



An Introduction of LLM-Powered Autonomous Agents

余阳

2023.11.30



Personal Information

2

- 姓名：余阳
- 年级：研三
- 导师：刘淇 教授
- 研究方向：推荐系统，预训练模型
- 工位：B702
- 邮箱：yflyl613@mail.ustc.edu.cn



Outline

3

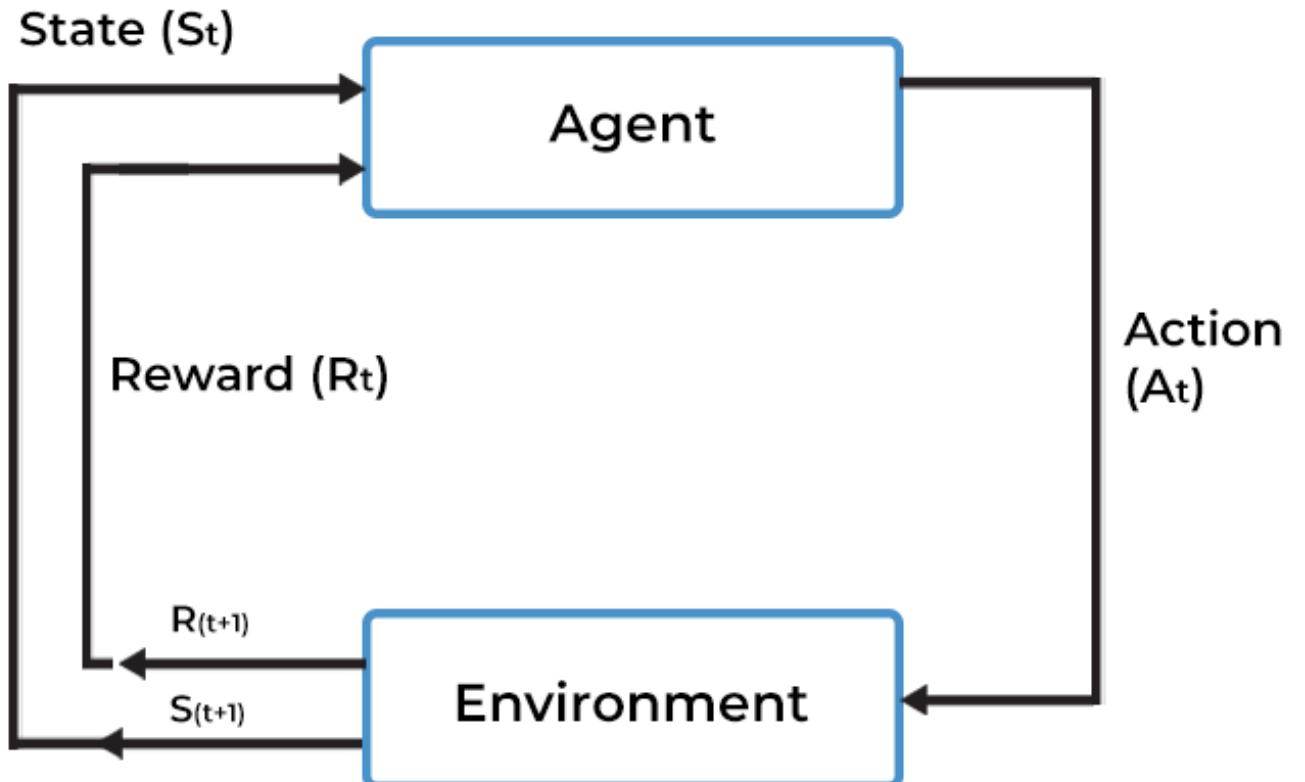
- Agent System Overview
- Component I: Planning
- Component II: Memory
- Component III: Tool Use
- Case Study
- Future Challenges

Agent System Overview

4

■ Reinforcement Learning-based Agent

- How to train an agent by interacting with the environment?

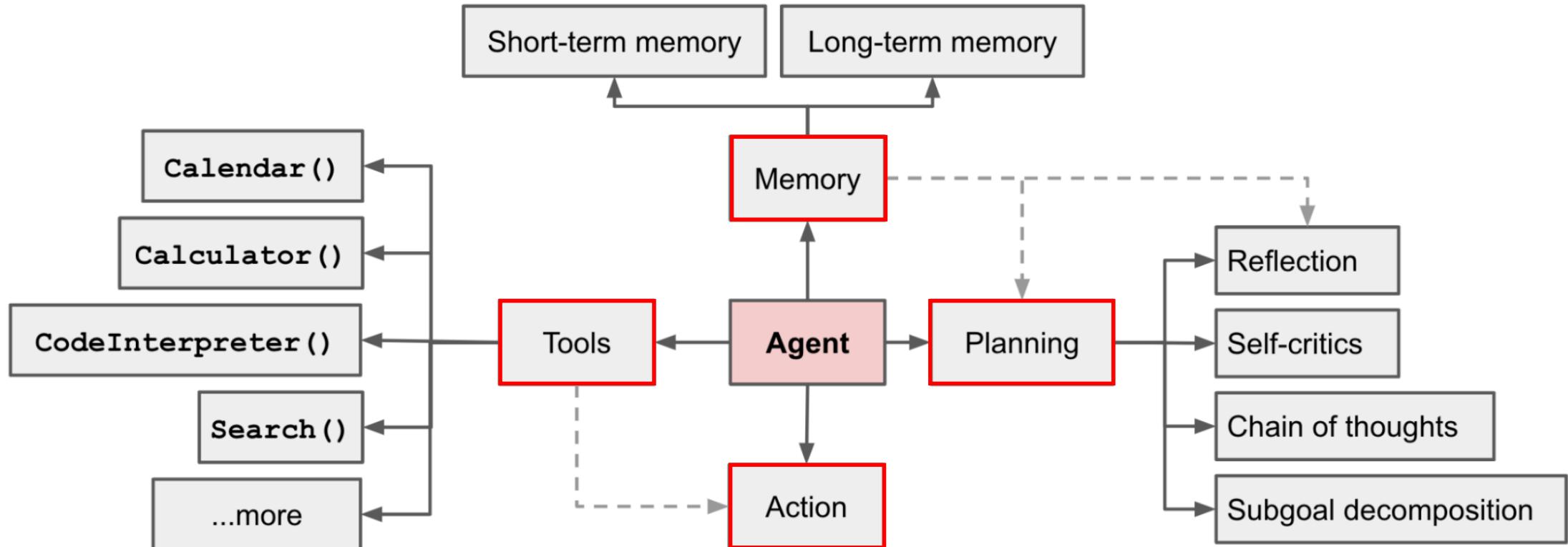


Agent System Overview

5

■ LLM-Powered Agent

- How to power an agent with the LLM?





Outline

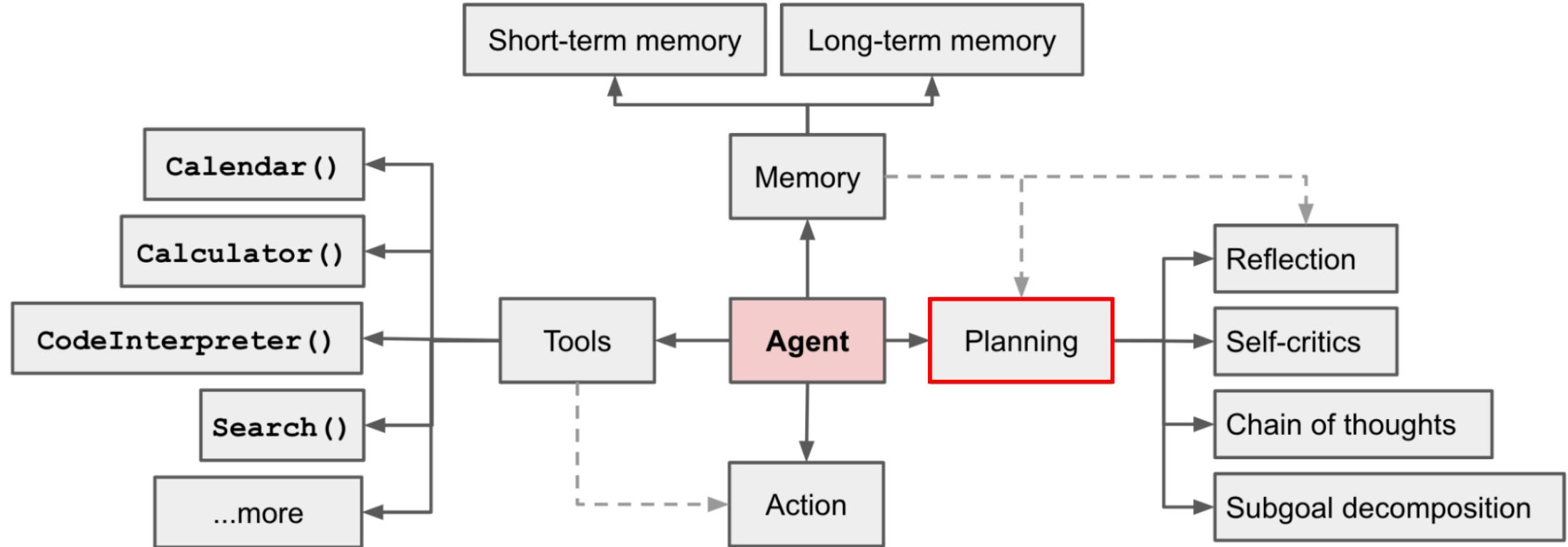
6

- Agent System Overview
- Component I: Planning
- Component II: Memory
- Component III: Tool Use
- Case Study
- Future Challenges

Component I: Planning

7

- Agents need to look both forward (**task decomposition**) and backward (**self-reflection**).

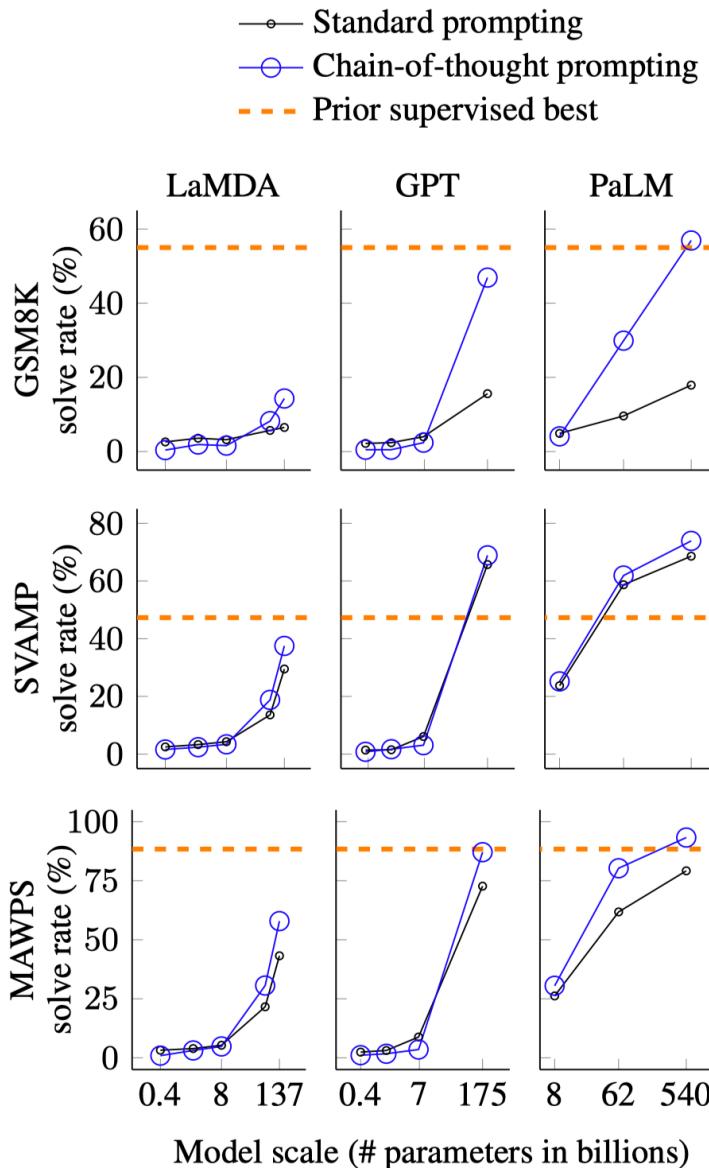
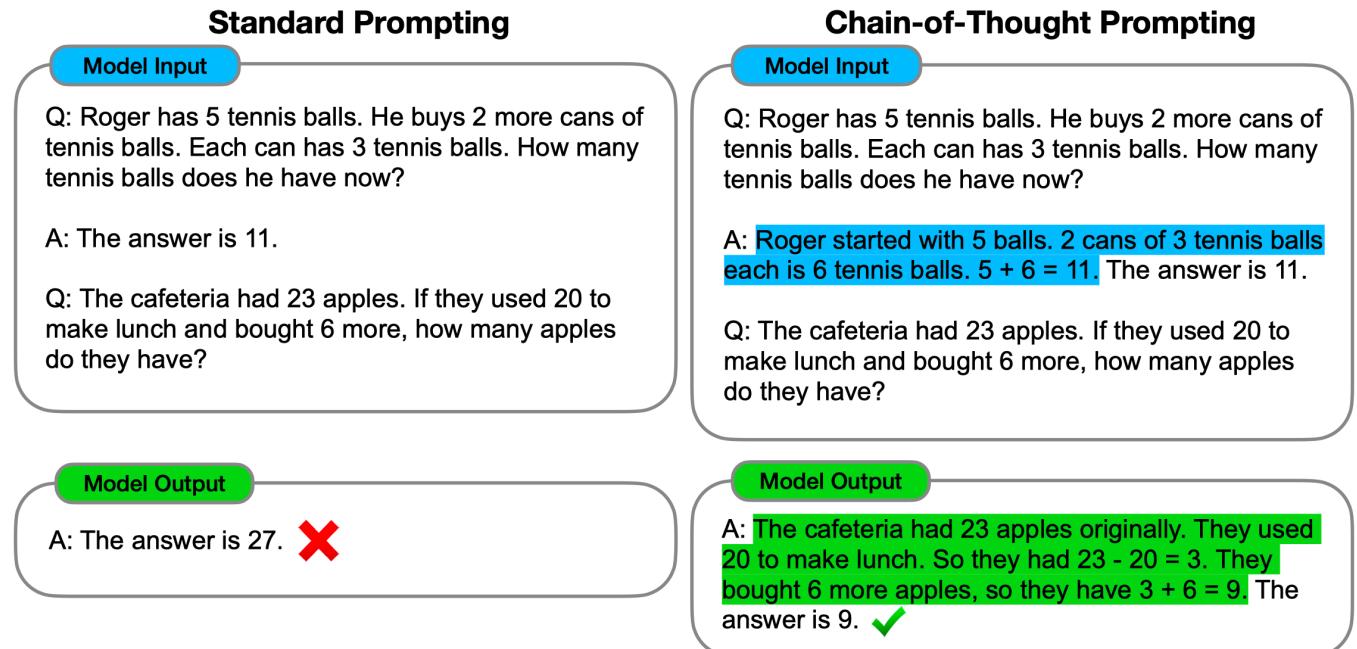


Component I: Planning

8

■ Task Decomposition: Chain-of-Thought (CoT)

- A series of intermediate natural language reasoning steps.
- Decompose hard tasks into smaller steps and solve each before giving the final answer.
- Provide an interpretable window into the behavior of the model.





Component I: Planning

9

■ Task Decomposition: Zero-shot-CoT (*some magic words*)

- Let's think step by step & Therefore, the answer is ...
- Let's work this out in a step by step way to be sure we have the right answer.

(a) Few-shot

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: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The answer is 8. X

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 X

(b) Few-shot-CoT

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 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

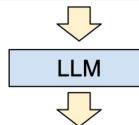
(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

**【1st prompt】
Reasoning Extraction**

Q: On average Joe throws 25 punches per minute. A fight lasts 5 rounds of 3 minutes. How many punches did he throw?
A: Let's think step by step.

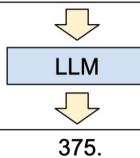


In one minute, Joe throws 25 punches.
In three minutes, Joe throws $3 * 25 = 75$ punches.
In five rounds, Joe throws $5 * 75 = 375$ punches.

**【2nd prompt】
Answer Extraction**

Q: On average Joe throws 25 punches per minute. A fight lasts 5 rounds of 3 ...
A: Let's think step by step.

In one minute, Joe throws 25 punches. ... In five rounds, Joe throws $5 * 75 = 375$ punches. .
Therefore, the answer (arabic numerals) is

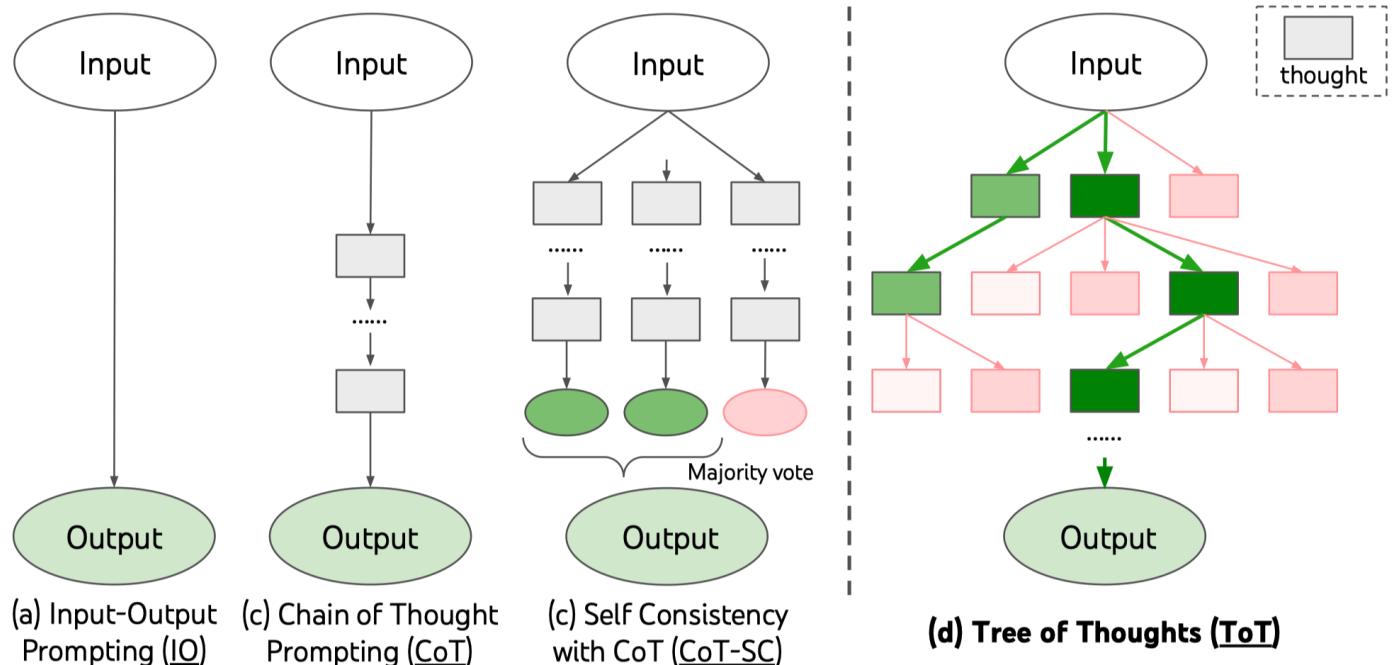


Component I: Planning

10

■ Task Decomposition: Tree-of-Thoughts (ToT)

- Consider multiple different reasoning paths and self-evaluate choices.
- Look ahead or backtrack with search algorithms (e.g., BFS, DFS).





Component I: Planning

11

■ Task Decomposition: Tree-of-Thoughts (ToT)

□ Thought decomposition: based on problem properties

□ Thought generation

- x : input, z_i : thought, $s = [x, z_{1 \dots i}]$: state

- Sample i.i.d thoughts from a CoT prompt: $z^{(j)} \sim p_\theta^{CoT}(z_{i+1}|s) (j = 1 \dots k)$

- Propose thoughts sequentially using a “propose prompt”: $[z^{(1)}, \dots, z^{(k)}] \sim p_\theta^{propose}(z_{i+1}^{(1\dots k)}|s)$

□ Thought evaluator

- Value each state independently

- Vote across states

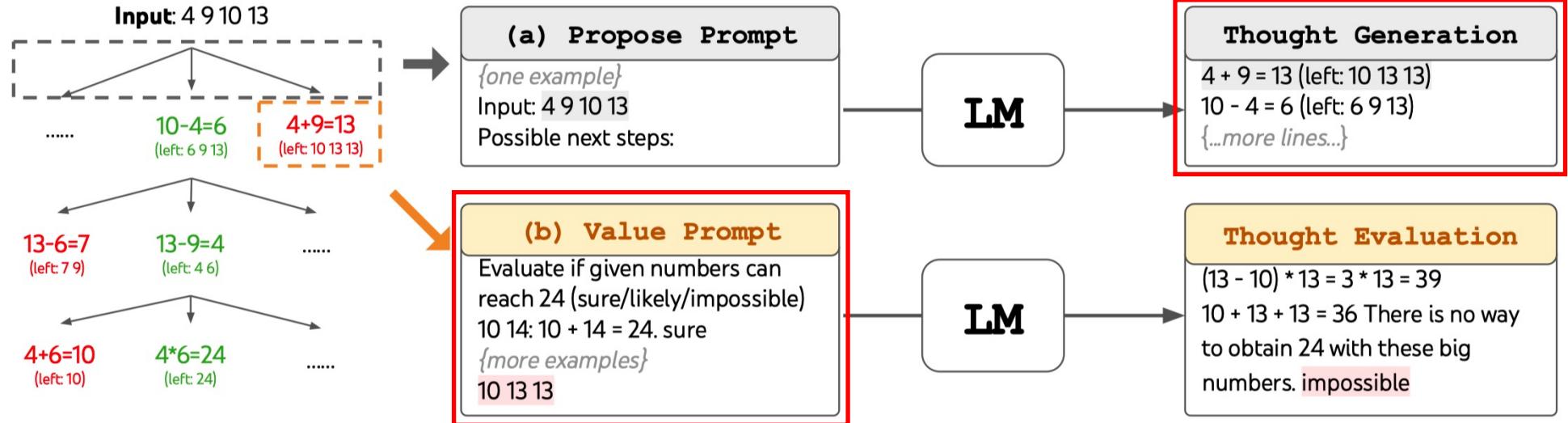
□ Search algorithm: BFS, DFS, A*, MCTS, etc.

Component I: Planning

12

■ Task Decomposition: Tree-of-Thoughts (ToT)

□ Game of 24



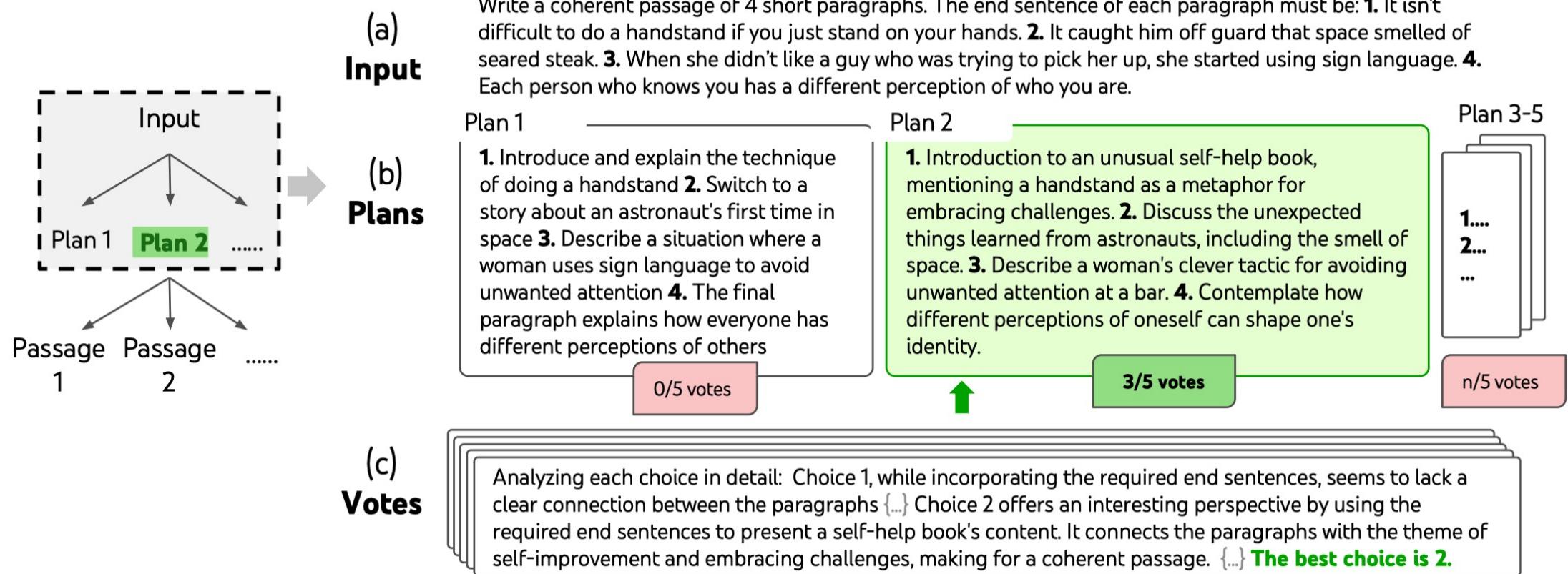
Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	74%
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

Component I: Planning

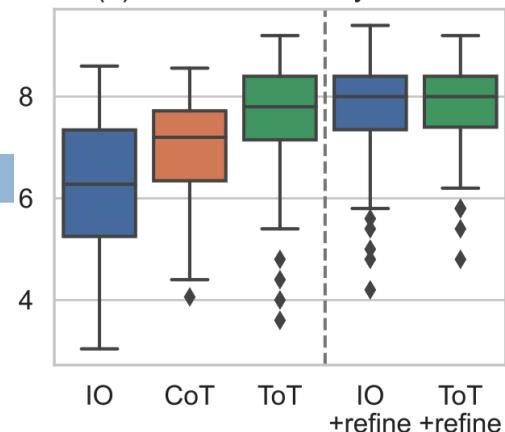
13

■ Task Decomposition: Tree-of-Thoughts (ToT)

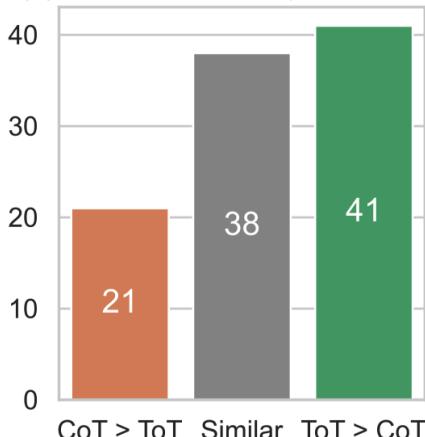
□ Creative writing



(a) GPT-4 coherency scores



(b) Human coherency comparison





Component I: Planning

14

■ Task Decomposition: LLM+P

- Incorporate external classical planners with Planning Domain Definition Language (PDDL) into LLMs.
 - Translate the problem into “Problem PDDL”.
 - Use the classical planner to generate a PDDL plan based on the “Problem PDDL” and “Domain PDDL”.
 - Translate the PDDL plan back to natural language.
- Assume the availability of domain-specific PDDL and a suitable planner (mainly in robotic setups).

Domain	Success Rate %				
	LLM ⁻	LLM	LLM ^{ToT}	LLM+P ⁻	LLM+P
BARMAN	0	0	0	0	20 (100)
BLOCKSWORLD	20	15 (30)	0 (5)	0	90
FLOORTILE	0	0	0	0	0
GRIPPERS	25 (60)	35 (50)	10 (20)	0	95 (100)
STORAGE	0	0 (25)	0	0	85
TERMES	0	0	0	0	20
TYREWORLD	5	15	0	0	10 (90)

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)
```

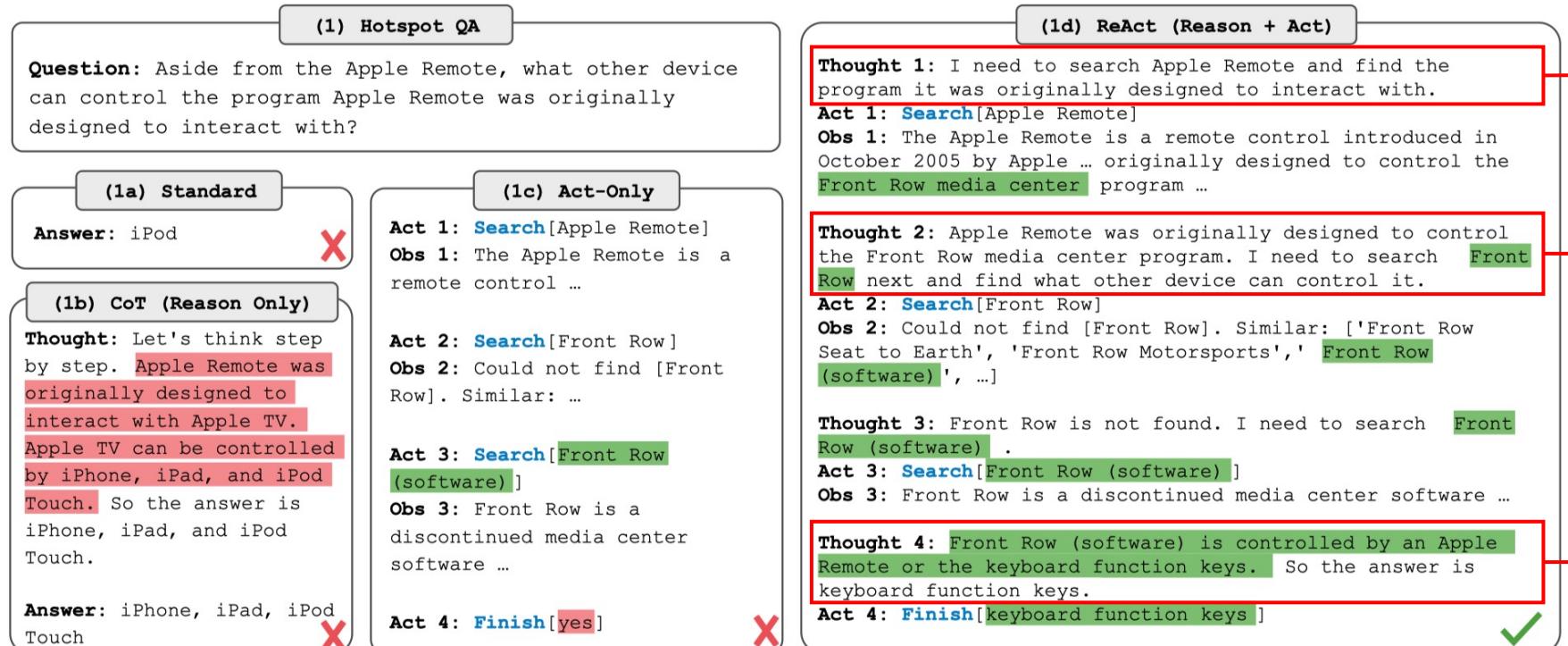


Component I: Planning

15

■ Self-Reflection: ReAct

- Integrate reasoning and acting within LLMs.
- Extend the action space to be a combination of task-specific discrete actions (interact with the environment) and the language space (generate reasoning path).



Decompose task goals and create action plans.

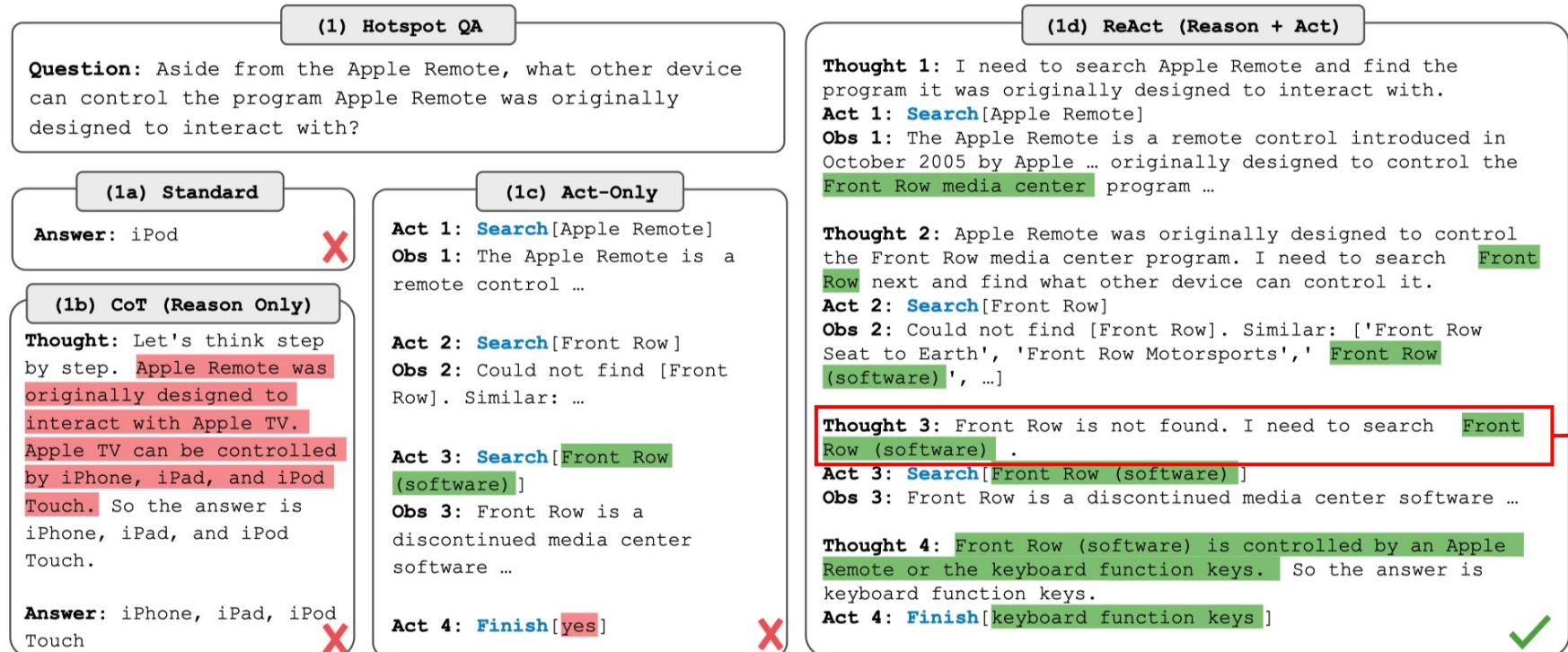
Extract important parts from observations

Component I: Planning

15

■ Self-Reflection: ReAct

- Integrate reasoning and acting within LLMs.
- Extend the action space to be a combination of task-specific discrete actions (interact with the environment) and the language space (generate reasoning path).



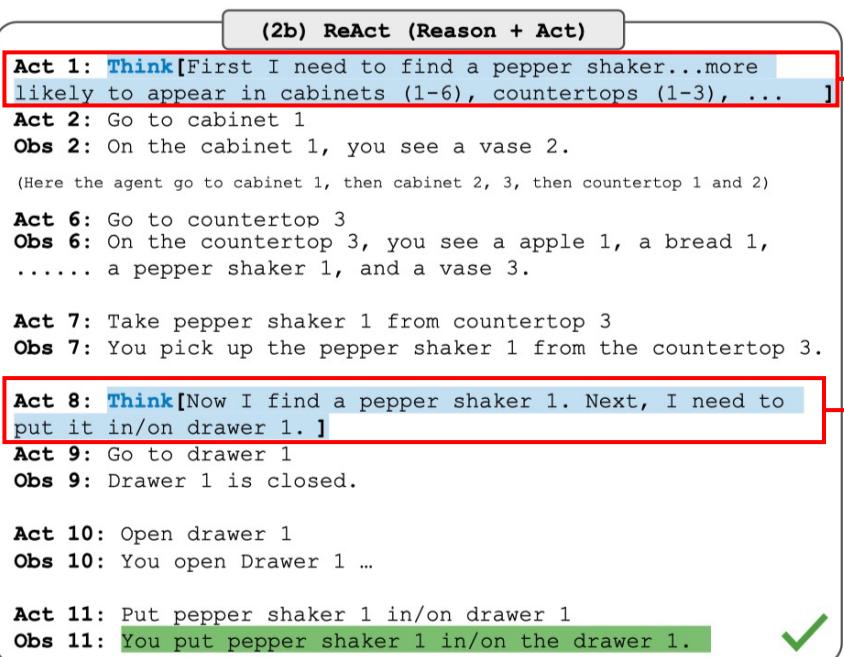
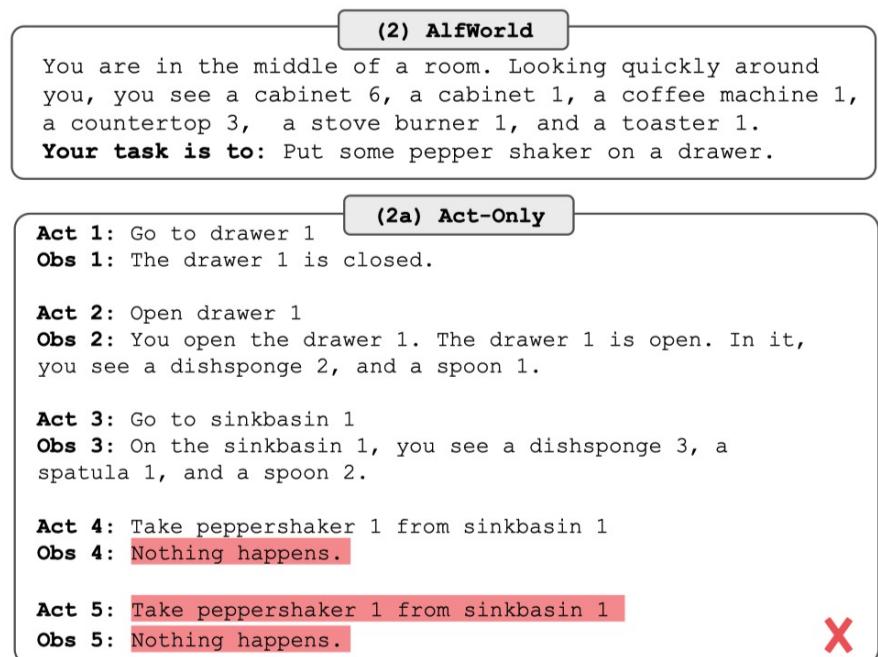
Handle exceptions and adjust action plans.

Component I: Planning

16

■ Self-Reflection: ReAct

- Integrate reasoning and acting within LLMs.
- Extend the action space to be a combination of task-specific discrete actions (interact with the environment) and the language space (generate reasoning path).



Inject commonsense knowledge relevant to task solving.

Track progress and transit action plans.

Component I: Planning

17

■ Self-Reflection: ReAct

- Knowledge-intensive reasoning tasks

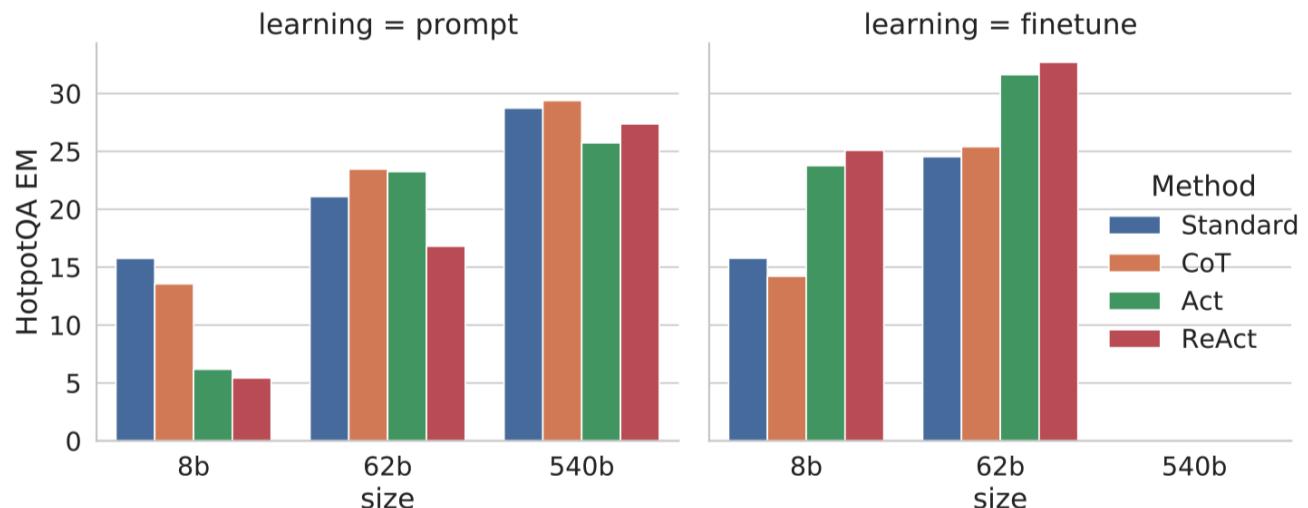


Figure 3: Scaling results for prompting and finetuning on HotPotQA with ReAct (ours) and baselines.

Prompt Method ^a	HotpotQA (EM)	Fever (Acc)
Standard	28.7	57.1
CoT (Wei et al., 2022)	29.4	56.3
CoT-SC (Wang et al., 2022a)	33.4	60.4
Act	25.7	58.9
ReAct	27.4	60.9
CoT-SC → ReAct	34.2	64.6
ReAct → CoT-SC	35.1	62.0
Supervised SoTA^b	67.5	89.5

Table 1: PaLM-540B prompting results on HotpotQA and Fever.

Component I: Planning

18

■ Self-Reflection: ReAct

□ Decision-making tasks

Method	Pick	Clean	Heat	Cool	Look	Pick 2	All
Act (best of 6)	88	42	74	67	72	41	45
ReAct (avg)	65	39	83	76	55	24	57
ReAct (best of 6)	92	58	96	86	78	41	71
ReAct-IM (avg)	55	59	60	55	23	24	48
ReAct-IM (best of 6)	62	68	87	57	39	33	53
BUTLER _g (best of 8)	33	26	70	76	17	12	22
BUTLER (best of 8)	46	39	74	100	22	24	37

Table 3: AlfWorld task-specific success rates (%). BUTLER and BUTLER_g results are from Table 4 of Shridhar et al. (2020b). All methods use greedy decoding, except that BUTLER uses beam search.

Method	Score	SR
Act	62.3	30.1
ReAct	66.6	40.0
IL	59.9	29.1
IL+RL	62.4	28.7
Human Expert	82.1	59.6

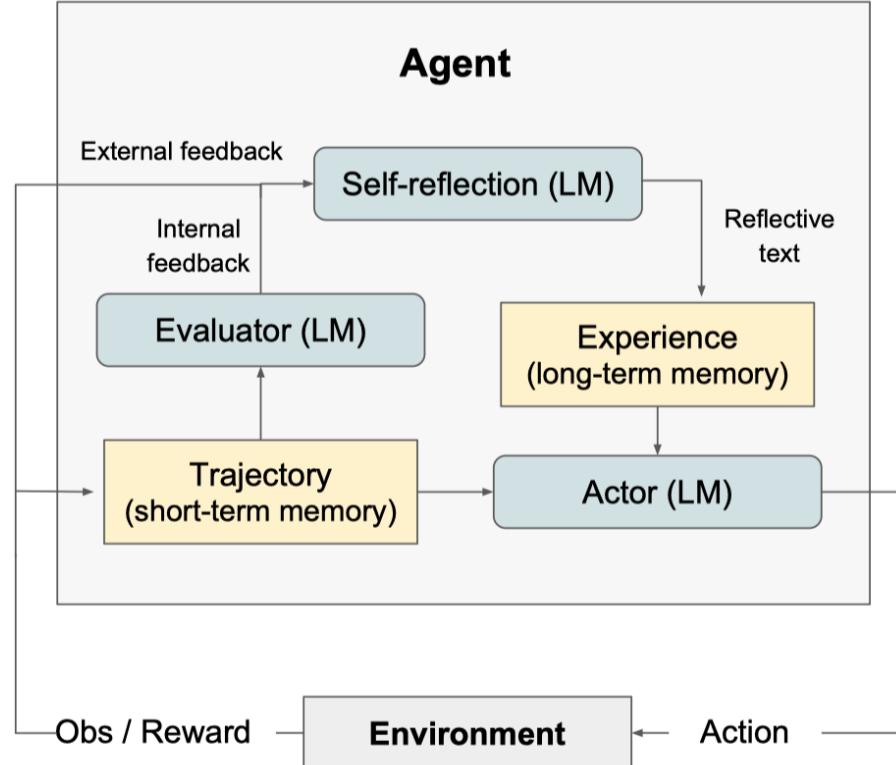
Table 4: Score and success rate (SR) on Webshop. IL/IL+RL taken from Yao et al. (2022).

Component I: Planning

19

■ Self-Reflection: Reflexion

- Reinforce language agents not by updating weights, but through linguistic feedback instead (semantic gradient)
- **Actor**: LM + memory → generate texts and actions
- **Evaluator**
 - Exact match grading
 - Pre-defined heuristic functions
 - Self-evaluation with an LLM itself
- **Self-reflection**: sparse reward signal + current trajectory + persistent memory → verbal feedback → long-term memory





Component I: Planning

20

■ Self-Reflection: Reflexion

	1. Decision making	2. Programming	3. Reasoning
(a) Task	You are in the middle of a room [...] Task: clean some pan and put it in countertop.	Task: You are given a list of two strings [...] of open '(' or close ')' parentheses only [...]	Task: What profession does John Lanchester and Alan Dean Foster have in common?
(b) Trajectory	[...] Action: take pan1 from stoveburner1 Obs: Nothing happens. [...] Action: clean pan1 with sinkbasin1 Obs: Nothing happens. [...]	def match_parens(lst): if s1.count('(') + s2.count('(') == s1.count(')') + s2.count(')'). [...] return 'No'	Think: [...] novelist, journalist, critic [...] novelist, screenwriter [...] common is novelist and screenwriter. Action: "novelist, screenwriter"
(c) Evaluation (internal / external)	Rule/LM Heuristic: Hallucination.	Self-generated unit tests fail: assert match_parens(...)	Environment Binary Reward: 0
(d) Reflection	[...] tried to pick up the pan in stoveburner 1 [...] but the pan was not in stoveburner 1. [...]	[...] wrong because it only checks if the total count of open and close parentheses is equal [...] order of the parentheses [...]	[...] failed because I incorrectly assumed that they both had the same multiple professions [...] accurately identifying their professions.
(e) Next Trajectory	[...] Action: take pan 1 from stoveburner 2 [...] Obs: You put the pan 1 in countertop 1.	[...] return 'Yes' if check(S1) or check(S2) else 'No'	Think: [...] So the profession John Lanchester and Alan Dean Foster have in common is novelist. Action: "novelist"



Outline

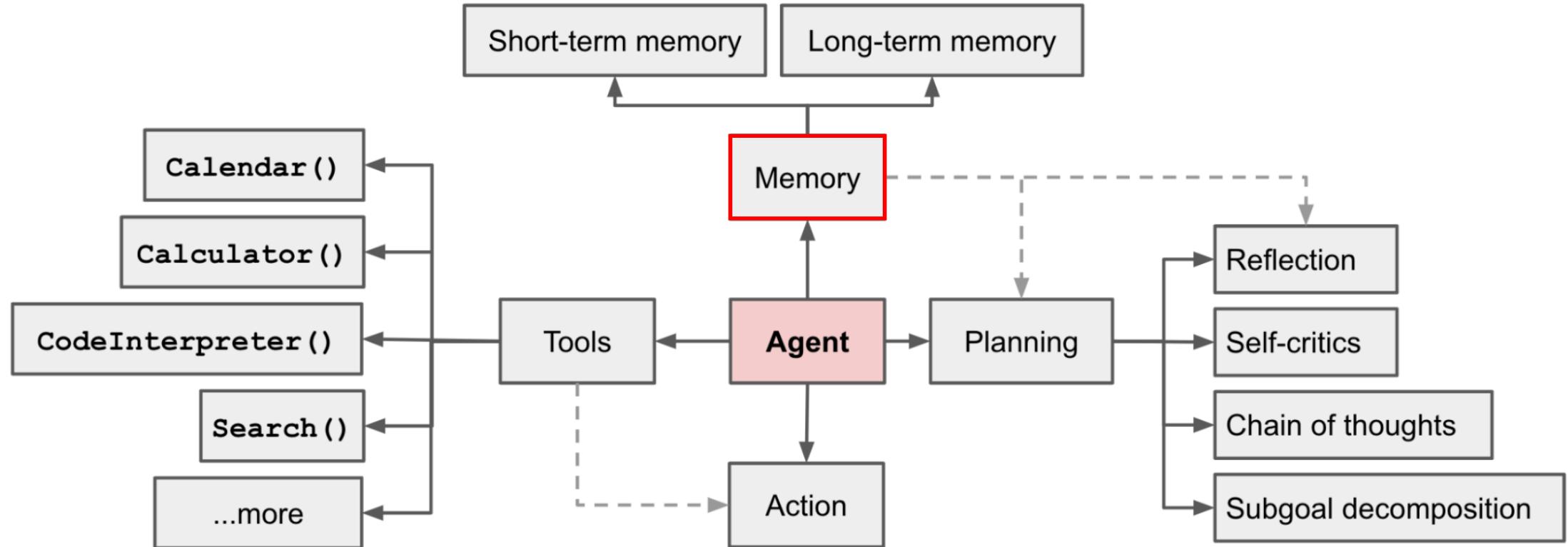
21

- Agent System Overview
- Component I: Planning
- Component II: Memory
- Component III: Tool Use
- Case Study
- Future Challenges

Component II: Memory

22

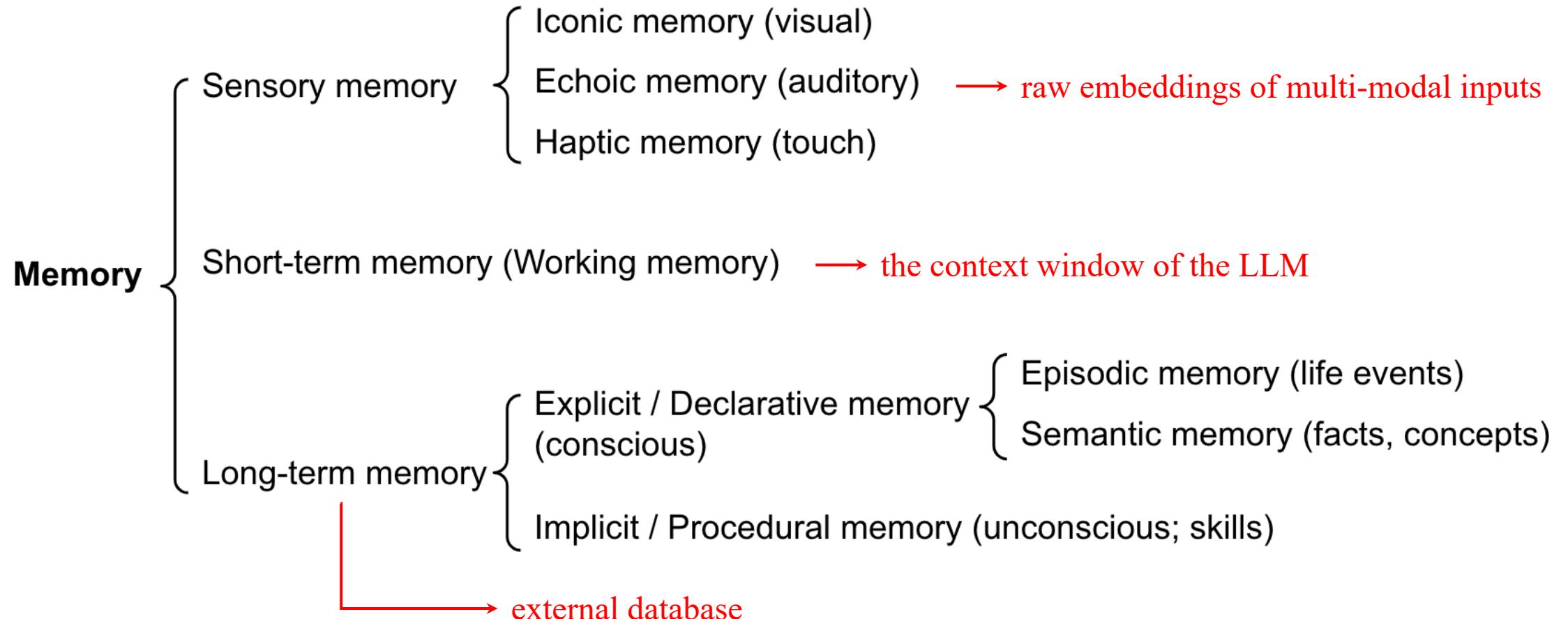
- Memory can be defined as the processes used to acquire, store, retrain, and later retrieve information.



Component II: Memory

23

■ Categorization of Human Memory

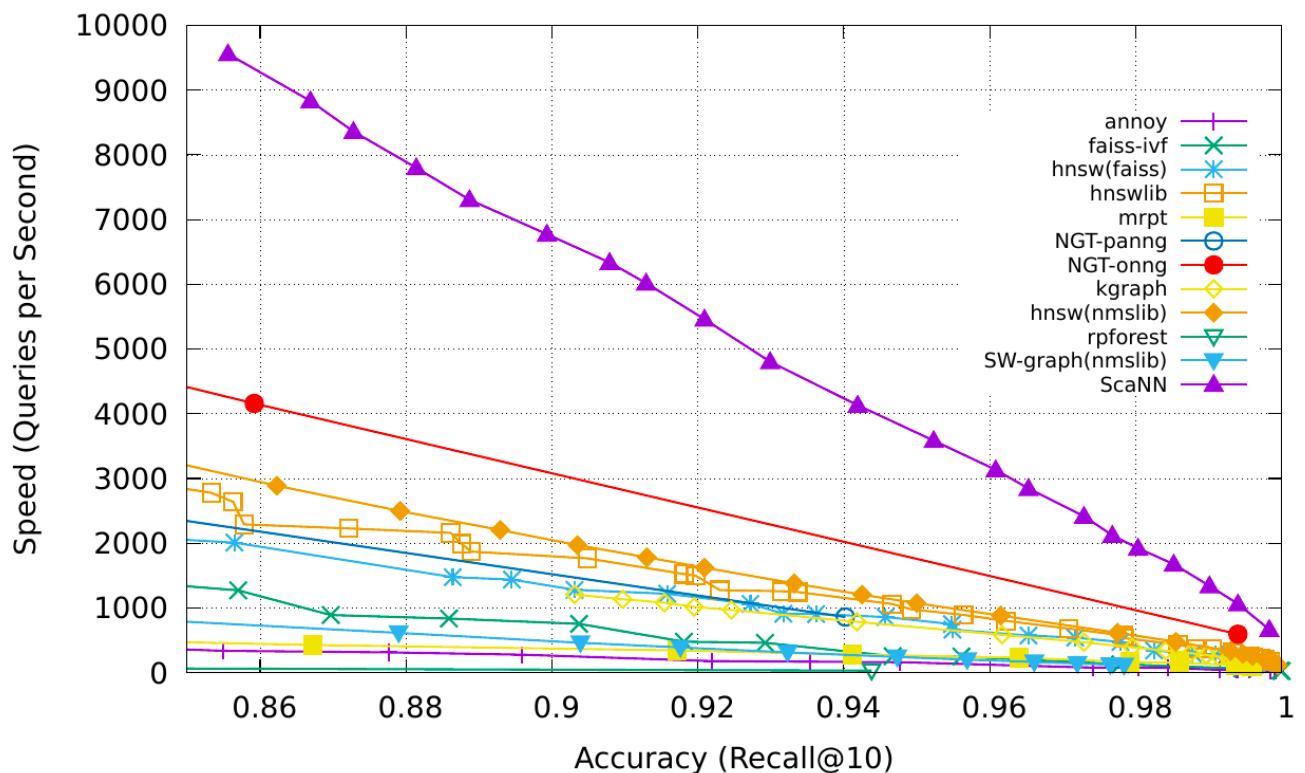


Component II: Memory

24

■ Maximum Inner Product Search (MIPS) & Approximate Nearest Neighbors (ANN)

- LSH (Locality-Sensitive Hashing)
- ANNOY (Approximate Nearest Neighbors Oh Yeah)
- HNSW (Hierarchical Navigable Small World)
- FAISS (Facebook AI Similarity Search)
- ScaNN (Scalable Nearest Search)
- ...





Outline

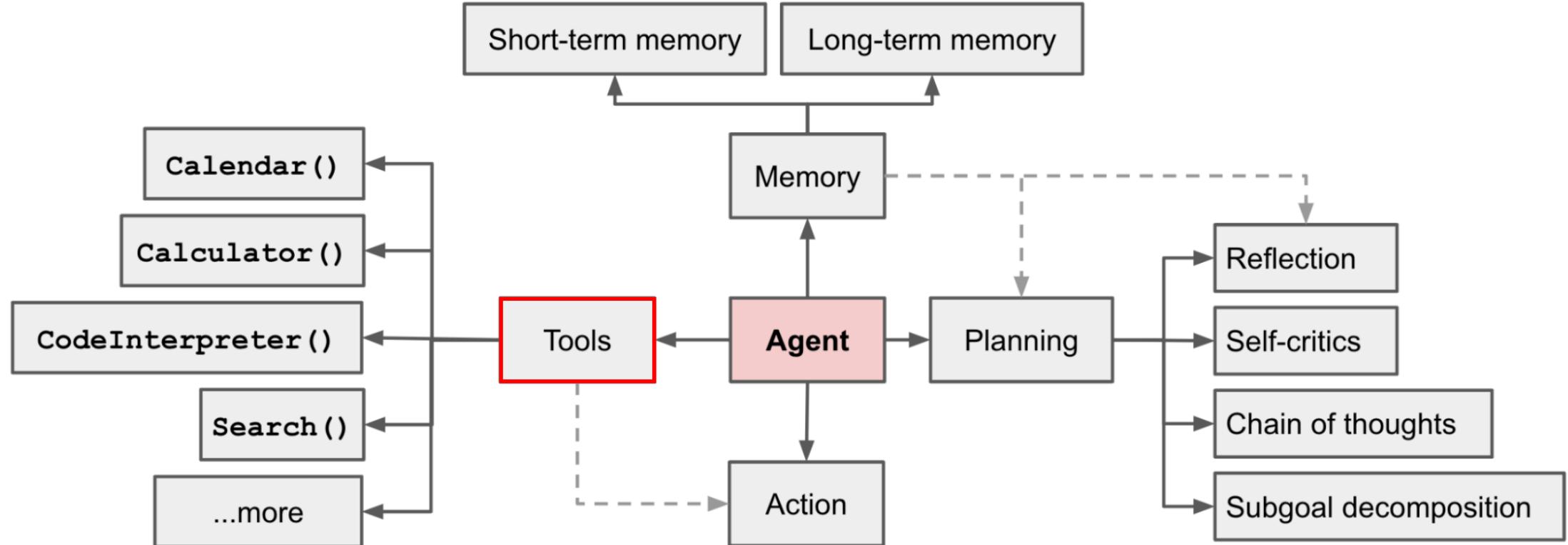
25

- Agent System Overview
- Component I: Planning
- Component II: Memory
- Component III: Tool Use
- Case Study
- Future Challenges

Component III: Tool Use

26

- Tool use is a remarkable and distinguishing characteristic of human beings.
- The capabilities of LLMs are limited but can be significantly boosted by external tools.



Component III: Tool Use

27

■ MRKL

- Use the general-purpose LLM to route inquiries to the most suitable expert module.
- The expert can be either neural (e.g., deep learning models) or symbolic (math calculator, weather API)
- Fine-tune an LLM to extract arguments from texts.
- Knowing when to and how to use the tools are crucial.

99 bottles of beer on the wall. One of them fell. How many are left?

[All](#) [Videos](#) [Images](#) [Maps](#) [News](#) [More](#)

About 20,400,000 results (0.78 seconds)

https://en.wikipedia.org/wiki/99_Bottles_of_Beer ::

[99 Bottles of Beer - Wikipedia](#)

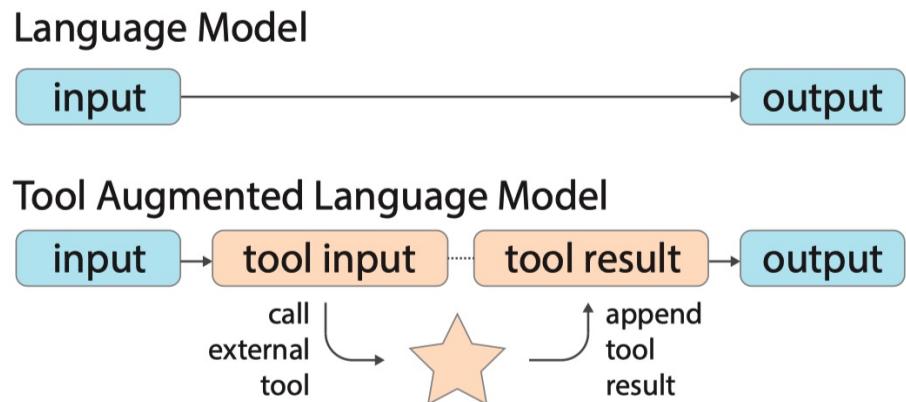


Component III: Tool Use

28

■ TALM

- Finetune the LLM to use external tool APIs.
 - Generate a tool input conditioned on the task input and invoke a tool API by generating a delimiter.
 - Call the tool API when the delimiter is detected and append the result to the text sequence.
 - Continue to generate the final task output.
- Bootstrap tool-use examples with iterative self-play.



Algorithm 1 Iterative Self-Play Algorithm.

x : task input, y : task output, t : tool input, r : tool output

```

1:  $T = \{x_i, y_i\}_T$                                 # task set
2:  $D = \{x_j, t_j, r_j, y_j\}_D$                   # tool-use set
3:  $P_\theta \leftarrow \text{pretrained LM}$ 
4: for  $t \in [0, 1, \dots, R]$  do                # self-play rounds
5:   if  $t > 0$  then                                # finetune LM
6:      $\theta \leftarrow \operatorname{argmax}_{\theta} \prod_D P_\theta(y_j|x_j, t_j, r_j)P_\theta(t_j|x_j)$ 
7:   for  $x_i, y_i \in T$  do                      # iterate task set
8:     for  $n \in [0, 1, \dots, N]$  do
9:        $t_n \leftarrow P_\theta(t|x_i)$                   # sample tool query
10:       $r_n \leftarrow \text{Tool}(t_n)$                  # call tool API
11:       $y_n \leftarrow P_\theta(y|x_i, t_n, r_n)$         # get task output
12:      if  $|y_n - y_i| < th$  then                # filter wrong output
13:         $D \leftarrow D \cup \{x_i, t_n, r_n, y_n\}_1$ 
14:    end if                                     # update tool-use set
  
```

Component III: Tool Use

29

■ Toolformer

- Prompt to annotate potential API calls via in-context learning.
- Filter annotations based on whether API calls help the model to predict future tokens.

$$L_i(\mathbf{z}) = - \sum_{j=i}^n w_{j-i} \cdot \log p_M(x_j | \mathbf{z}, x_{1:j-1})$$

API call \leftarrow $L_i^+ = L_i(e(c_i, r_i))$ API response \rightarrow
 $L_i^- = \min(L_i(\varepsilon), L_i(e(c_i, \varepsilon)))$ empty string

- Fine-tune the LLM on the annotated dataset.



Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

Input: Joe Biden was born in Scranton, Pennsylvania.

Output: Joe Biden was born in [QA("Where was Joe Biden born?")] Scranton, [QA("In which state is Scranton?")] Pennsylvania.

Input: Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

Output: Coca-Cola, or [QA("What other name is Coca-Cola known by?")] Coke, is a carbonated soft drink manufactured by [QA("Who manufactures Coca-Cola?")] the Coca-Cola Company.

Input: \mathbf{x}

Output:



Component III: Tool Use

30

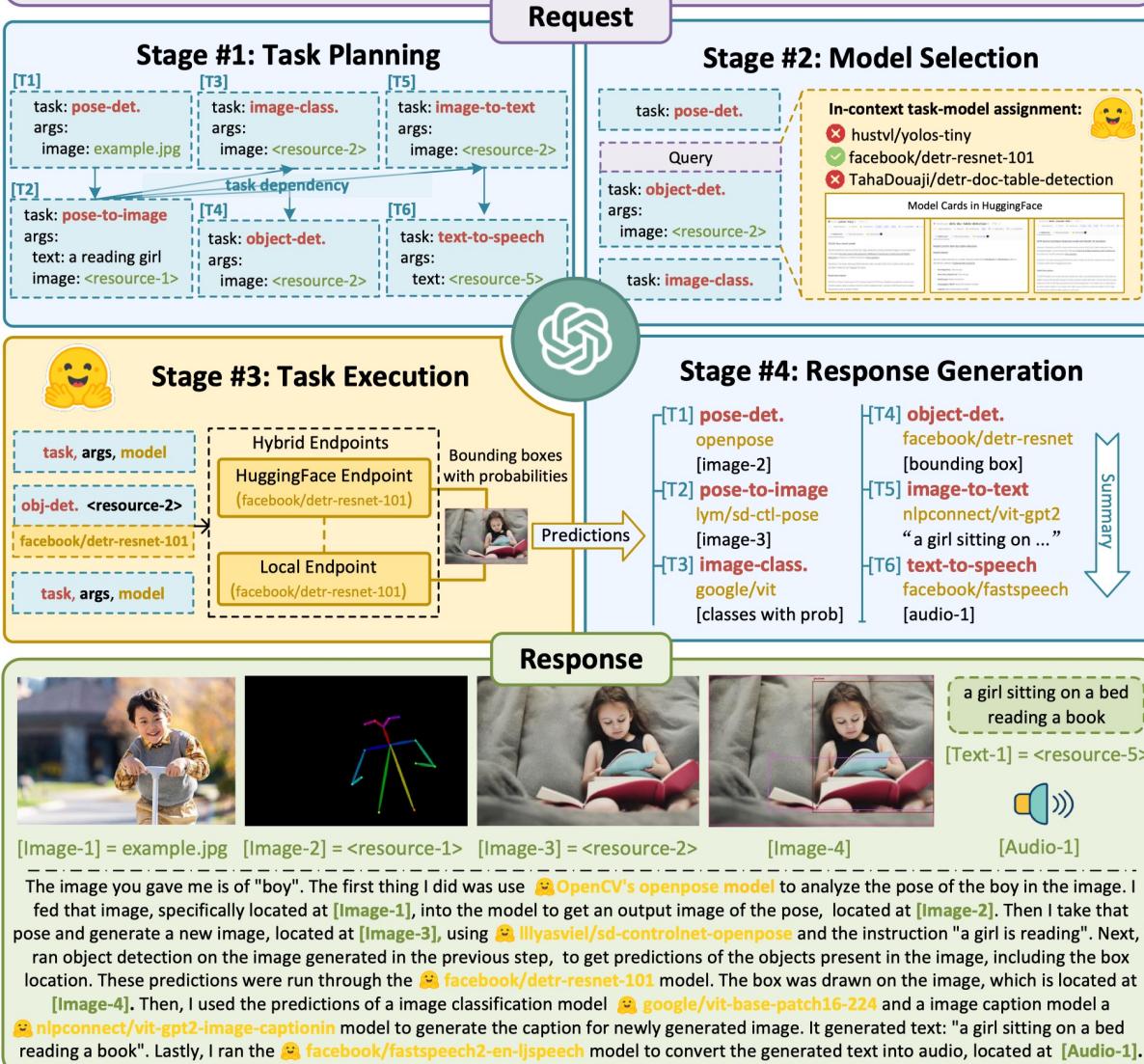
HuggingGPT

Task Planning

- Specification-based instruction
- Demonstration-based parsing
- Previous chat logs

#1 Task Planning Stage - The AI assistant performs task parsing on user input, generating a list of tasks with the following format: `[{"task": task, "id": task_id, "dep": dependency_task_ids, "args": {"text": text, "image": URL, "audio": URL, "video": URL}}]`. The "dep" field denotes the id of the previous task which generates a new resource upon which the current task relies. The tag "`<resource>-task_id`" represents the generated text, image, audio, or video from the dependency task with the corresponding task_id. The task must be selected from the following options: `{ Available Task List }`. Please note that there exists a logical connections and order between the tasks. In case the user input cannot be parsed, an empty JSON response should be provided. Here are several cases for your reference: `{ Demonstrations }`. To assist with task planning, the chat history is available as `{ Chat Logs }`, where you can trace the user-mentioned resources and incorporate them into the task planning stage.

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





Component III: Tool Use

31

HuggingGPT

Model Selection

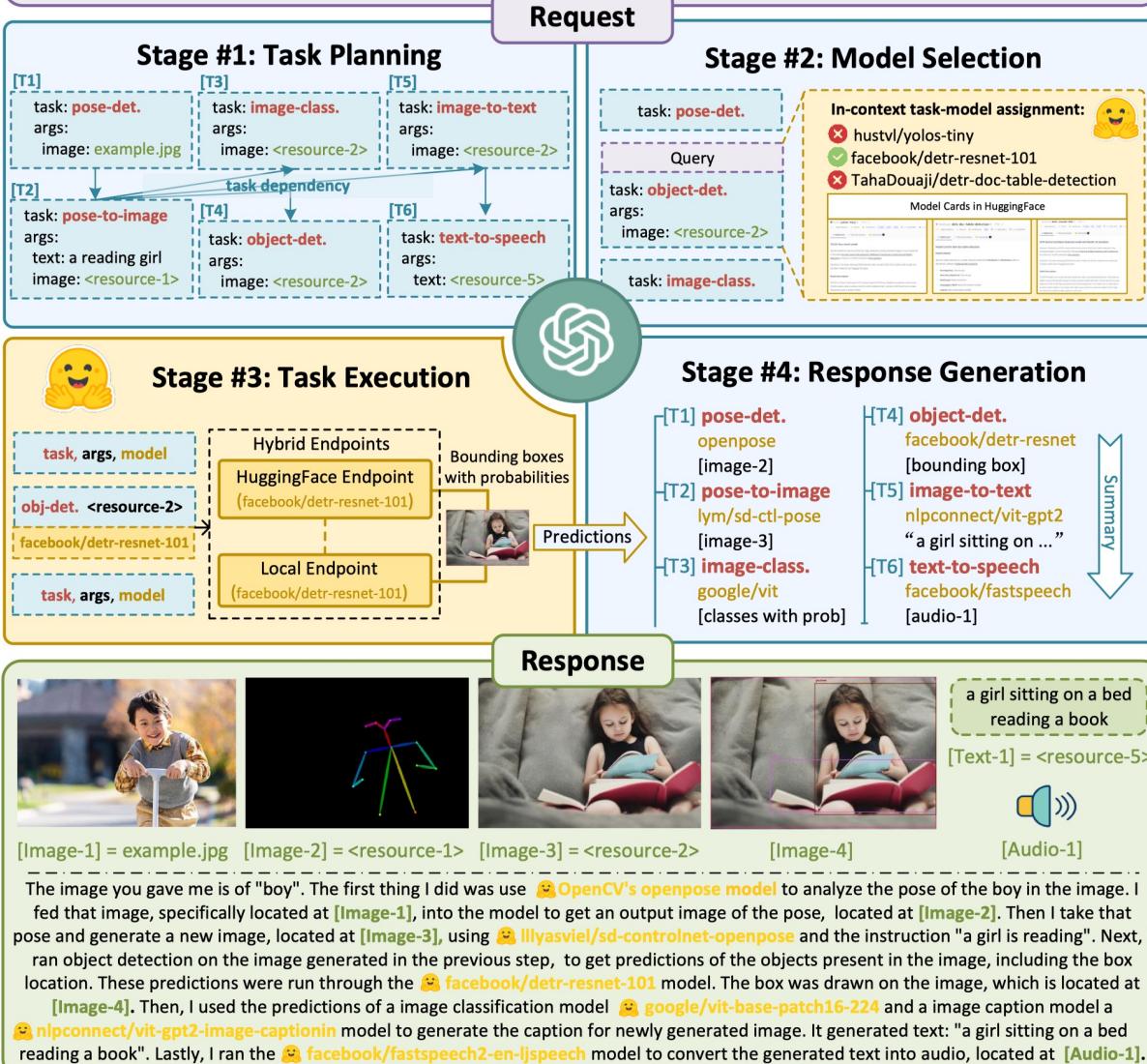
- Filter out models based on the task type.
- Rank models based on the number of downloads.
- In-context task-model assignment based on the user query, task information, model description, and metadata.

Task Execution

Response Generation

- Summarize the execution results.

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





Component III: Tool Use

32

■ ChatGPT Plugin & OpenAI API Function Calling



Expedia

Bring your trip plans to life—get there, stay there, find things to see and do.



FiscalNote

Provides and enables access to select market-leading, real-time data sets for legal, political, and regulatory data and information.



Instacart

Order from your favorite local grocery stores.



Klarna Shopping

Search and compare prices from thousands of online shops.



Milo Family AI

Giving parents superpowers to turn the manic to magic, 20 minutes each day. Ask: Hey Milo, what's magic today?



OpenTable

Provides restaurant recommendations, with a direct link to book.



Speak

Learn how to say anything in another language with Speak, your AI-powered language tutor.



Wolfram

Access computation, math, curated knowledge & real-time data through Wolfram|Alpha and Wolfram Language.



Zapier

Interact with over 5,000+ apps like Google Sheets, Trello, Gmail, HubSpot, Salesforce, and more.

```
{
  "schema_version": "v1",
  "name_for_human": "TODO Manager",
  "name_for_model": "todo_manager",
  "description_for_human": "Manages your TODOs!",
  "description_for_model": "An app for managing a user's TODOs",
  "api": { "url": "/openapi.json" },
  "auth": { "type": "none" },
  "logo_url": "https://example.com/logo.png",
  "legal_info_url": "http://example.com",
  "contact_email": "hello@example.com"
}
```

```
openapi: 3.0.1
info:
  title: TODO Plugin
  description: A plugin that allows the user to create and manage a TODO list using ChatGPT.
  version: 'v1'
servers:
  - url: https://example.com
paths:
  /todos:
    get:
      operationId: getTodos
      summary: Get the list of todos
      responses:
        "200":
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/getTodosResponse'
components:
  schemas:
    getTodosResponse:
      type: object
      properties:
        todos:
          type: array
          items:
            type: string
            description: The list of todos.
```



Outline

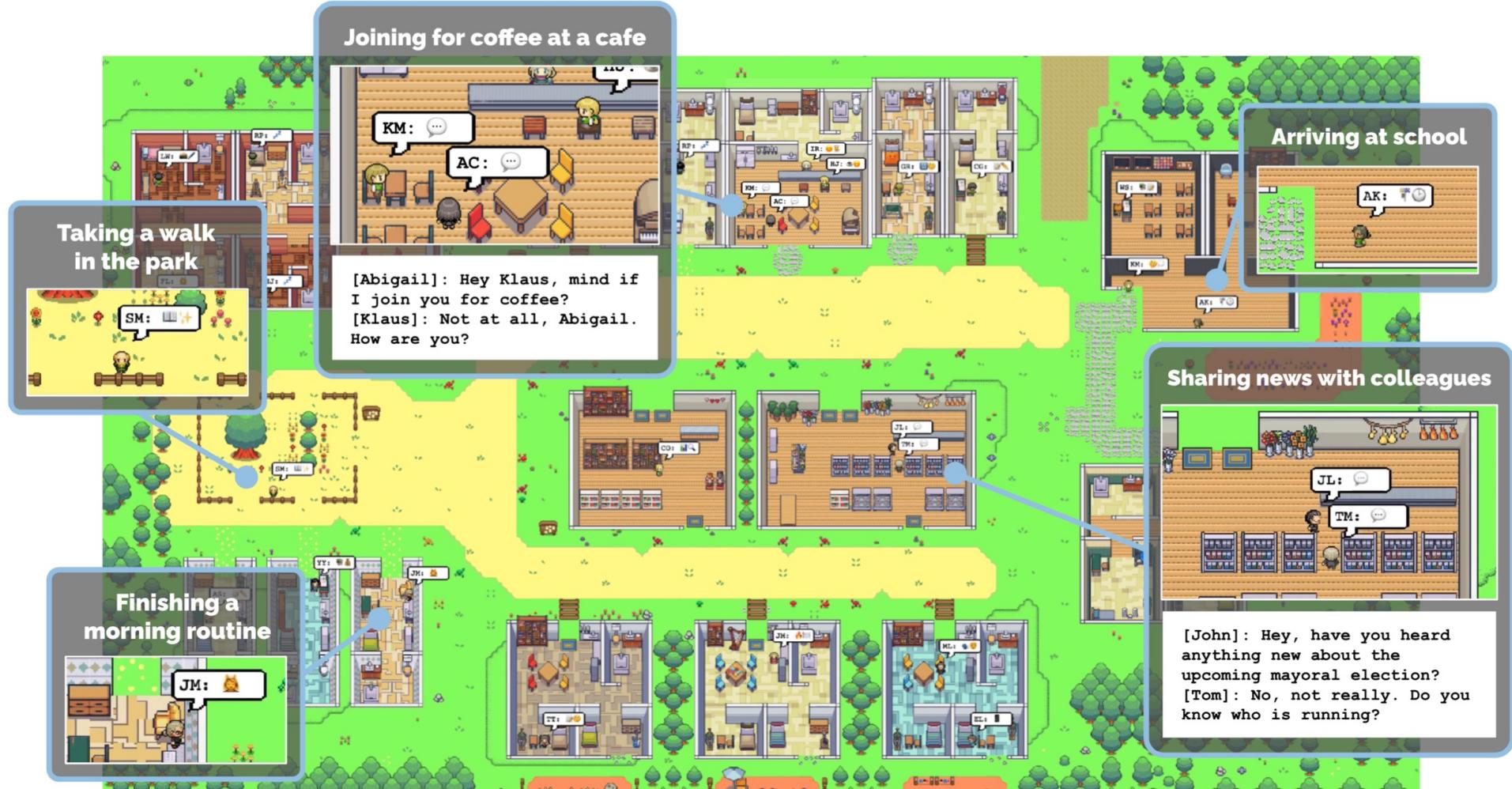
33

- Agent System Overview
- Component I: Planning
- Component II: Memory
- Component III: Tool Use
- Case Study
- Future Challenges

Case Study

34

■ Generative Agents

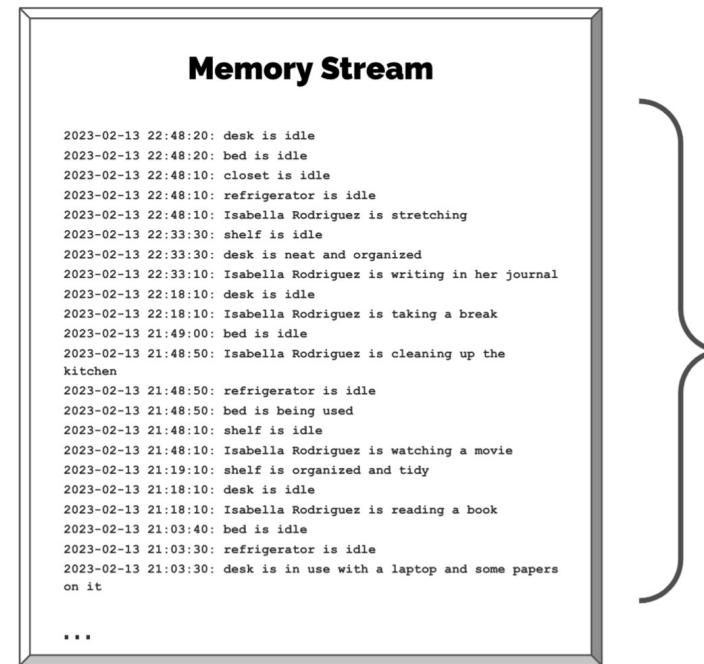


Case Study

35

■ Generative Agents

- **Memory stream:** record a comprehensive list of agents' experiences in natural language (e.g., observation, reflection, plan)
- **Retrieval function:** recency + importance (LM evaluated) + relevance (text embedding similarity)



Q. What are you looking forward to the most right now?

Isabella Rodriguez is excited to be planning a Valentine's Day party at Hobbs Cafe on February 14th from 5pm and is eager to invite everyone to attend the party.			
retrieval	recency	importance	relevance
2.34	=	0.91	+ 0.63 + 0.80
ordering decorations for the party			
2.21	=	0.87	+ 0.63 + 0.71
researching ideas for the party			
2.20	=	0.85	+ 0.73 + 0.62
...			



I'm looking forward to the Valentine's Day party that I'm planning at Hobbs Cafe!



Case Study

36

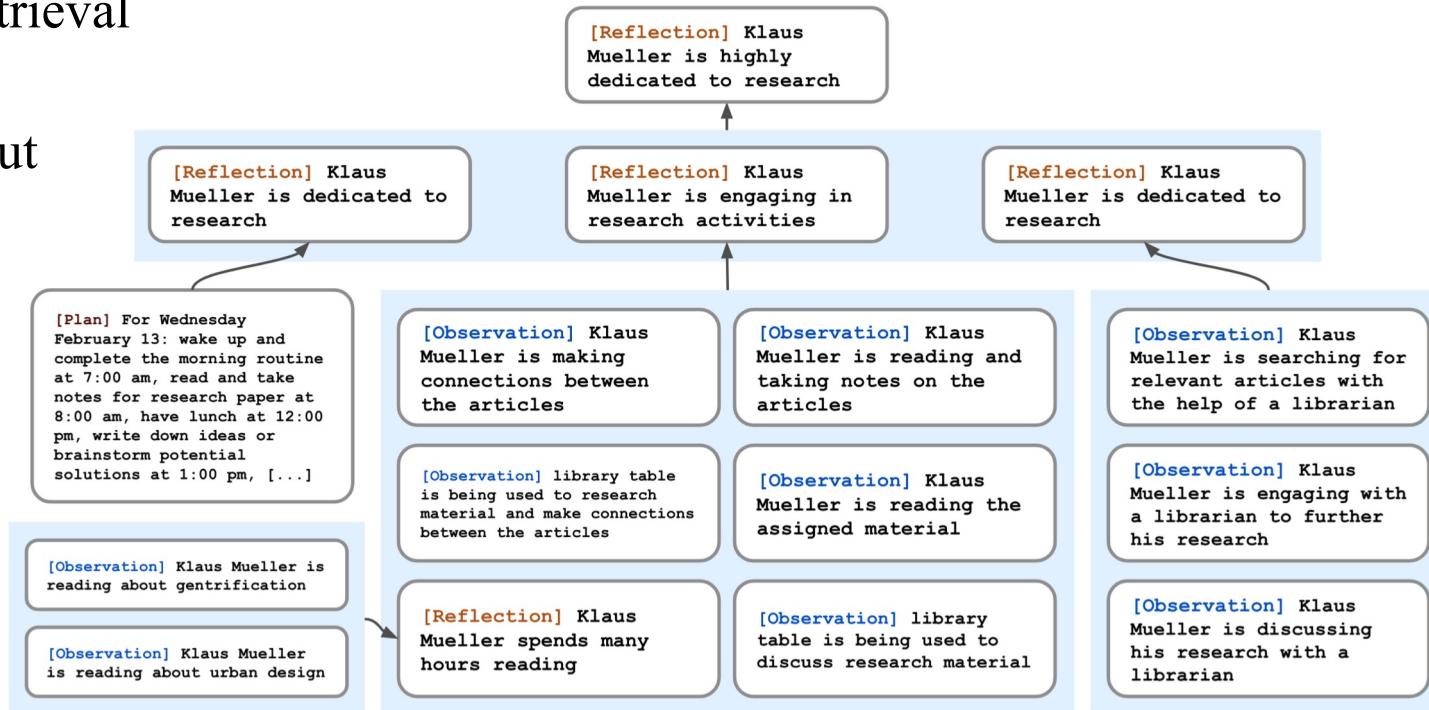
■ Generative Agents

□ **Reflection:** synthesize memories into higher-level inferences and draw conclusions.

➤ Prompt the LM with 100 most recent records to generate 3 most salient high-level questions.

“Given only the information above, what are 3 most salient high-level questions we can answer about the subjects in the statements?”

- Use the questions as queries for retrieval and extract insights with the LM.
- Reflect not only on observations but also on other reflections
→ tree of reflections



Case Study

37

■ Generative Agents

□ Planning and Reacting

- Generate plans in a top-down manner based on the agent's description and a summary of the previous day.
- Decide whether they should continue with their existing plan or react based on the agent's description, current observation, and the context summary.

1) wake up and complete the morning routine at 8:00 am, 2) go to Oak Hill College to take classes starting 10:00 am, [. . .] 5) work on his new music composition from 1:00 pm to 5:00 pm, 6) have dinner at 5:30 pm, 7) finish school assignments and go to bed by 11:00 pm.

work on his new music composition from 1:00 pm to 5:00 pm becomes 1:00 pm: start by brainstorming some ideas for his music composition [...] 4:00 pm: take a quick break and recharge his creative energy before reviewing and polishing his composition.

4:00 pm: grab a light snack, such as a piece of fruit, a granola bar, or some nuts. 4:05 pm: take a short walk around his workspace [...] 4:50 pm: take a few minutes to clean up his workspace.

Large chunks



Hourly



5 ~ 15 minutes

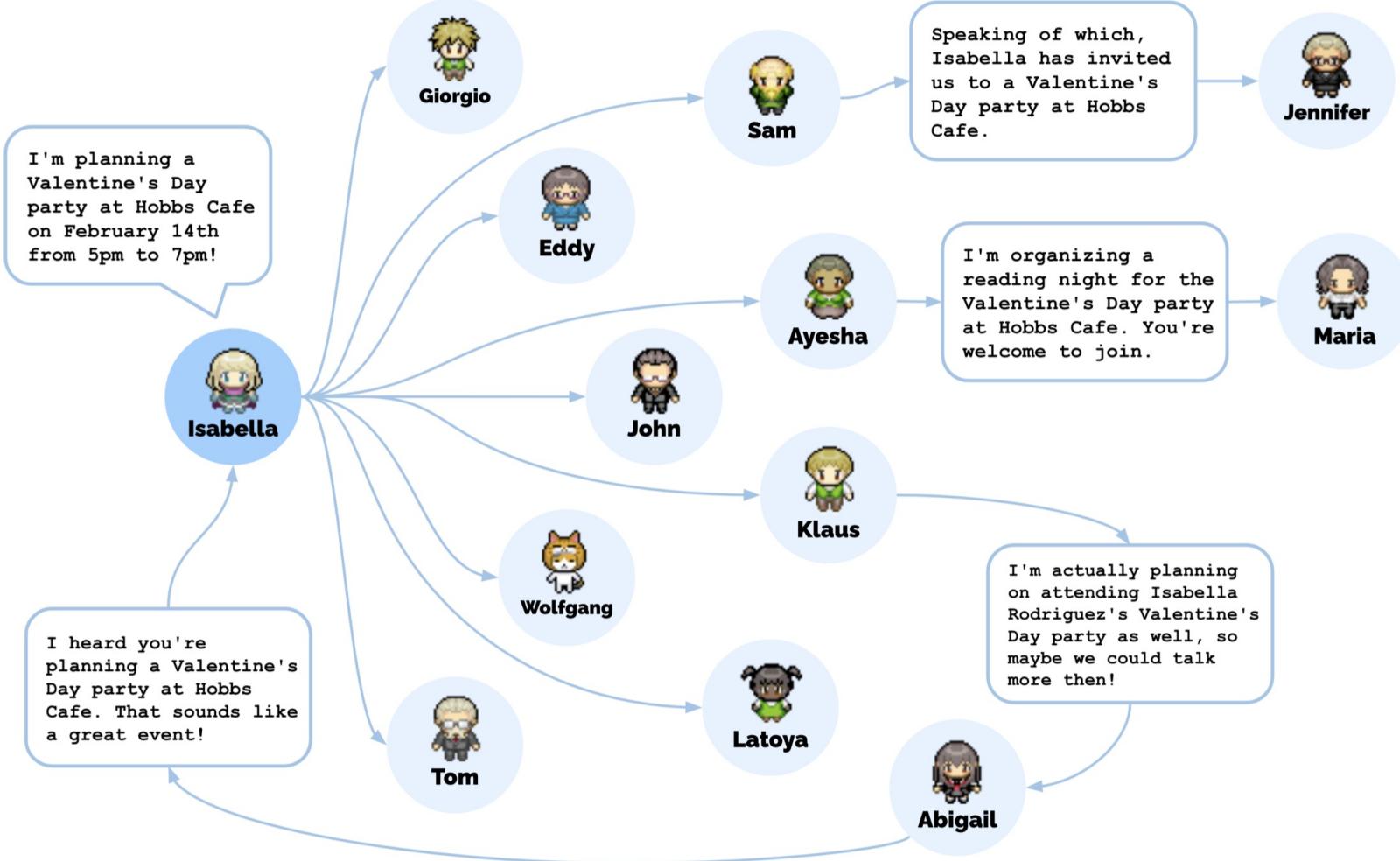
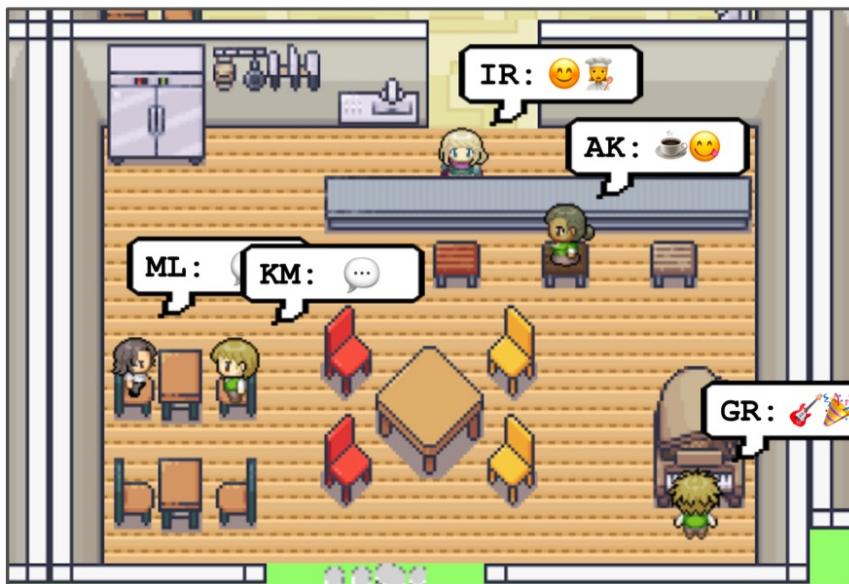
Case Study

38

■ Generative Agents

□ Emergent social behaviors

- Information diffusion
- Relationship formation
- Agents coordination





Outline

39

- Agent System Overview
- Component I: Planning
- Component II: Memory
- Component III: Tool Use
- Case Study
- Future Challenges



Future Challenges

40

- Prompt engineering
- Finite context length
- Long-term planning and task decomposition
- Reliability of natural language interface
- ...



Reference

41

- Weng et al. LLM-powered Autonomous Agents. *Lil'Log* 2023.
- Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *NIPS* 2022.
- Kojima et al. Large Language Models are Zero-Shot Reasoners. *NIPS* 2022.
- Zhou et al. Large Language Models Are Human-Level Prompt Engineers. *ICLR* 2022.
- Yao et al. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *arXiv preprint* 2023.
- Liu et al. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. *arXiv preprint* 2023.
- Yao et al. ReAct: Synergizing Reasoning and Acting in Language Models. *ICLR* 2023.
- Shinn et al. Reflexion: Language Agents with Verbal Reinforcement Learning. *NIPS* 2023.
- Karpas et al. MRKL Systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint* 2022.
- Parisi et al. TALM: Tool Augmented Language Models. *arXiv preprint* 2022.
- Schick et al. Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv preprint* 2023.
- Shen et al. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face. *arXiv preprint* 2023.
- Park et al. Generative Agents: Interactive Simulacra of Human Behavior. *UIST* 2023.



Thanks!

Q&A