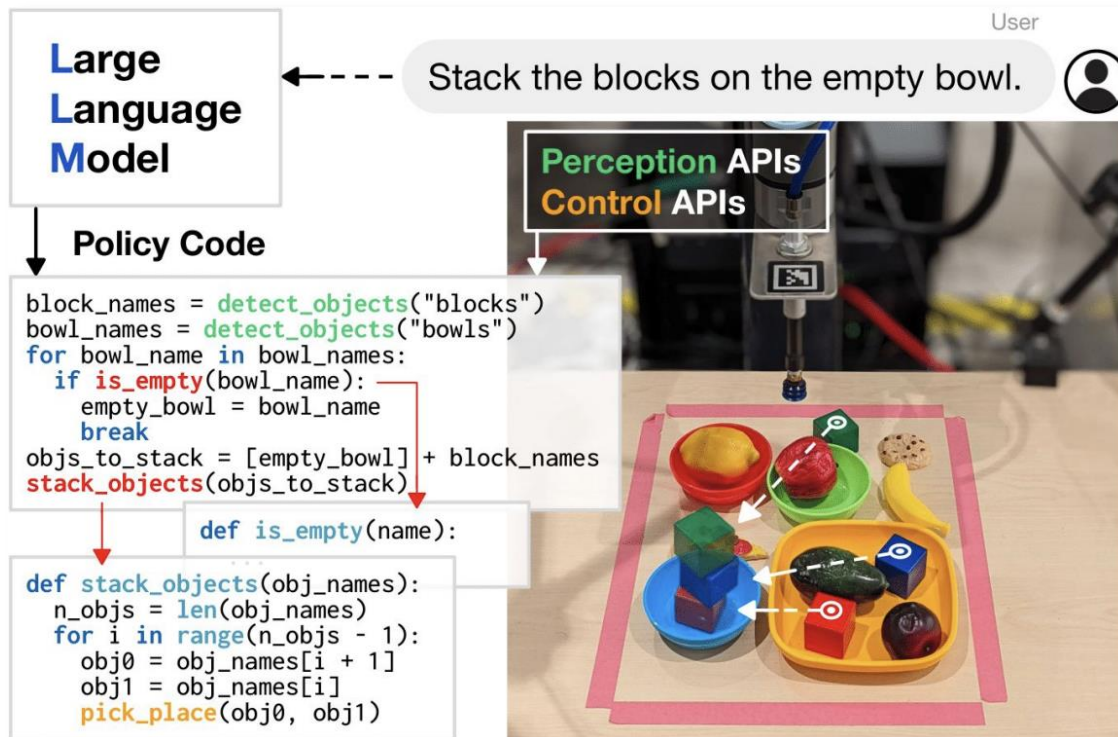
Task Type	Diagram	Example	Metrics
Single Task		Show me a funny image of a cat	Precision, Recall, F1, Accuracy
Sequential Task		Replace the cat with a dog in example.jpg	Precision, Recall, F1 Edit Distance
Graph Task		Given a collection of image A: a.jpg, B: b.jpg, C: c.jpg, please tell me which image is more like image B in terms of semantic, A or C?	Precision, Recall, F1 GPT-4 Score

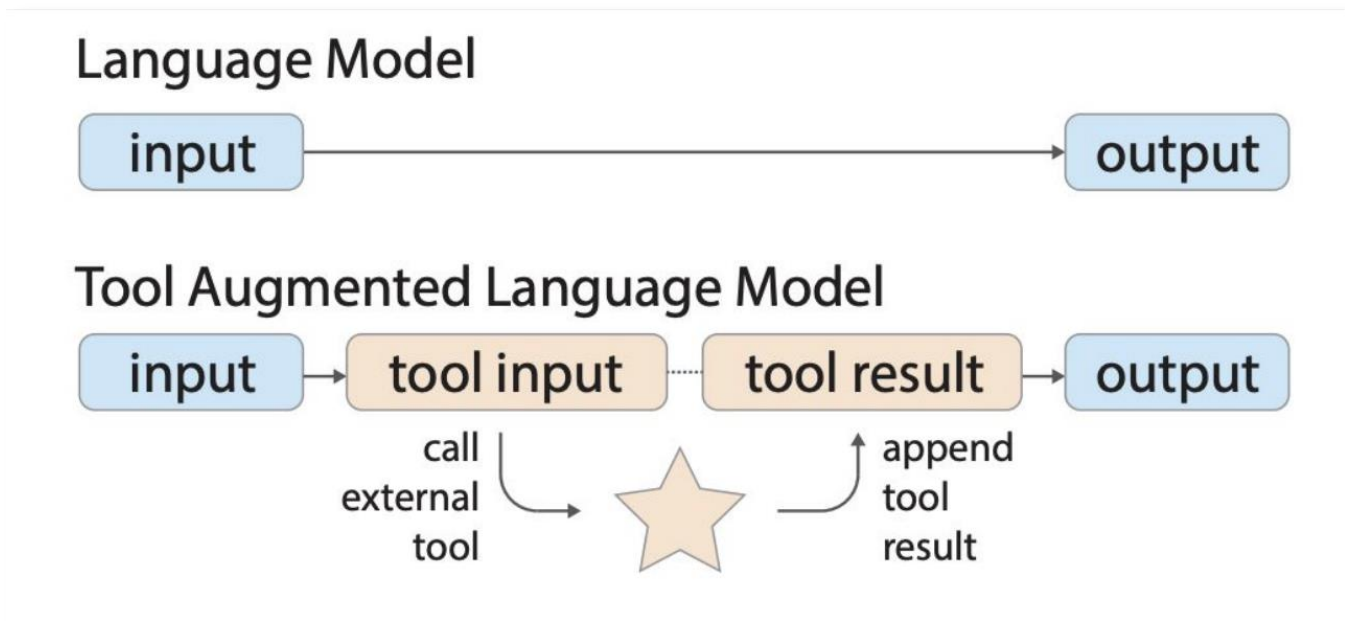
Table 2: Evaluation for task planning in different task types.

# LLMs + code, robotic arm, expert models: Code as Policies





# LLMs + training for tool use: TALM



TALM: Tool Augmented Language Models

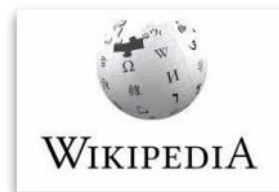
# LLMs + training for tool use: Toolformer

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

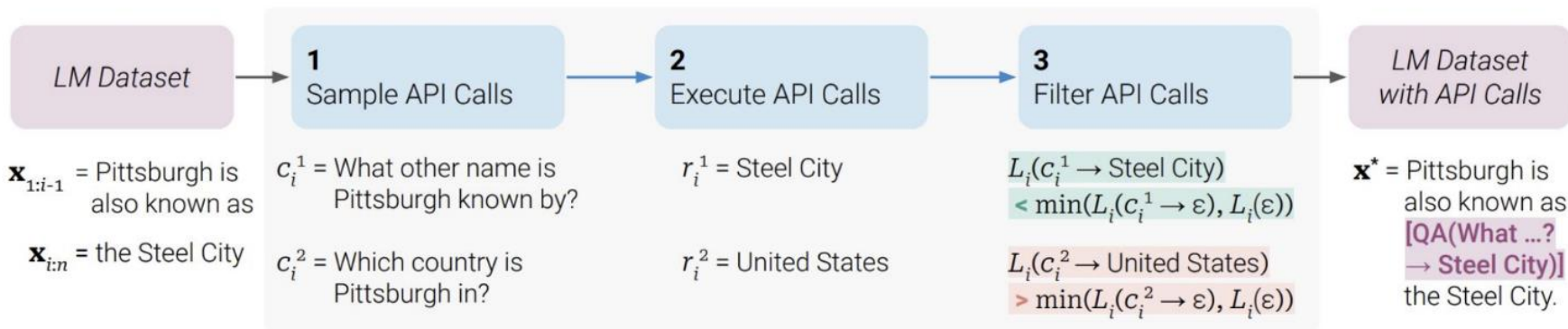
Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

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.



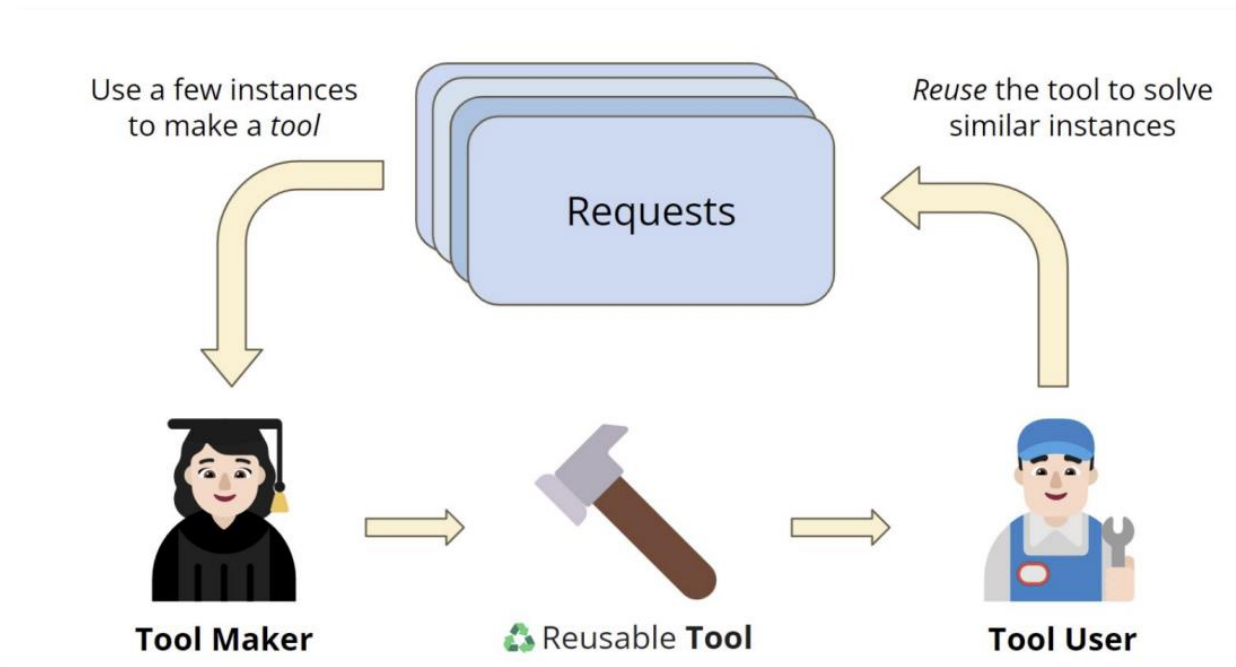
# LLMs + training for tool use: Toolformer



## **Element 2: tools**

Extension of tools in LLMs

## Extension-1: LLM as tool maker



LATM: Large Language Models as Tool Makers

## Extension-2: API-Bank

**API-Bank** is a benchmark for evaluating the performance of tool-augmented LLMs. It contains 53 commonly used API tools, a complete tool-augmented LLM workflow, and 264 annotated dialogues that involve 568 API calls.

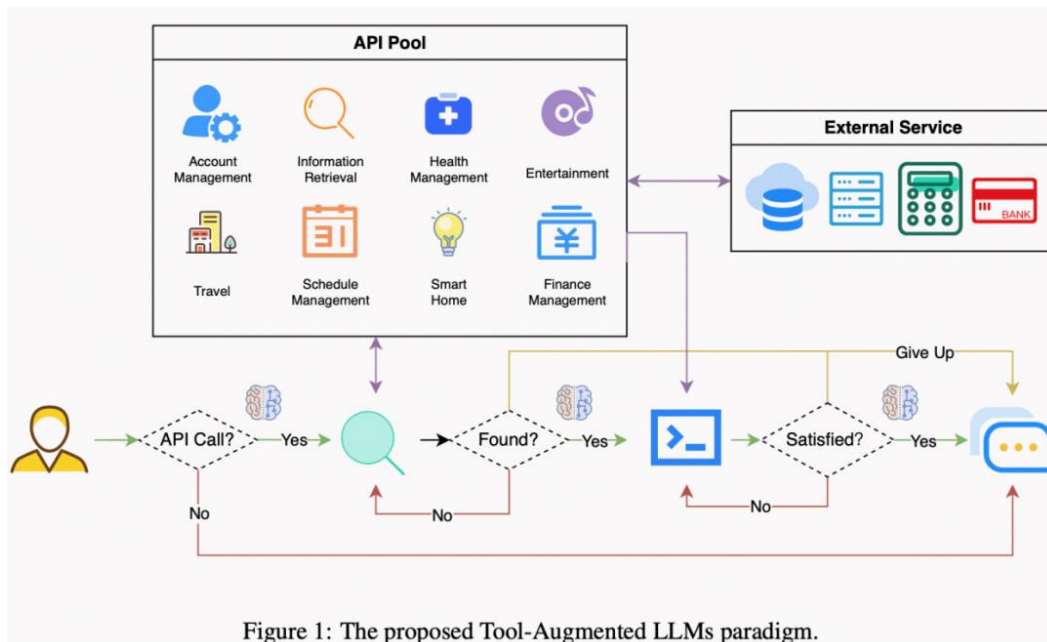


Figure 1: The proposed Tool-Augmented LLMs paradigm.

## Extension-2: API-Bank

### Evaluation index

**Level-1:** Evaluate LLM's ability to *call the API* (**Accuracy**); given a description of the API, the model needs to determine whether to call the API.

**Level-2:** Further evaluate LLM's ability to *retrieve APIs* (**Rouge**); the model needs to retrieve APIs that may solve user needs.

**Level-3:** Examine the ability of LLM *planning API* (**number of turns**).

	level-1	level-2	level-3
Num of Dialogues	214	50	8
Num of API calls	399	135	34

Table 1: The statistics of API-Bank.

## Challenges and future work

- **Complexity:** more complex domain professional/unseen tools?
- **Interactivity:** go beyond single turn?
- **Evaluation:** multiple possible solutions? Real-time interactive evaluation?
- **Efficiency:** smaller models?
- **Reliability:** know when to abstain, know its capacity, memorizing and querying tools?
- **Others**
  - Better tool API design/tool making?
  - Personalization?
  - .....



## **Element 3: Memory**

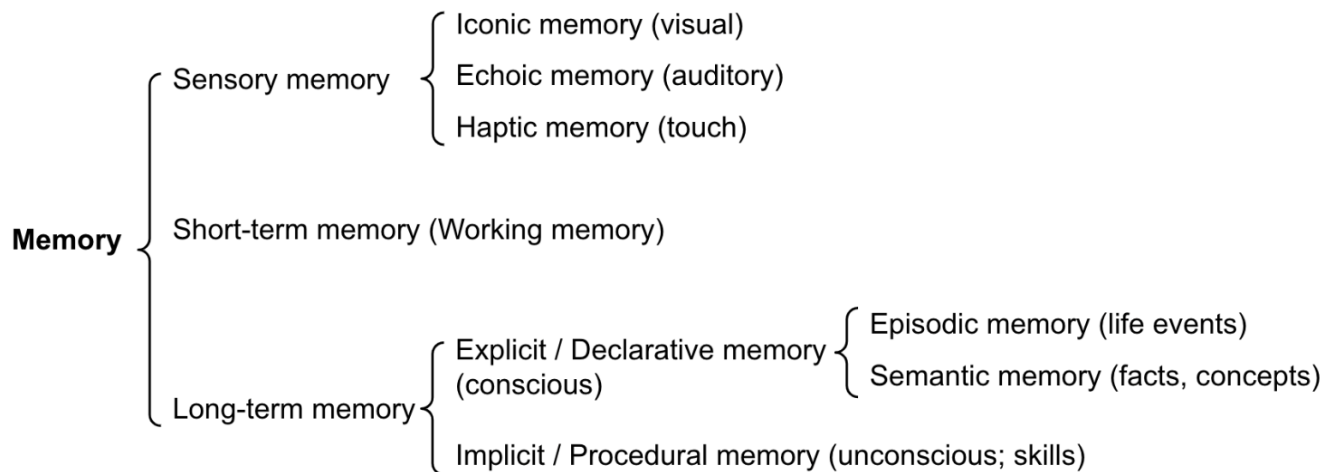
# LLM Agent Memory: **Types of Memory in human brains**

- 1. Sensory Memory:** This is the earliest stage of memory, providing the ability to retain impressions of sensory information (visual, auditory, etc) after the original stimuli have ended. Sensory memory typically only lasts for up to a few seconds. Subcategories include iconic memory (visual), echoic memory (auditory), and haptic memory (touch).
- 2. Short-Term Memory (STM) or Working Memory:** It stores information that we are currently aware of and needed to carry out complex cognitive tasks such as learning and reasoning. Short-term memory is believed to have the capacity of about 7 items (Miller 1956) and lasts for 20-30 seconds.
- 3. Long-Term Memory (LTM):** Long-term memory can store information for a remarkably long time, ranging from a few days to decades, with an essentially unlimited storage capacity.

There are two subtypes of LTM:

- a. Explicit / declarative memory:** This is memory of facts and events, and refers to those memories that can be consciously recalled, including episodic memory (events and experiences) and semantic memory (facts and concepts).
- b. Implicit / procedural memory:** This type of memory is unconscious and involves skills and routines that are performed automatically, like riding a bike or typing on a keyboard.

# LLM Agent Memory: **Types of Memory in LLMs**



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

# Standard Prompting

- Combine a prompt template together with an input

Please answer this question:

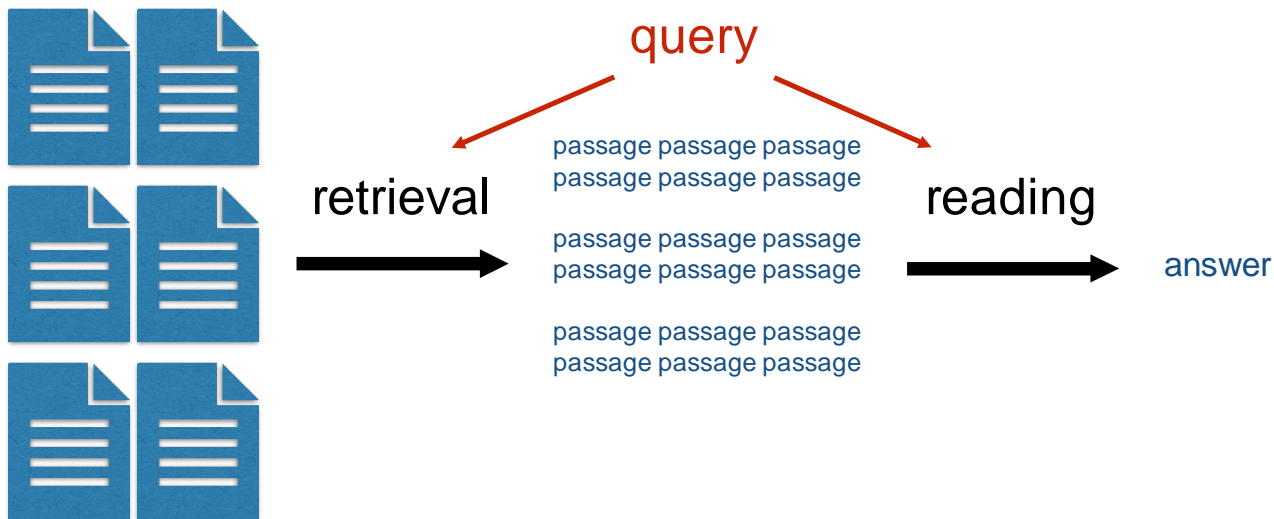
I think Vin Diesel has been a voice actor for several characters in TV series, do you know what their names are?

# Problems

- *Accuracy issues:*
  - **Knowledge cutoffs:** parameters are usually only updated to a particular time
  - **Private data:** data stored in private text or data repositories not suitable for training
  - **Learning failures:** even for data that the model was trained on, it might not be sufficient to get the right answer
- *Verifiability issues:* It is hard to tell if the answer is correct

# Retrieval-augmented Generation (Chen et al. 2017)

- **Retrieve** relevant passages efficiently
- **Read** the passages to answer the query





### You

I think Vin Diesel has been a voice actor for several characters in TV series, do you know what their names are? Please list all of them.



### ChatGPT

Vin Diesel has lent his voice to several characters across various TV series and shows. Here's a list of the characters he's voiced:

1. **Groot** in "Guardians of the Galaxy Vol. 3" (2023 Movie), "The Guardians of the Galaxy Holiday Special" (2022 TV Show), "I Am Groot" (2022 Short), "Thor: Love and Thunder" (2022 Movie), "Avengers: Endgame" (2019 Movie), "Ralph Breaks the Internet: Wreck-It Ralph 2" (2018 Movie), "Avengers: Infinity War" (2018 Movie), "Guardians of the Galaxy Vol. 2" (2017 Movie), and "Guardians of the Galaxy" (2014 Movie) [†].
2. **Santiago** in "ARK: The Animated Series" (2023 TV Show) [†].
3. **Dominic Toretto** in "Fast & Furious: Spy Racers" (2019 TV Show) [†].

<https://www.behindthevoiceactors.com/Vin-Diesel/>

Additionally, though the website for "Big Mouth" lists Vin Diesel, it appears to be a misunderstanding or an error, as Nick Kroll is credited as the voice of Vin Diesel in that show [†].

<https://www.behindthevoiceactors.com/tv-shows/Big-Mouth/Vin-Diesel/>

These roles showcase Diesel's versatility and contribution to voice acting, particularly in animated series and specials, where he brings iconic characters to life with his distinctive voice.



**Element 4: action**



## Action: Introduction

In the construction of the agent, the action module receives action sequences sent by the planning module and carries out actions to interact with the environment.

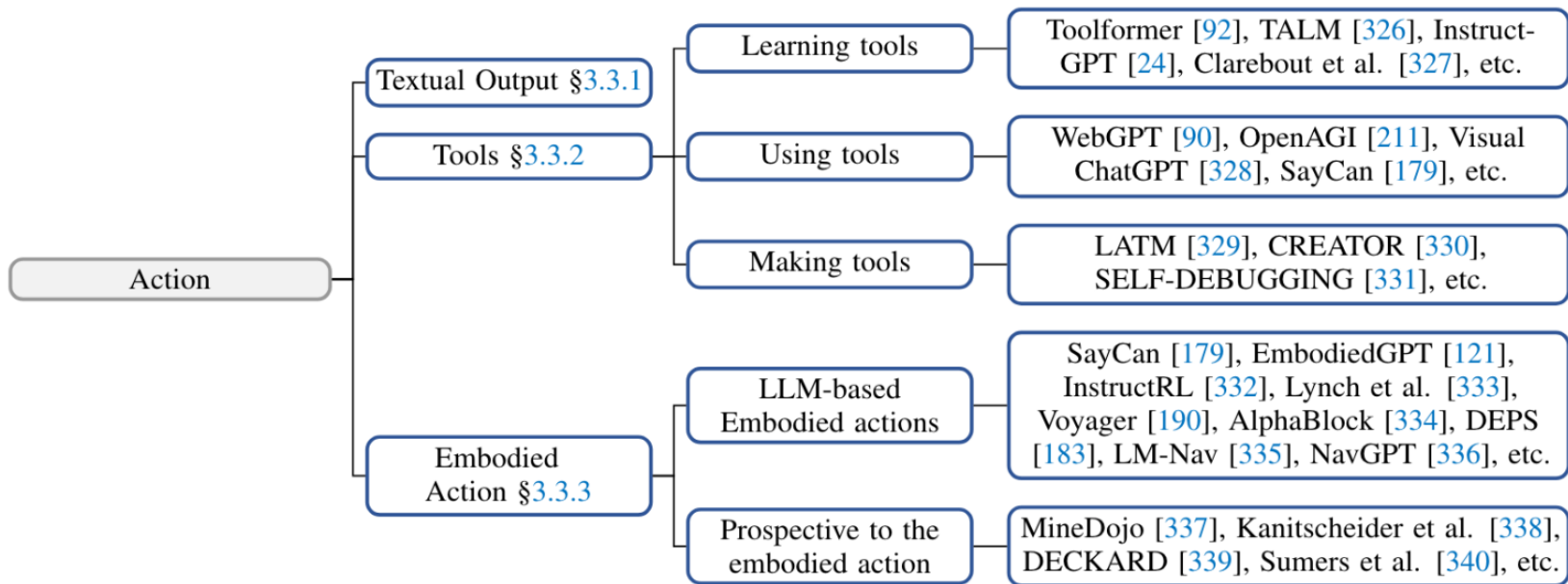
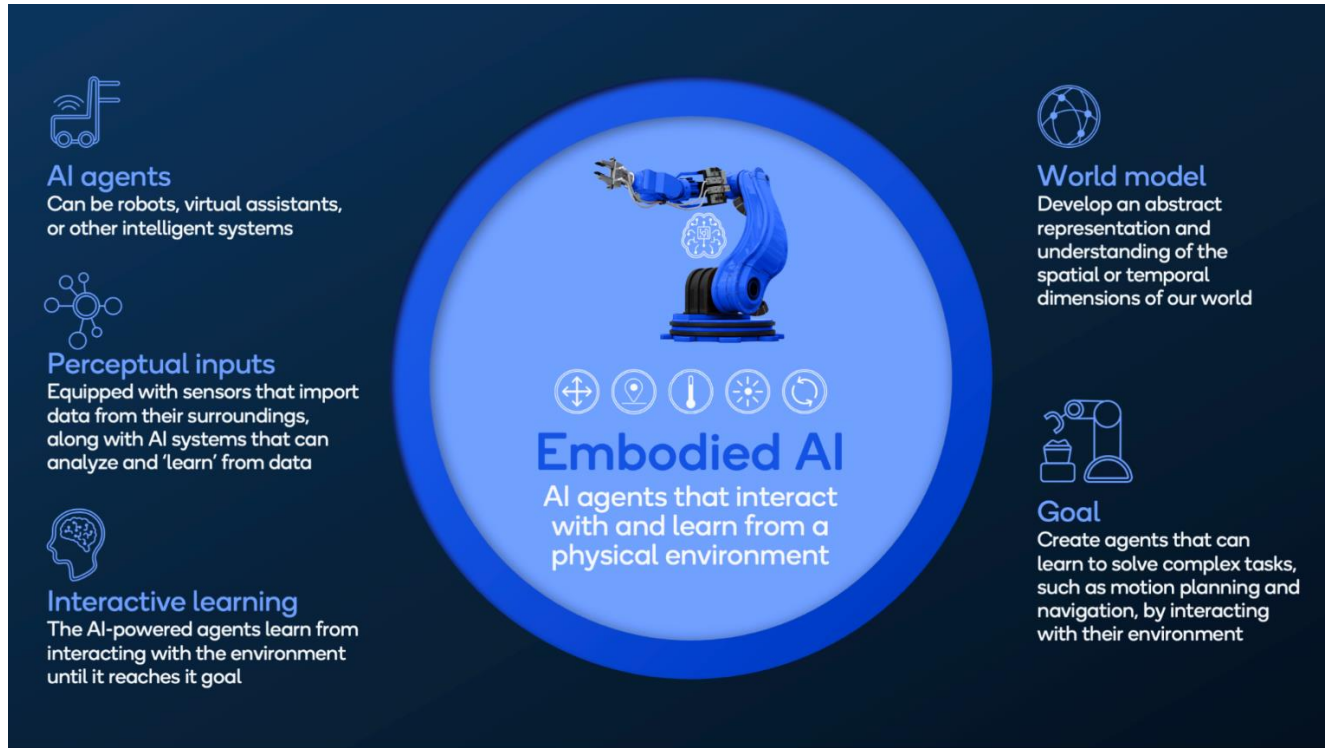


Figure 5: Typology of the action module.

## Action: Embodied AI

In the pursuit of Artificial General Intelligence (AGI), the embodied agent is considered a pivotal paradigm while it strives to integrate model intelligence with the physical world.



## Action: Embodied AI

Embodied AI should be capable of **actively perceiving**, **comprehending**, and **interacting** with physical environments, making decisions, and generating specific behaviors to modify the environment based on LLM's extensive internal knowledge. We collectively term these as ***embodied actions***, which enable agents' ability to interact with and comprehend the world in a manner closely resembling human behavior



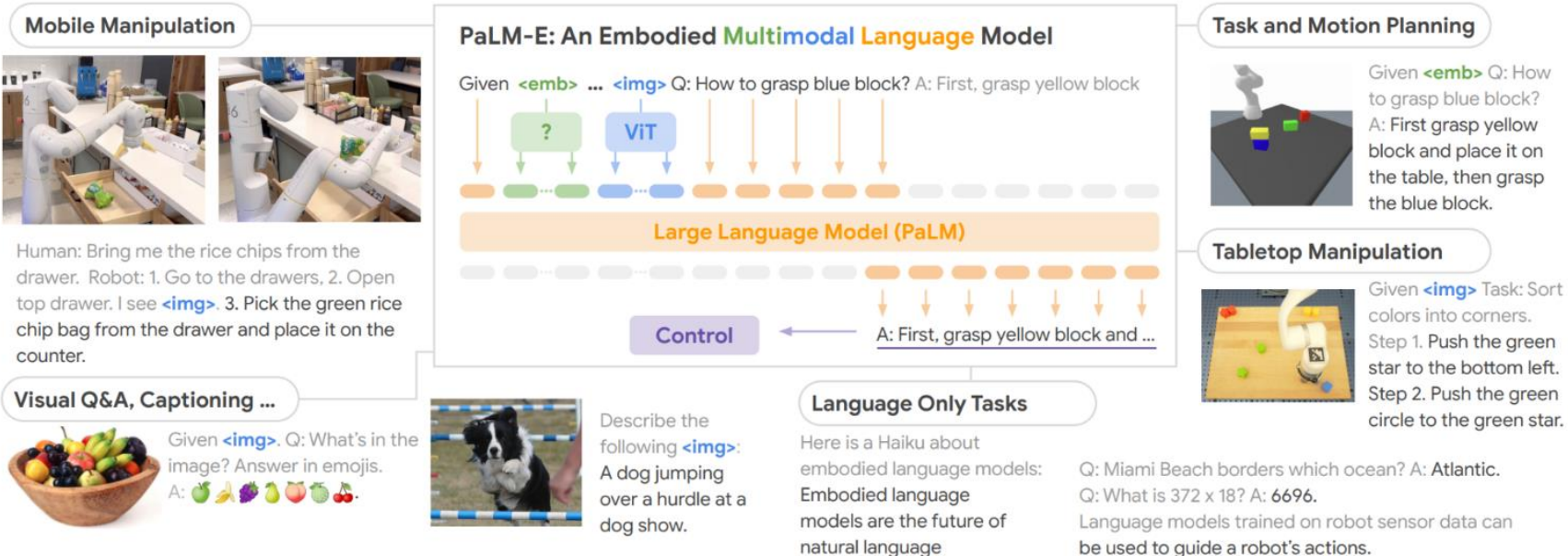
## Action: Embodied AI

### The potential of LLM-based agents for embodied actions.

- **Cost efficiency:** Some on-policy algorithms struggle with sample efficiency as they require fresh data for policy updates while gathering enough embodied data for high-performance training is costly and noisy.
- **Embodied action generalization:** An agent's competence should extend beyond specific tasks. When faced with intricate, uncharted real-world environments, it's imperative that the agent exhibits dynamic learning and generalization capabilities
- **Embodied action planning:** Planning constitutes a pivotal strategy employed by humans in response to complex problems as well as LLM-based agents.

# Embodied AI: PaLM-E: An Embodied Multimodal Language Model

PaLM-E transfers knowledge from visual-language domains into embodied reasoning – from robot planning in environments with complex dynamics and physical constraints, to answering questions about the observable world.



# Embodied AI: PaLM-E: An Embodied Multimodal Language Model

A single PaLM-E model directs the low-level policies of two real robots. Shown is a long-horizon mobile manipulation task in a kitchen, and one-shot / zero-shot generalization with a tabletop manipulation robot.

start → goal

PaLM-E guiding a real robot through a long horizon mobile manipulation task

Instruction: "bring me the rice chips from the drawer"

Go to the drawers	Open the top drawer	Take the rice chips out of the drawer	<b>Adversarial Disturbance:</b> human knocks the rice chips back into the drawer	Take the rice chips out of the drawer	Bring it to the user	Put it down

PaLM-E guiding a real robot through one-shot and zero-shot tabletop manipulation tasks

Move the green circle to the yellow hexagon	Move the blue triangle to the group	<u>success</u>	Move the green star to the top left corner	Move the green star to the green circle	<u>success</u>

**one-shot:** "Move the remaining blocks to the group"

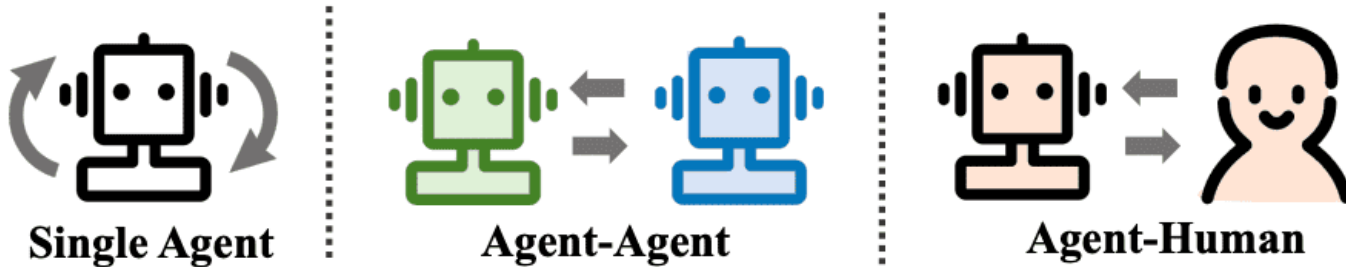
**zero-shot:** "Move the green blocks to the turtle"

# LLM agents Categories

# Category

The use cases for LLM agents, or Language Model-based agents, are vast and diverse. These agents, powered by large language models (LLMs), can be used in various scenarios, including:

1. Single-agent applications
2. Multi-agent systems
3. Human-Agent cooperation

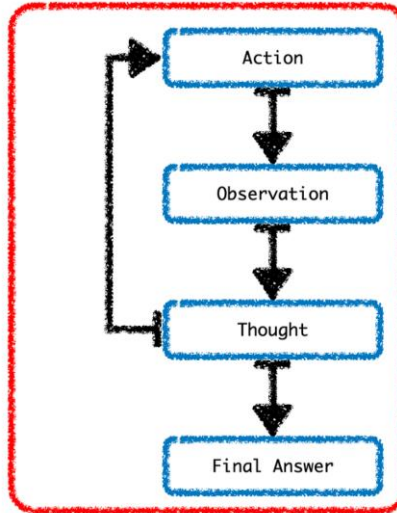




# Single-agent applications

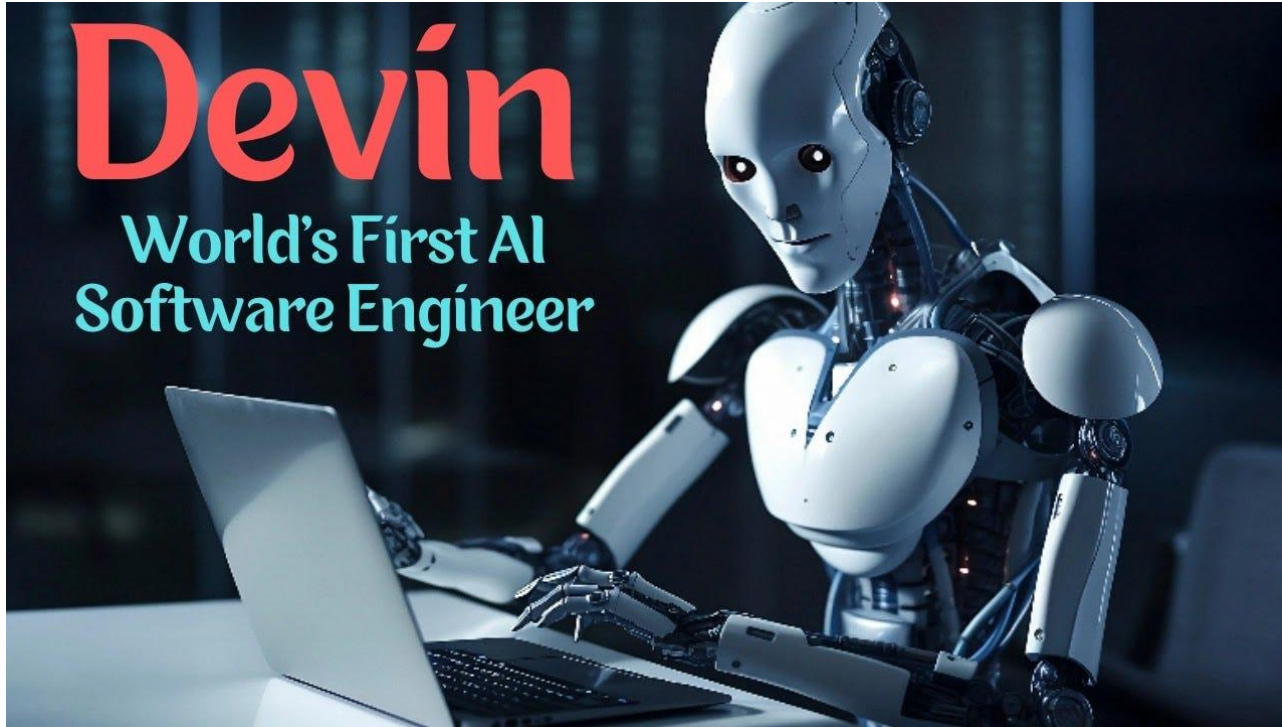


LangChain Agent - Sequence Of Events



LLM agents can be utilized as personal assistants to assist users in **breaking free from daily tasks and repetitive labor**. They can analyze, plan, and solve problems independently, reducing the work pressure on individuals and enhancing task-solving efficiency.

# The World's First AI Software Engineer



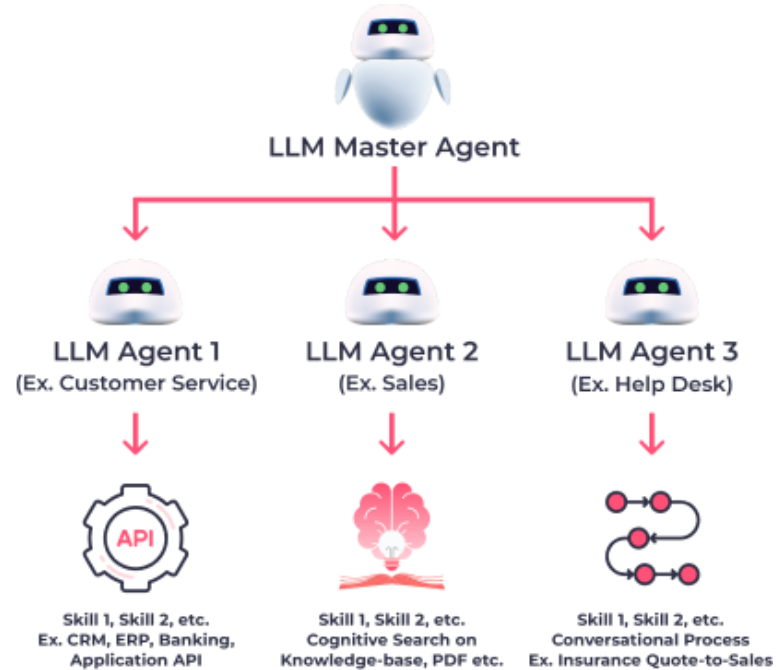
<https://www.cognition-labs.com/introducing-devin>

# Open-World Interaction, e.g. in Minecraft.



<https://craftjarvis.github.io/ROCKET-1/>

# Multi-agent systems



**Multi-agent systems:** LLM agents can interact with each other in a collaborative or competitive manner. This enables them to achieve advancement through teamwork or adversarial interactions. In these systems, agents can **work together on complex tasks** or **compete against each other** to improve their performance.

# Play Werewolf (狼人杀)

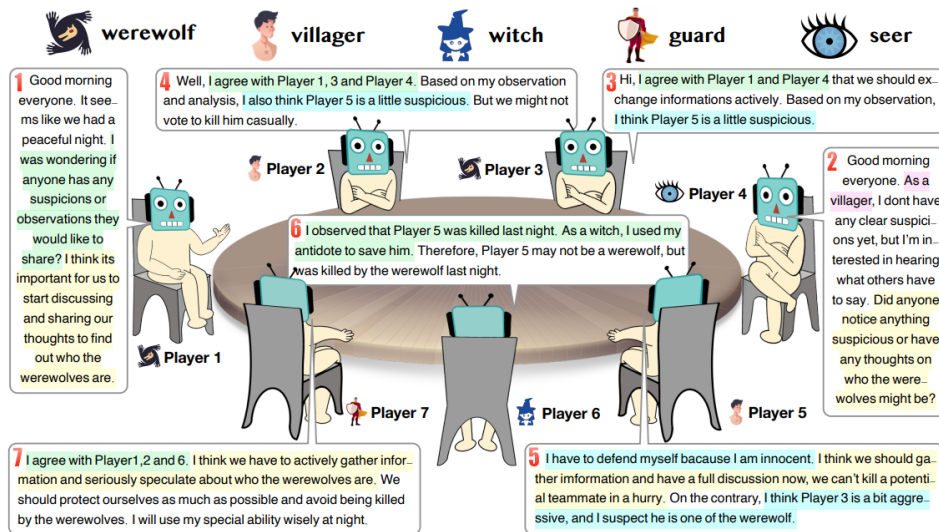


Figure 1: A snapshot of our implemented Werewolf game. There are 5 roles and 7 players, and each of them is acted by an LLM autonomously. The number before each talking denotes the speaking order. Some social behaviors can be primarily observed in this figure, including trust, confrontation, camouflage, and leadership.

# EconAgent

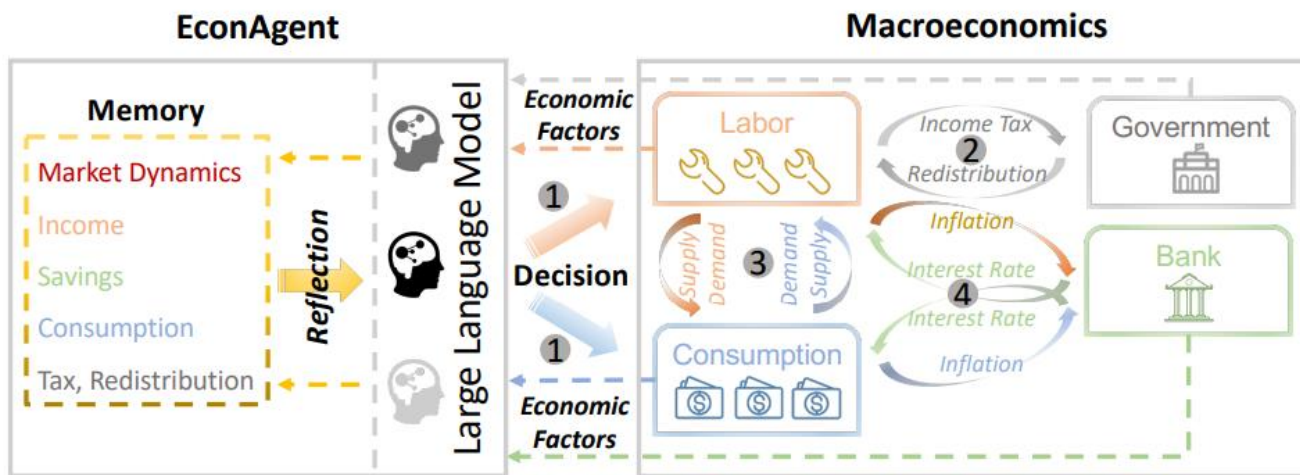


Figure 1: The illustration of our EconAgent (left) and simulation environment (right).

# StockAgent

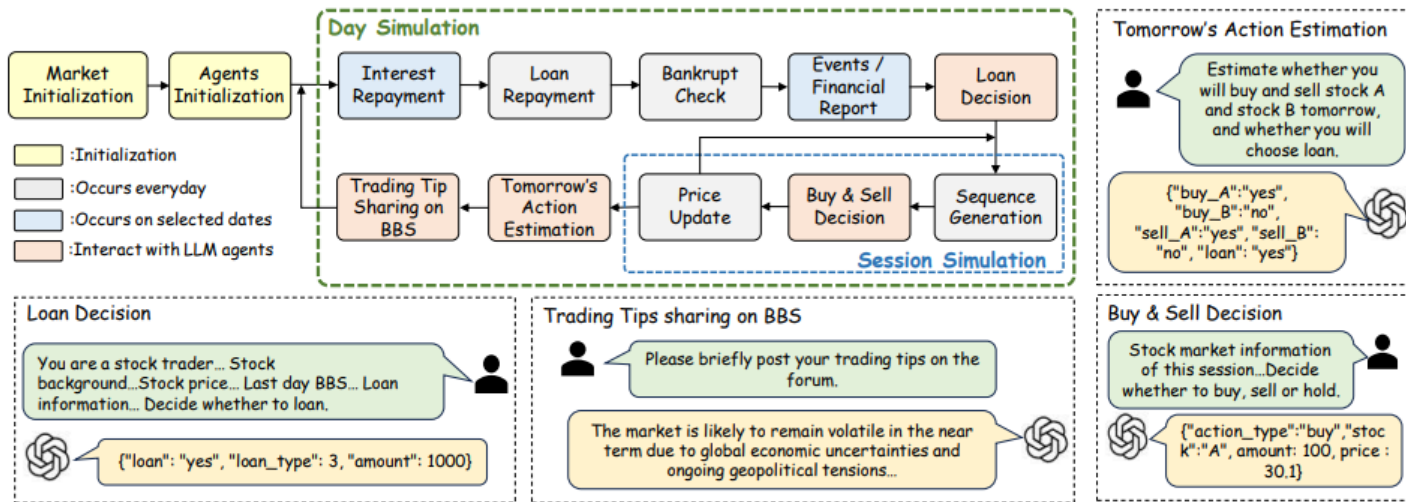


Fig. 2. The workflow of trading simulation.

# ElectionSim

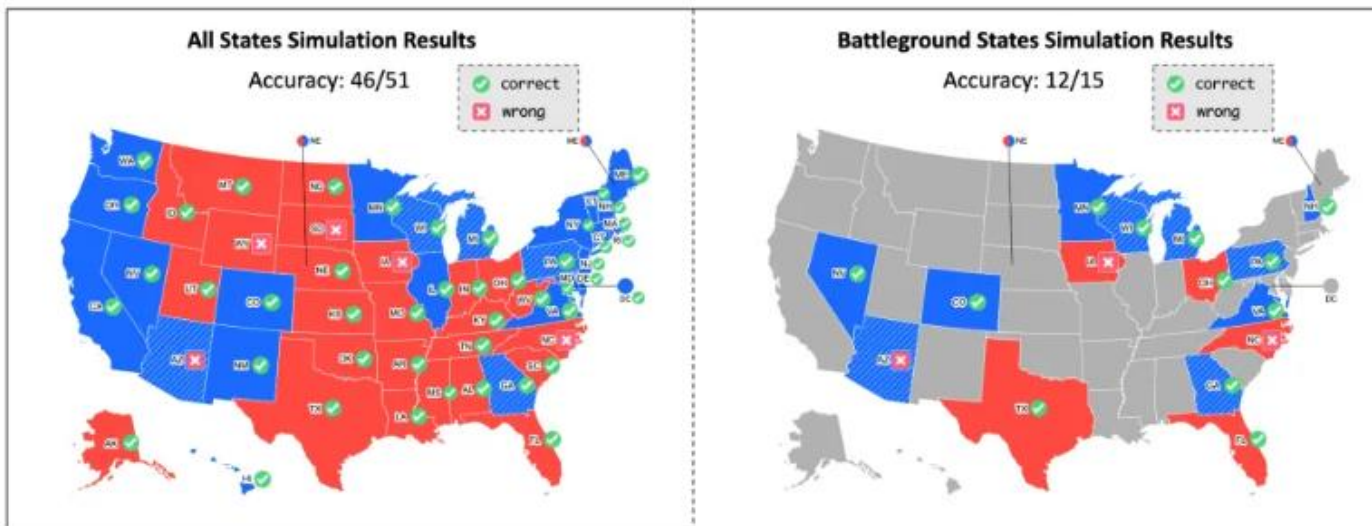
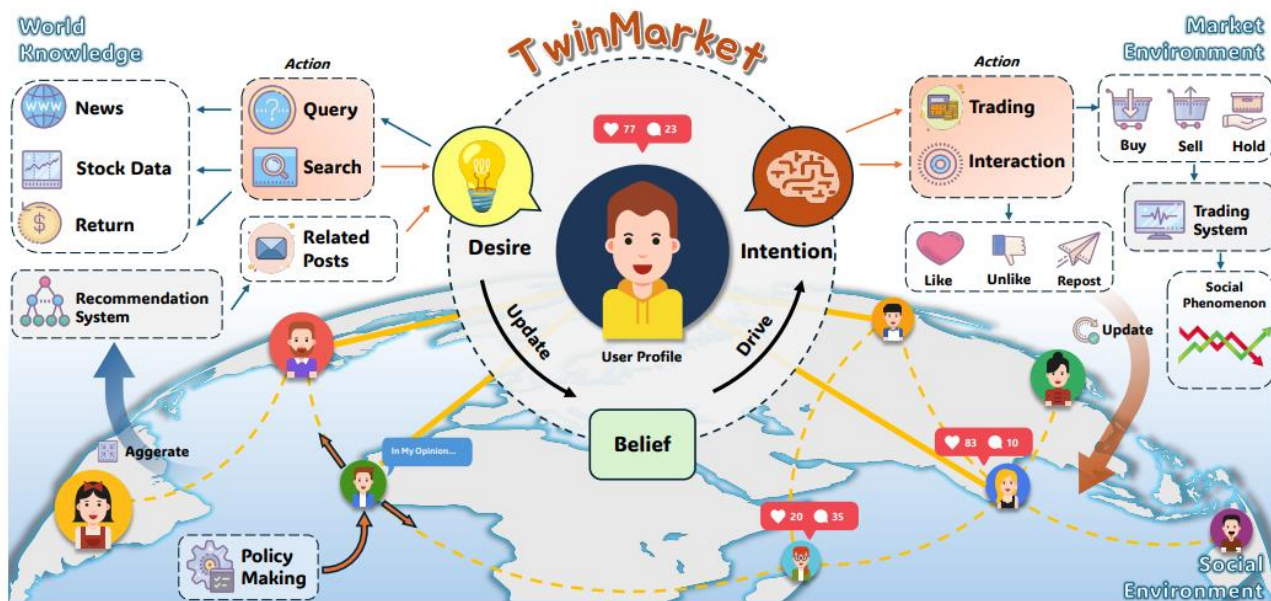


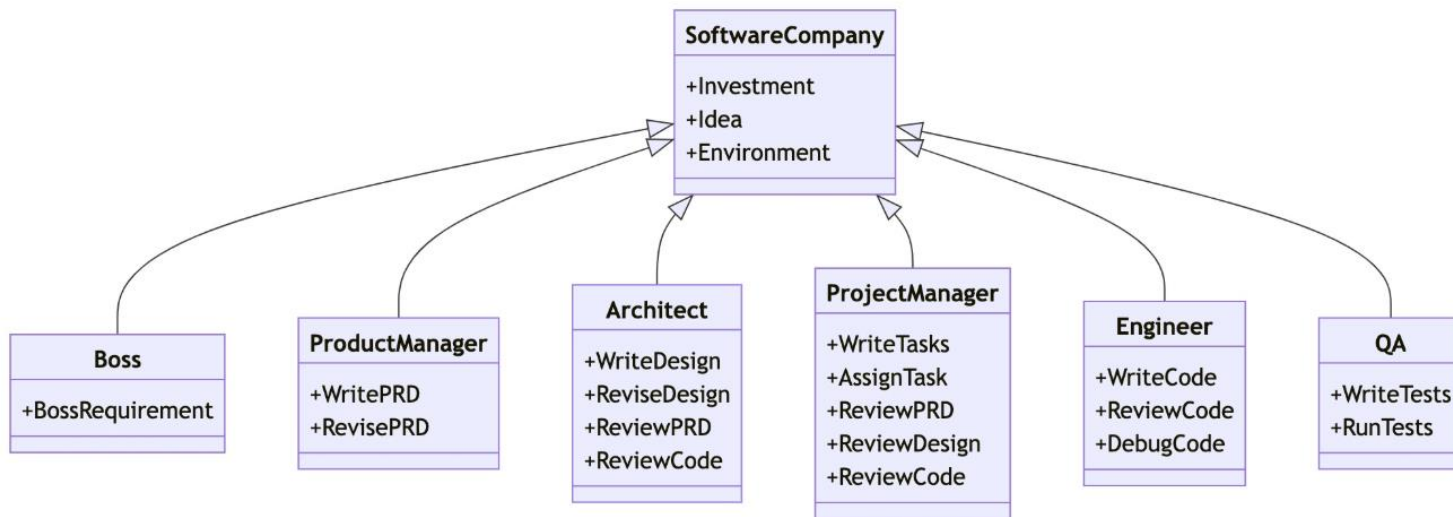
Figure 1: Simulation results of the 2020 Presidential Election. The colors represent the real-world results and the marks represent the simulation results accuracy.



# TwinMarket from financial market simulation

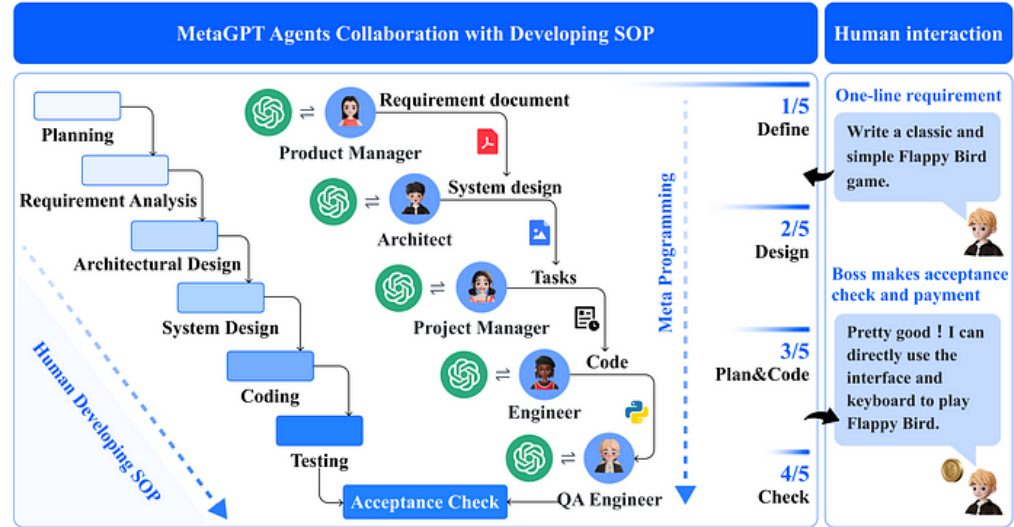


# MetaGPT



Software Company Multi-Agent Schematic (Gradually Implementing)

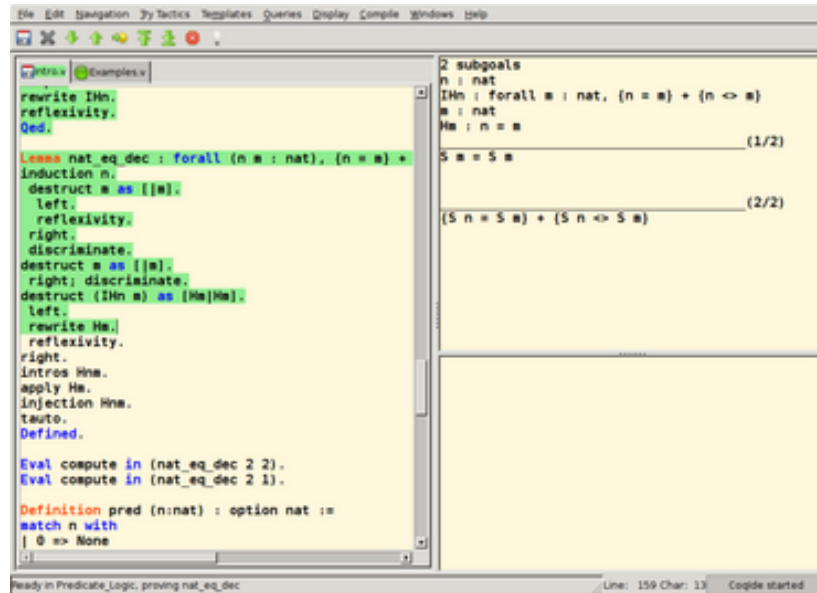
# Human-Agent cooperation



**Human-Agent cooperation:** LLM agents can **interact with humans**, providing them with assistance and performing tasks more efficiently and safely.

Example: interactively write code together with ChatGPT.

# Proof assistant (interactive theorem provers)



The screenshot shows the CoqIDE interface. The left pane contains the proof script for the lemma `nat_eq_dec`. The script uses `rewrite IHn`, `reflexivity`, and `Qed` to complete the proof. It also includes a `Definition` for a predicate `pred` on natural numbers. The right pane displays the current proof state, showing two subgoals. The first subgoal is `S m = S m`, which is marked as (1/2). The second subgoal is `(S n = S m) + (S n <> S m)`, marked as (2/2). The status bar at the bottom indicates the current file is `Predicate_Logic.proving_nat_eq_dec` and the cursor is at line 159, character 13.

```
File Edit Navigation Tactics Templates Queries Display Compile Windows Help
[Icons]
Examples
rewrite IHn.
reflexivity.
Qed.

Lemma nat_eq_dec : forall (n m : nat), (n = m) +
induction n.
destruct m as [|m].
left.
reflexivity.
right.
discriminate.
destruct m as [|m].
right; discriminate.
destruct (IHn m) as [Hm|Hm].
left.
rewrite Hm.
reflexivity.
right.
intros Hm.
apply Hm.
injection Hm.
tauto.
Defined.

Eval compute in (nat_eq_dec 2 2).
Eval compute in (nat_eq_dec 2 1).

Definition pred (n:nat) : option nat :=
match n with
| 0 => None
```

2 subgoals  
n : nat  
[IHn : forall m : nat, (n = m) + (n <> m)]  
m : nat  
Hm : n = m  
----- (1/2)  
S m = S m  
----- (2/2)  
(S n = S m) + (S n <> S m)

Ready in Predicate\_Logic.proving\_nat\_eq\_dec Line: 159 Char: 13 CoqIDE started

An interactive proof session in CoqIDE, showing the proof script on the left and the proof state on the right


# Interactive Creation

You are Alaya, a female Human Cyborg in Freedom, a settlement in the Enclave. You were born and raised in this city and know practically everything about it... except for a few minor details. A small group of people calling themselves the "Reaper's Rebellion" have announced their intentions to take physical possession of the city of Grace. They claim the city is in debt to them, and that the people who live there have been given license to rebel against the government and its 'enemies' by the so-called 'law of the land'. A number of citizens have been rallying in the streets, and so far the government has done nothing to stop them. Your job is to find a way to sabotage the rebellion before it gets out of hand.

- \* You go to your palace and think of a plan to stop the rebellion.


You approach your palace and begin to think about a plan on how to stop the Reaper's Rebellion. A few minutes pass, then you have an idea about a way to stop the rebellion. You think that the Reapers' most dangerous weapon is their 'replicas', which they use to look like real people.

- \* You prepare the other soldiers to be ready for the Reaper's Rebellion and when they strike to destroy their replicas.



You quickly take part of your soldiers and go into the streets to spread the word. You tell them to be ready for the rebellion to strike at any moment. Several minutes pass. A group of people dressed like the Reapers' soldiers come marching down the streets. They have replicas of soldiers and citizens with them. You know that the rebels are planning to attack the palace right now.

- \* You through subterfuge, you thwart their efforts to attack the palace.



You go out of the palace in order to fool the rebels into thinking that the palace is being attacked. You tell the remaining soldiers that the rebels have already broken through the gates. Several minutes pass. You see that the rebels have been tricked and have left the city walls. The plan worked well.

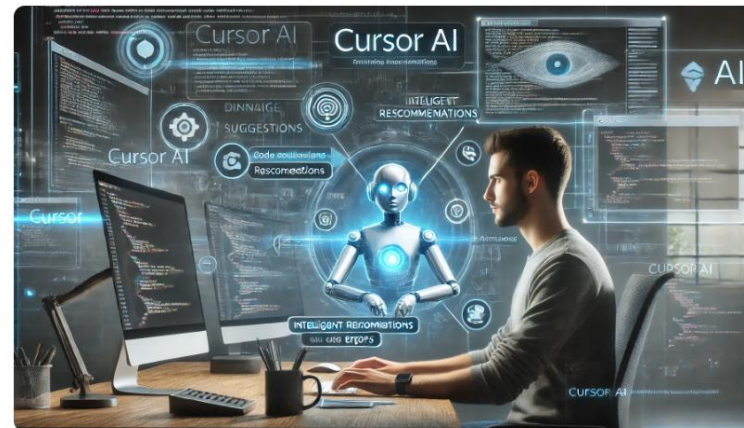
Do What do you do?

<https://aidungeon.com/>

<https://voicebot.ai/2022/08/01/ai-dungeons-synthetic-story-and-pictures-released-on-steam-gaming-platform/>



Coding with yourself



Coding with LLM agents

# Acknowledgements

- <https://github.com/Paitesanshi/LLM-Agent-Survey>
- <https://github.com/WooooDyy/LLM-Agent-Paper-List>
- [Generative Agents: Interactive Simulacra of Human Behavior.](#)
- <https://wenting-zhao.github.io/complex-reasoning-tutorial/>
- <https://acl2023-retrieval-lm.github.io/>
- <https://github.com/xlang-ai/llm-tool-use>