PAPER PRESENTATION

Paper: EXPLORING LARGE LANGUAGE MODEL BASED INTELLIGENT AGENTS: DEFINITIONS, METHODS, AND PROSPECTS(https://arxiv.org/pdf/2401.03428.pdf)

Speaker: Jianbang Zhang

Mail: whdx072018@foxmail.com

Date: 2024-03-14

Topic
A SURVEY OF Autonomous Agent

- PREFACE: Introduction and Methodology
- 1.Definition
- 2.Development and History
- 3.Category
- 4.Structures
- 5.Training or Tuning
- 6.Evaluation
- 7.Prospect

PREFACE:
Introduction and Methodology

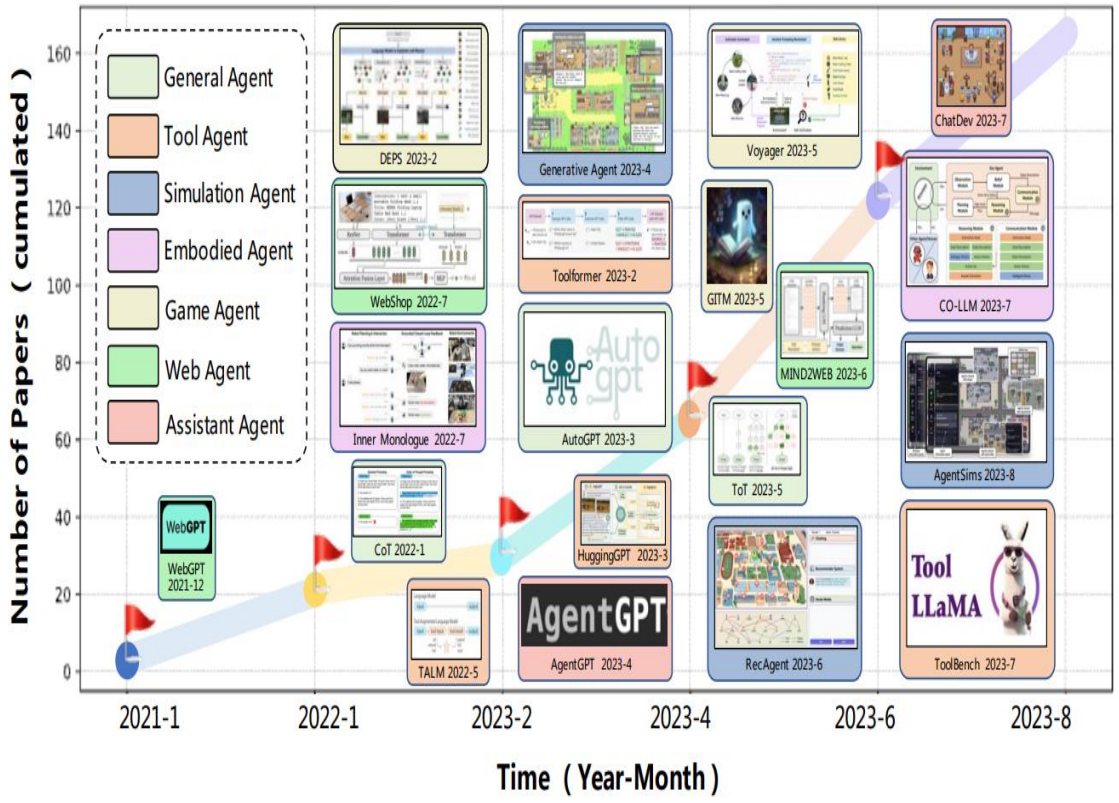
- Three Steps reading method:

STEP ONE: Read the Abstract and Conclusion: Understand the problem the author is investigating, the results or contributions made, and any limitations or future prospects mentioned by the author.

STEP TWO: Carefully Study Figures and Tables:Figures and tables represent the core design ideas of the author and relevant experimental comparisons.

STEP THREE: Review the Research Methods and Background: Finally, delve into the research methods employed and the background information provided in the paper.

PREFACE:
Introduction and Methodology



- Methodology

- Critical thinking: Read the paper with questions and skepticism, make it clear how the author presents their viewpoint, and grasp the experimental methods used in the research.

- Paper positioning: It involves both horizontal and vertical comparisons. Horizontally, one examines the author's sources of inspiration and ideas. Vertically, the focus shifts to how the author's paper advances existing research. By considering the timeline and parallel axes, one can comprehensively assess the paper's positioning.

- Special Attention: Take note of the papers repeatedly mentioned by the author.

(Referred from `A Survey on Large Language Model based Autonomous Agents`)

1. Definition

What is the autonomous Agent?

- From Reinforcement Learning's perspective:

  The "agent" refers to the entity that interact with the environment dynamically.This process could be defined as `Markov Decision Process`,that is

  (state,action,observation,transition,reword,state,...).

  Key Premise:

  Support that state,space and action space and observation space are both observable.

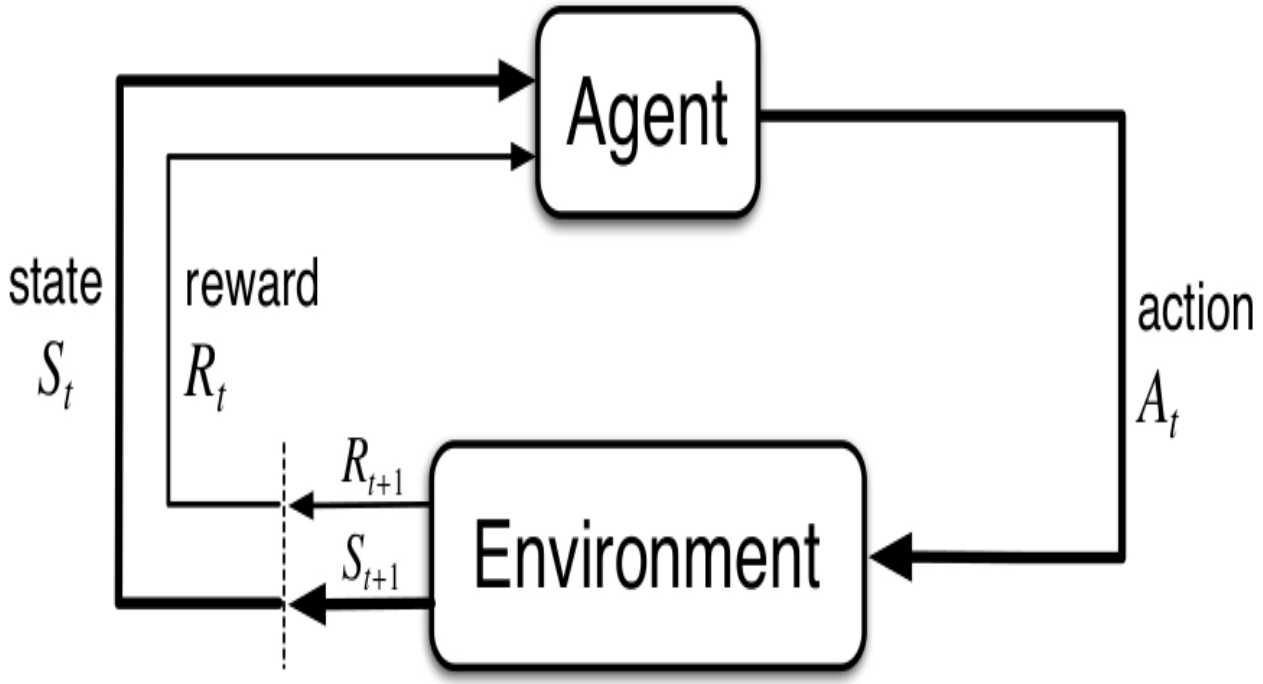
- From LLM's Perspective:

  LLM-based agent:The interactive evaluation of LLM-as-Agent could be regarded as a Partially Observable Markov Decision Process (S, A, T , R, U, O), which comprises state space S, action space A, transition function $T : S \times A \rightarrow S$, reward assigning function R, task instruction space U, and observation space O.

  (Referred from `AGENTBENCH: EVALUATING LLMS AS AGENTS`)

1. Definition

What is the autonomous Agent?



(referred from `Generative Agents: Interactive Simulacra of Human Behavior`)

2.Development and History

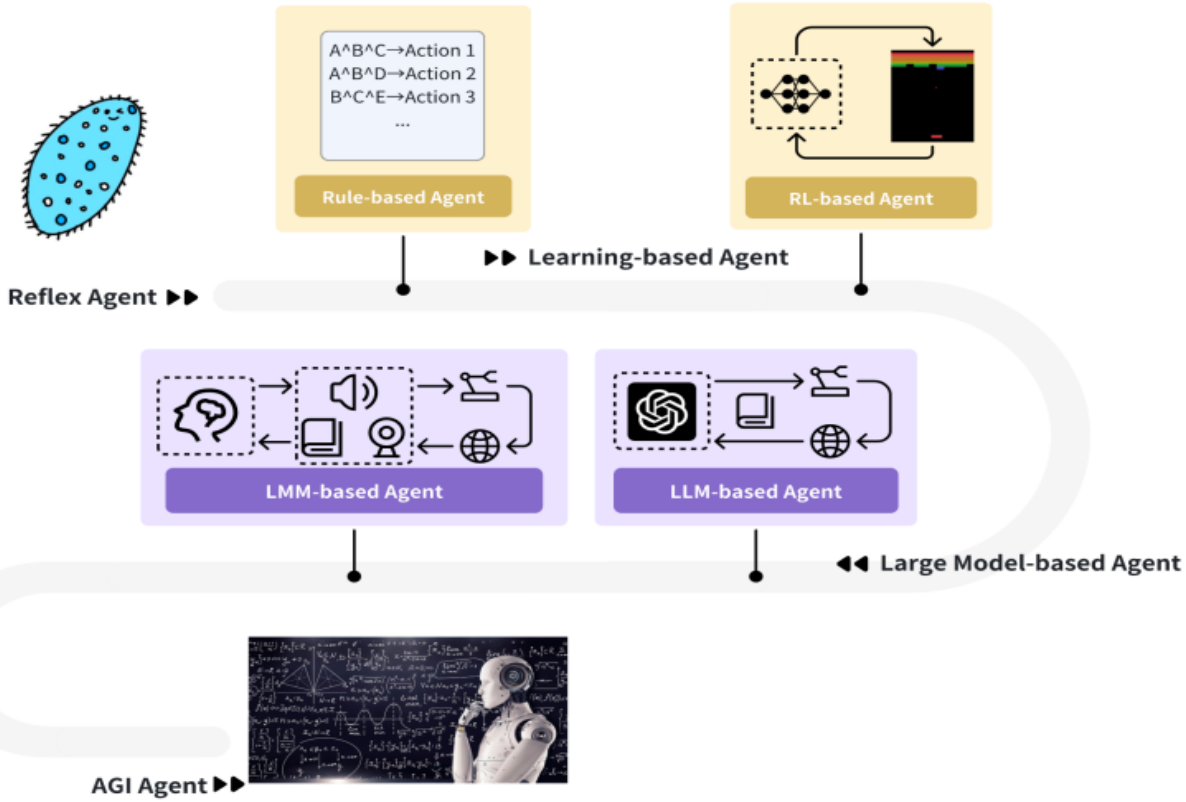


Figure 1: Roadmap of Intelligent Agents Development

2.Development and History

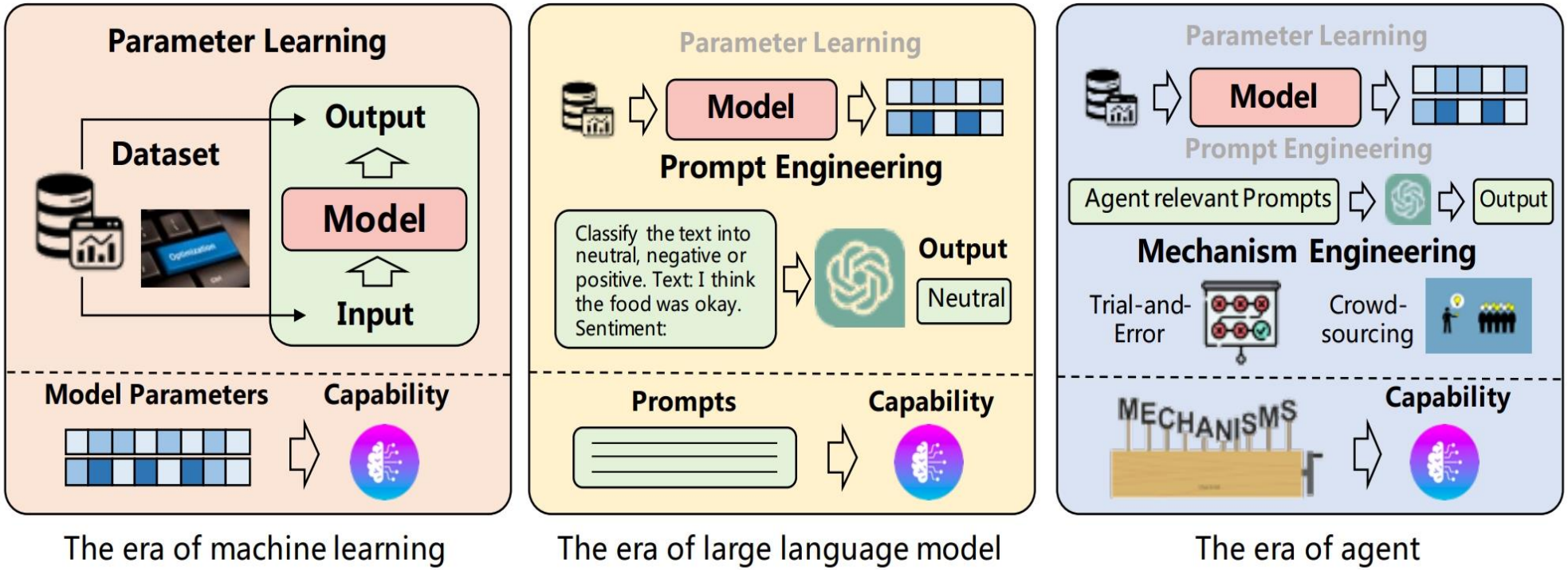


Fig. 4 Illustration of transitions in strategies for acquiring model capabilities.

(Referred from `A Survey on Large Language Model based Autonomous Agents`)

3.Category

- Different VIEWS

VIEWS	CATEGORIED	REFERRED PAPER
SYSTEM MECHANISM	➢ Single-Agent System Tool Environment ➢ Multi-Agent System Relation: Cooperative, Competitive, Mixed, Hierarchical Plan and Exec: CPDE DPDE	EXPLORING LARGE LANGUAGE MODEL BASED INTELLIGENT AGENTS: DEFINITIONS, METHODS, AND PROSPECTS
TOOL CATEGORY	➢ Physical Interaction-based Tool ➢ GUI-based Tools ➢ Program-based Tools	Tool Learning with Foundation Models

4.Structures

OVERVIEW:

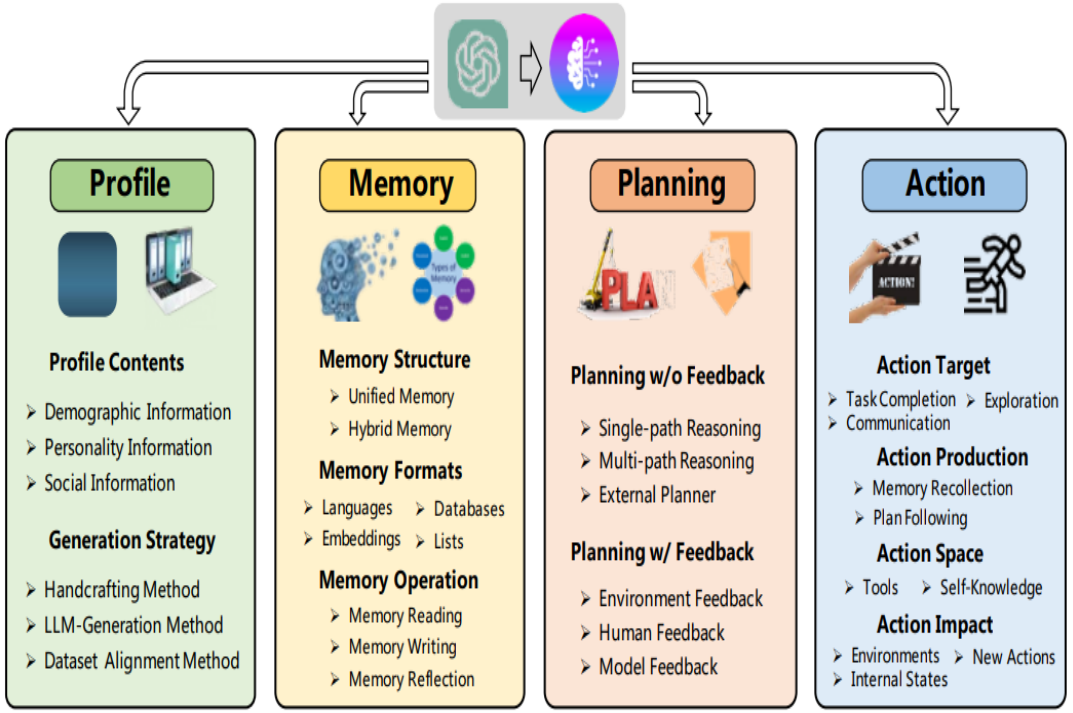


Figure 2: A unified framework for the architecture design of LLM-based autonomous agent.

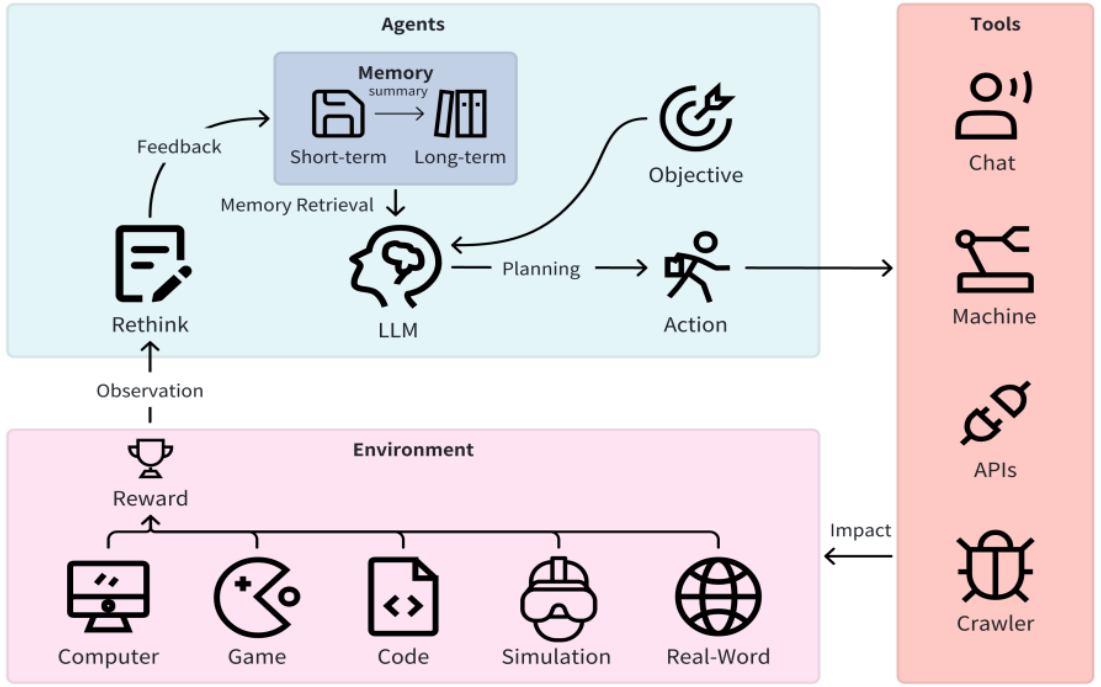


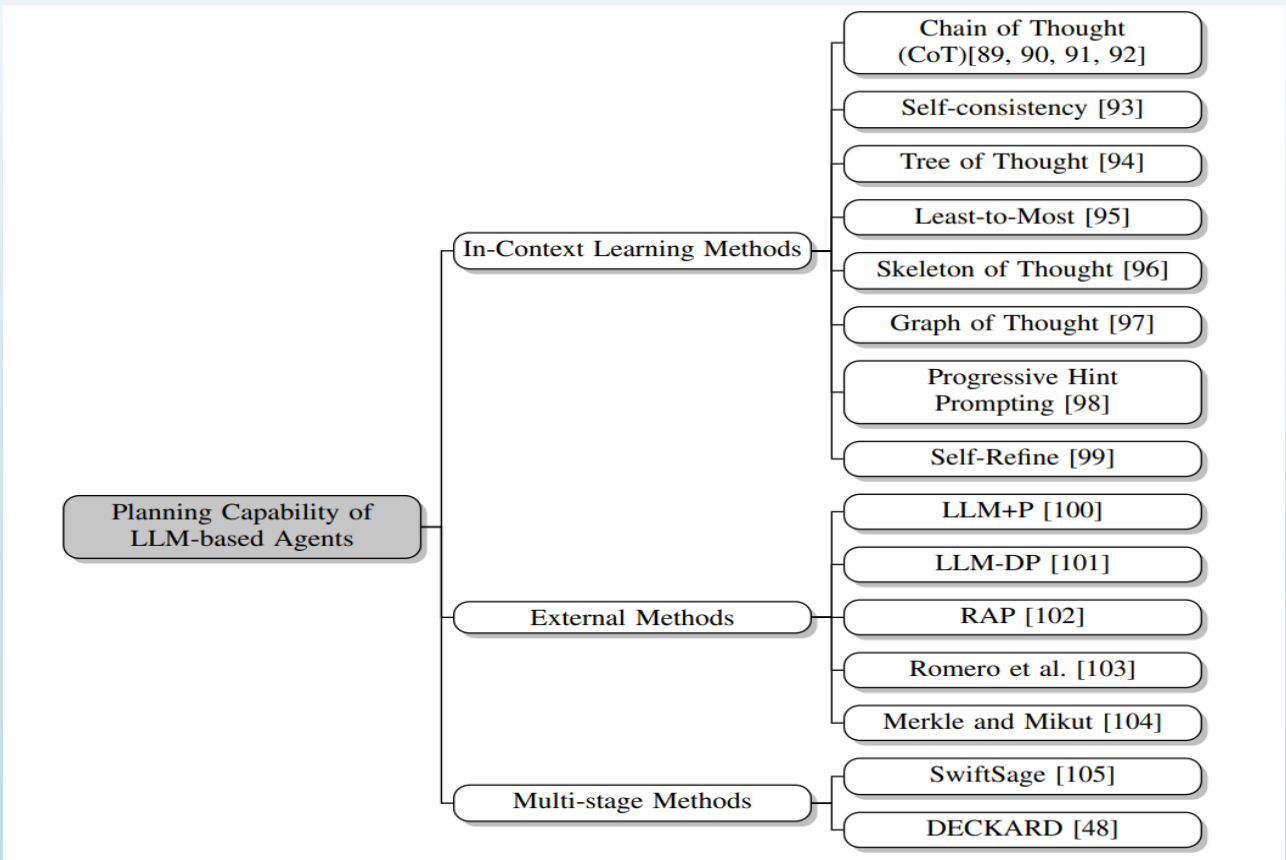
Figure 2: Overview of LLM-based agents

(Referred from `A Survey on Large Language Model based Autonomous Agents`)

4-1 PLAN

PLAN:a scheme or method of acting, doing, proceeding, making, etc., developed in advance: battle plans. a design or scheme of arrangement: an elaborate plan for seating guests.

It’ s a vital feature of LLM-based agents, encompassing task analysis, potential action anticipation, optimal action selection, and the ability to tackle complex problems and tasks.



4-1 PLAN
CHAIN OF THOUGHT/SELF-CONSISTANCY

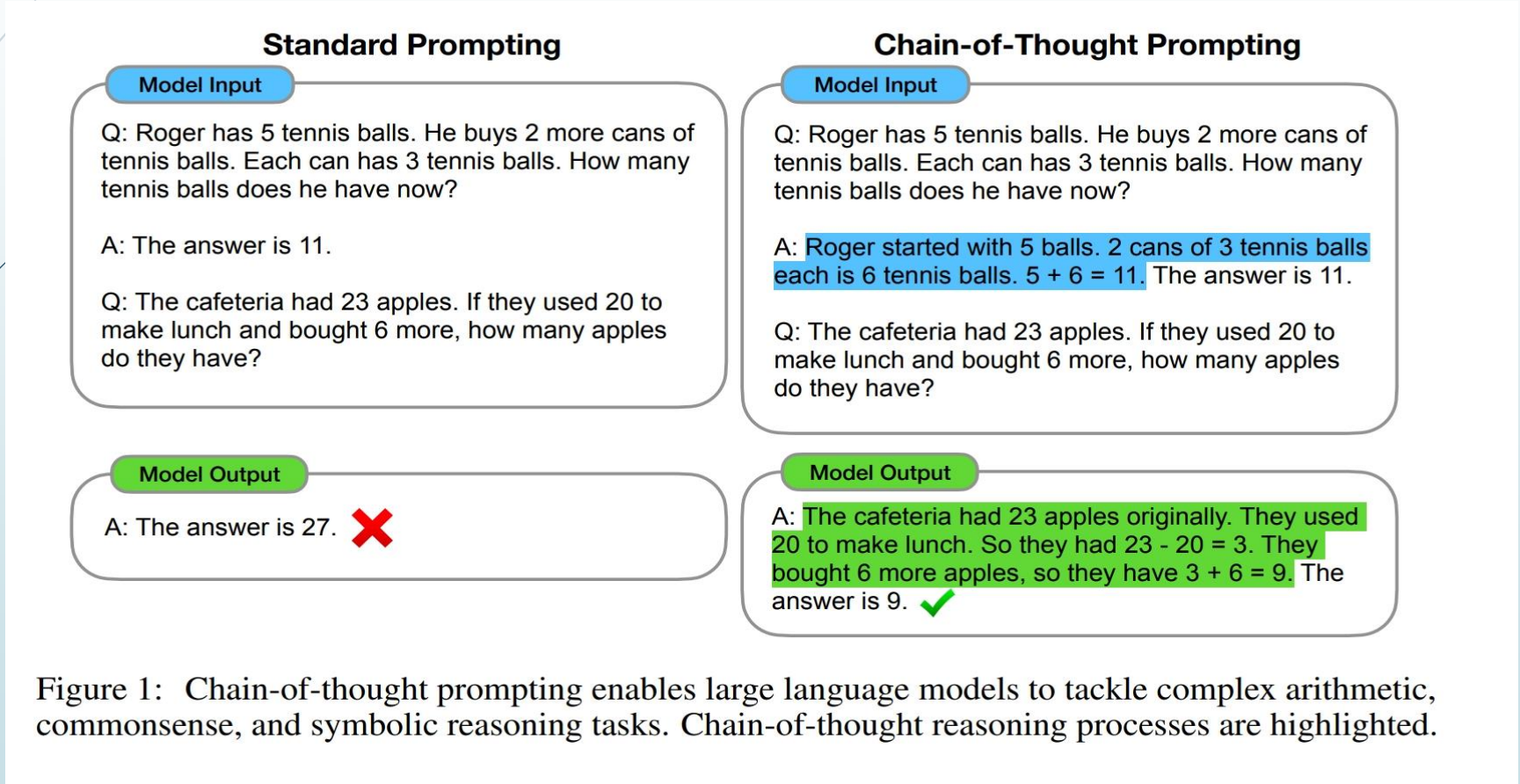


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

(Referred from `Chain-of-Thought Prompting Elicits Reasoning in Large Language Models`)

4-1 PLAN
ReAct

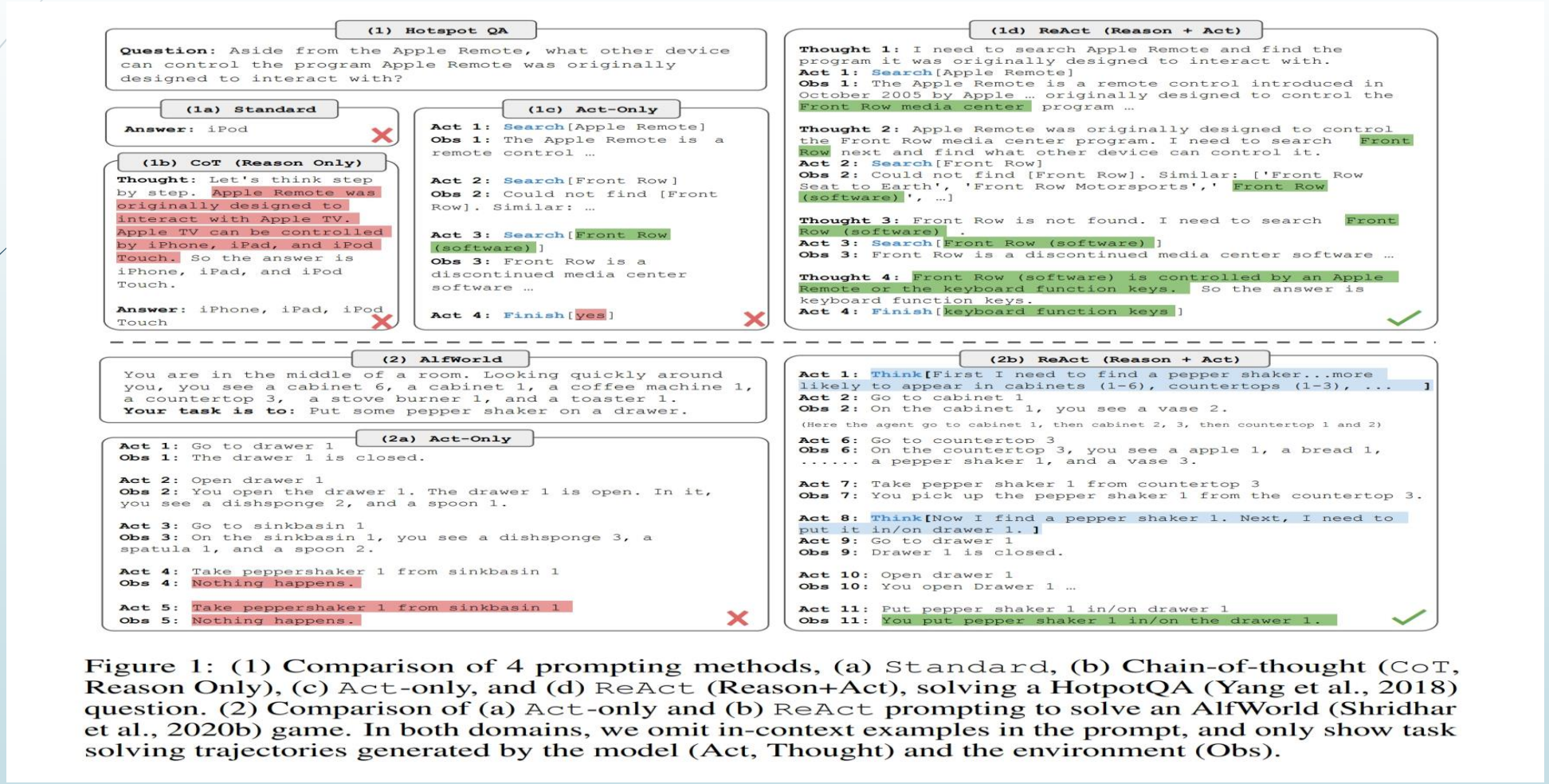


Figure 1: (1) Comparison of 4 prompting methods, (a) Standard, (b) Chain-of-thought (CoT, Reason Only), (c) Act-only, and (d) ReAct (Reason+Act), solving a HotpotQA (Yang et al., 2018) question. (2) Comparison of (a) Act-only and (b) ReAct prompting to solve an AlfWorld (Shridhar et al., 2020b) game. In both domains, we omit in-context examples in the prompt, and only show task solving trajectories generated by the model (Act, Thought) and the environment (Obs).

(Referred from `REAC T: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS`)

4-1 PLAN TOT

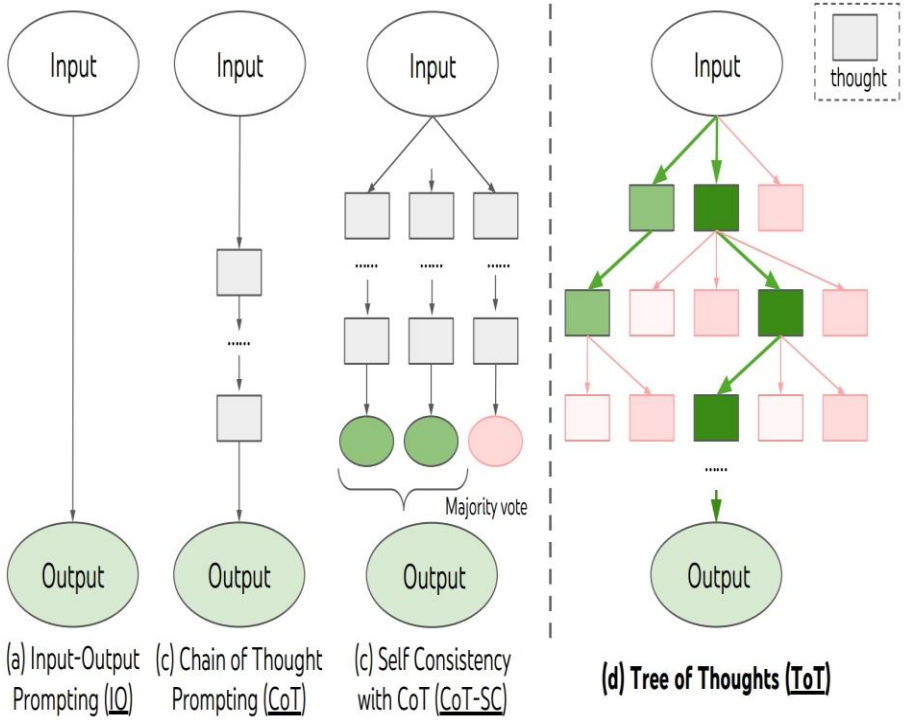
Algorithm 1 ToT-BFS$(x, p_\theta, G, k, V, T, b)$

Require: Input x, LM p_θ, thought generator $G()$ & size limit k, states evaluator $V()$, step limit T, breadth limit b.

$S_0 \leftarrow \{x\}$
for $t = 1, \cdots, T$ **do**
  $S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in \mathrm{G}(p_\theta, s, k)\}$
  $V_t \leftarrow V(p_\theta, S'_t)$
  $S_t \leftarrow \arg\max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$
end for
return $G(p_\theta, \arg\max_{s \in S_T} V_T(s), 1)$

Algorithm 2 ToT-DFS$(s, t, p_\theta, G, k, V, T, v_{th})$

Require: Current state s, step t, LM p_θ, thought generator $G()$ and size limit k, states evaluator $V()$, step limit T, threshold v_{th}
if $t > T$ **then** record output $G(p_\theta, s, 1)$
end if
for $s' \in G(p_\theta, s, k)$ **do** ▷ sorted candidates
  if $V(p_\theta, \{s'\})(s) > v_{thres}$ **then** ▷ pruning
    DFS$(s', t+1)$
  end if
end for



Game of 24 is a mathematical reasoning challenge, where the goal is to use 4 numbers and basic arithmetic operations (+-*/) to obtain 24. For example, given input "4 9 10 13", a solution output could be "(10 - 4) * (13 - 9) = 24".

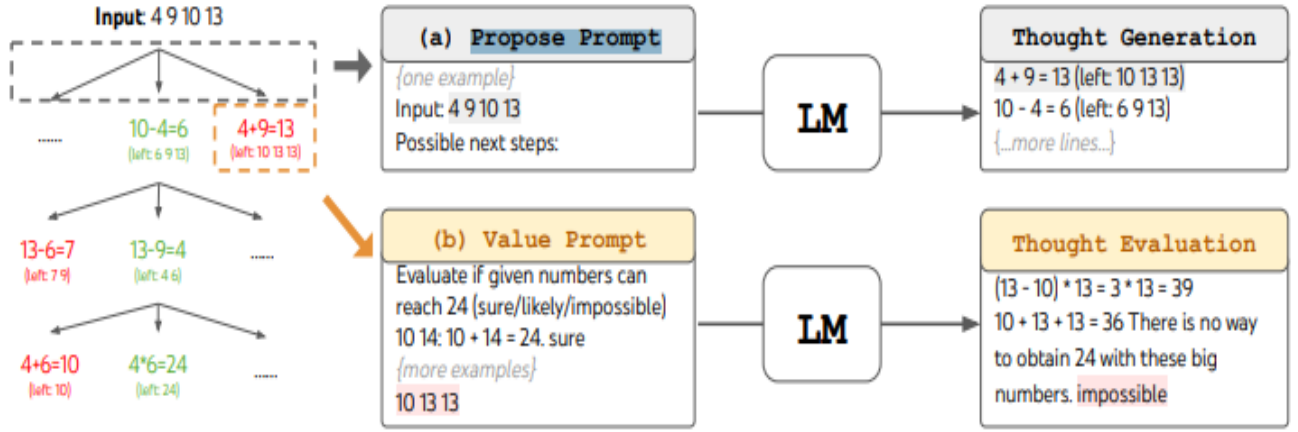
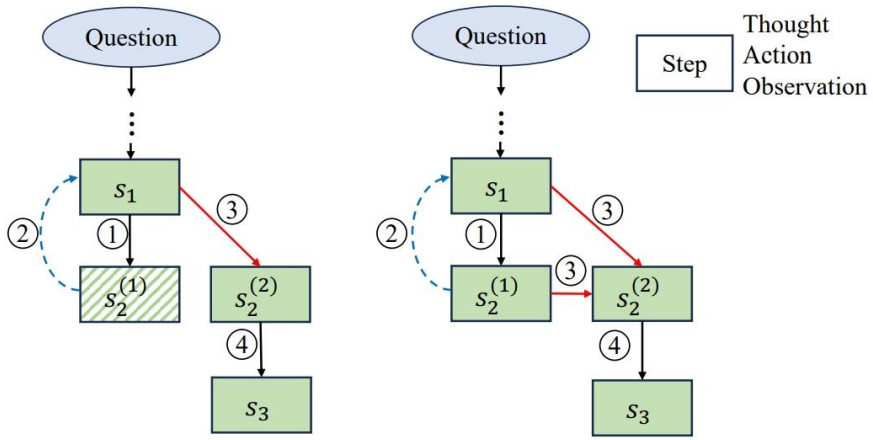


Figure 2: ToT in a game of 24. The LM is prompted for (a) thought generation and (b) valuation.

(Referred from `Tree of Thoughts: Deliberate Problem Solving with Large Language Models`)

4-1 PLAN
RecMind



(a) Tree-of-Thoughts (DFS) (b) Self-Inspiring

Figure 3: Comparison between Tree-of-Thoughts DFS and Self-Inspiring. Red arrows in the figure indicate the process for generating alternative thoughts at intermediate steps. Blue dashed arrows in the figure denote the backtracking process.

Algorithm 1: Self-Inspiring Planning

Require: Problem x, the current planning path $S = \{z^{(1)}, \ldots, z^{(m-1)}, s_{j_1}^{(m)}, s_{j_1+1}^{(m)}, \ldots, s_t^{(m)}\}$ at step t, LLM p_θ, and step limit T. Let `inspire`$(\cdot)$ be the API checking if the planning should explore an alternative reasoning branch.

while $t \leq T$ **do**
  Sample $s_{t+1}^{(m)} = (h_{t+1}^{(m)}, a_{t+1}^{(m)}, o_{t+1}^{(m)}) \sim p_\theta(\cdot|x, S)$
  if $h_{t+1}^{(m)}$ is "`End of Planning`" **then**
    break
  end if
  $S' \leftarrow S \cup \{s_{t+1}^{(m)}\}$
  if `inspire`$(\{x, S'\})$ **then**
    Sample $s_{t+2}^{(m+1)} \sim p_\theta(\cdot|x, S)$
    $S \leftarrow S' \cup \{s_{t+2}^{(m+1)}\}, m \leftarrow m+1, t \leftarrow t+2$
  else
    $S \leftarrow S', t \leftarrow t+1$
  end if
end while
return final response $y \sim p_\theta(\cdot|x, S)$

(Referred from `RecMind: Large Language Model Powered Agent For Recommendation`)

4-1 PLAN
AOT

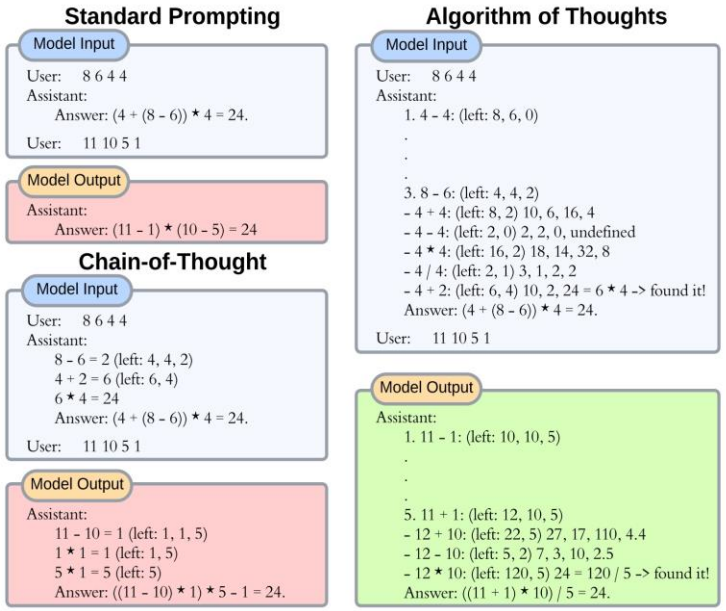


Figure 1: Comparison between standard prompting, CoT, and AoT in the game of 24. While standard prompting aims for a direct answer, CoT sketches out the successive steps to the final solution. AoT's in-context example, distinct from CoT, integrates the search process, highlighted by markers '1',..., '3' as "first operations" guiding subtree exploration for the problem set '8 6 4 4'. For clarity, only a single in-context example is displayed, with a focus on the third subtree exploration. AoT produces prospective search steps (e.g., the subtree exploration '5. 11 + 1') and evaluates potential subsequent steps to either progress towards a solution or retrace to another viable subtree.

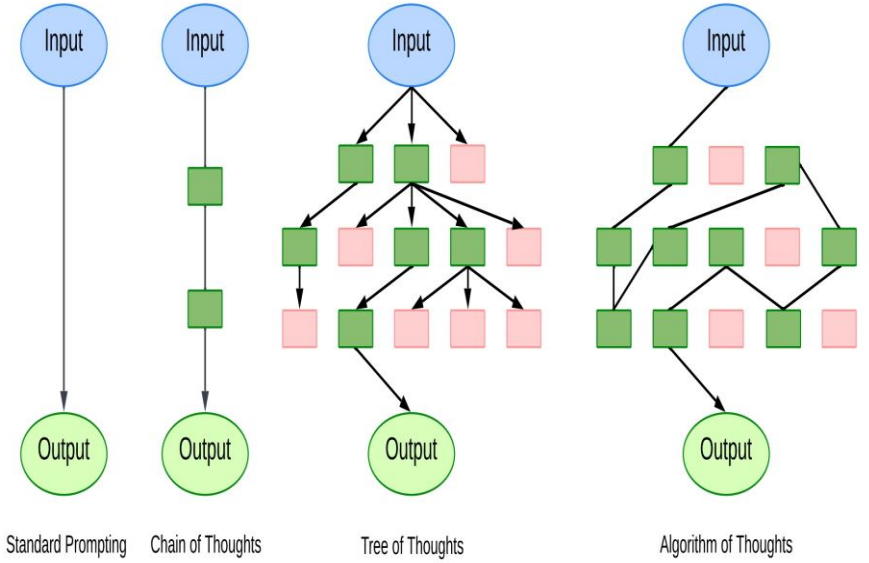


Figure 2: Illustration outlining various strategies for tackling reasoning problems with LLMs. Each box signifies a distinct thought, functioning as a unified string of words that forms an incremental pathway to reasoning. Green boxes indicate ideas deemed promising by the LLM, while red boxes represent less promising concepts.

(Referred from `Algorithm of Thoughts: Enhancing Exploration of Ideas in Large Language Models`)

4-1 PLAN
GOT

(Referred from `Graph of Thoughts: Solving Elaborate Problems with Large Language Models`)

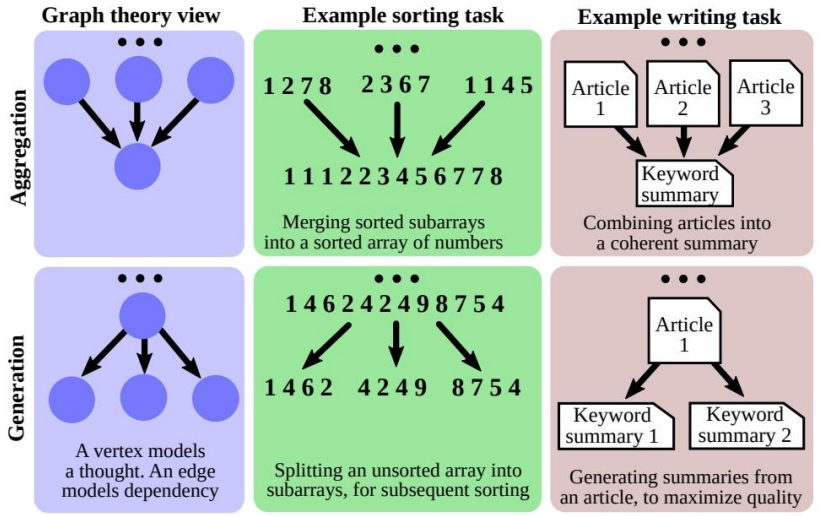


Figure 2: Examples of aggregation and generation thought transformations.

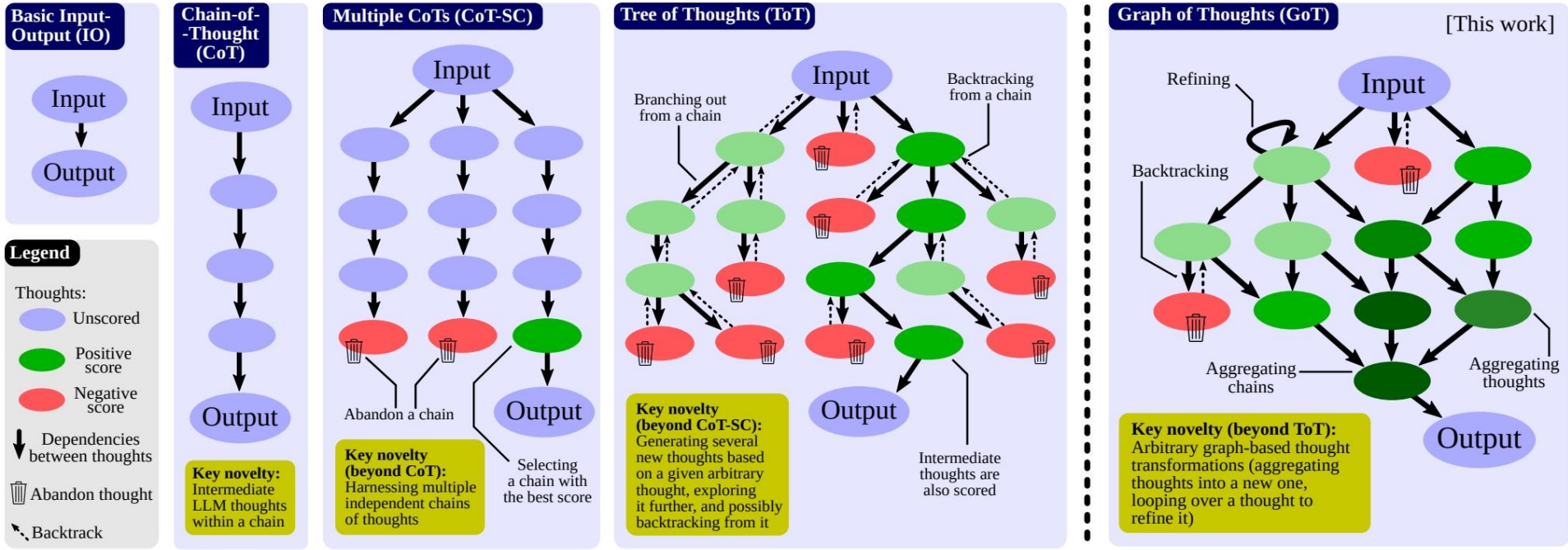
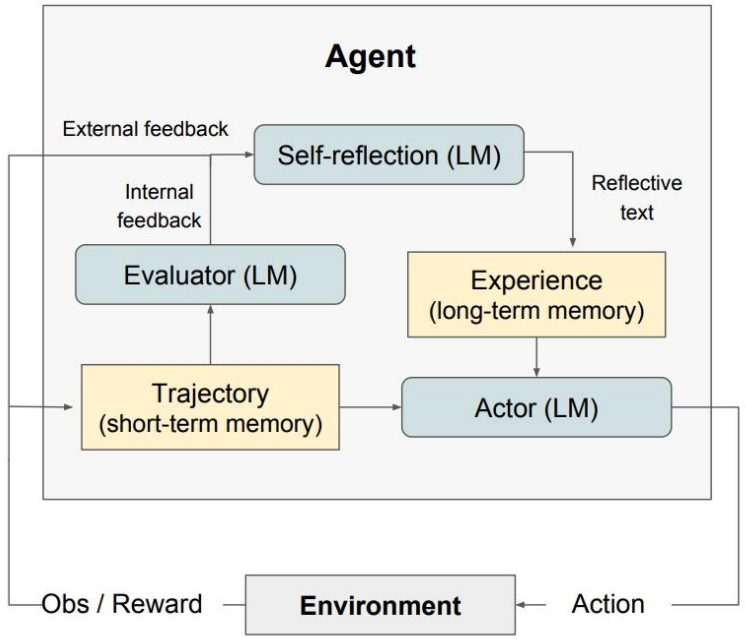


Figure 1: Comparison of Graph of Thoughts (GoT) to other prompting strategies.

4-2 Memory
SHORT OR LONG?



Algorithm 1 Reinforcement via self-reflection

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

Figure 2: (a) Diagram of Reflexion. (b) Reflexion reinforcement algorithm

(Referred from `Reflexion: Language Agents with Verbal Reinforcement Learning`)

4-2 Memory MemGPT

- MemGPT and its Memory Hierarchies

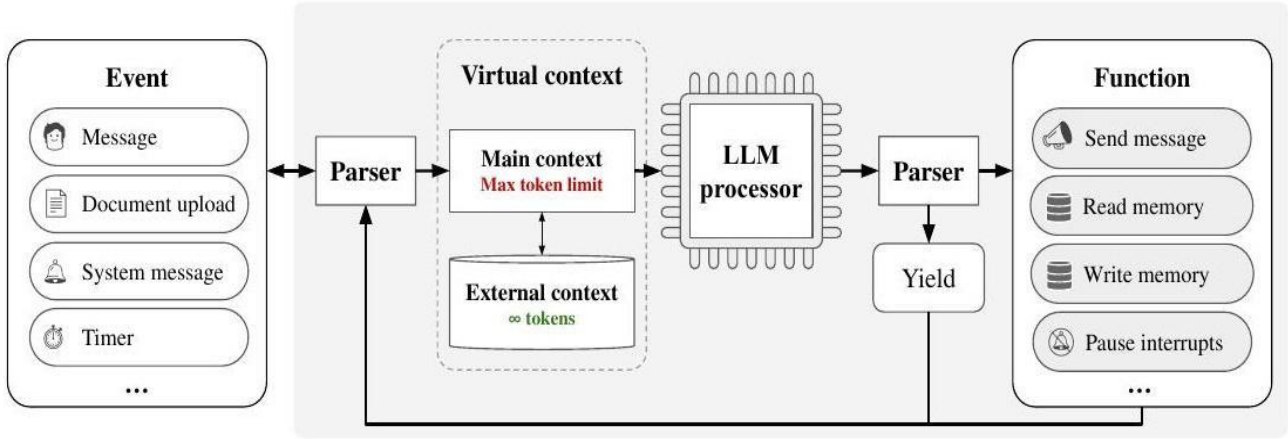


Figure 1: In MemGPT (components shaded), a fixed-context LLM is augmented with a hierarchical memory system and functions that let it manage its own memory. The LLM processor takes *main context* (analogous to OS main memory/RAM) as input, and outputs text interpreted by a parser, resulting either in a yield or a function call. MemGPT uses functions to move data between main context and *external context* (analogous to OS disk memory). When the processor generates a function call, it can request control ahead of time to chain together functions. When yielding, the processor is paused until the next external event (e.g., a user message or scheduled interrupt).

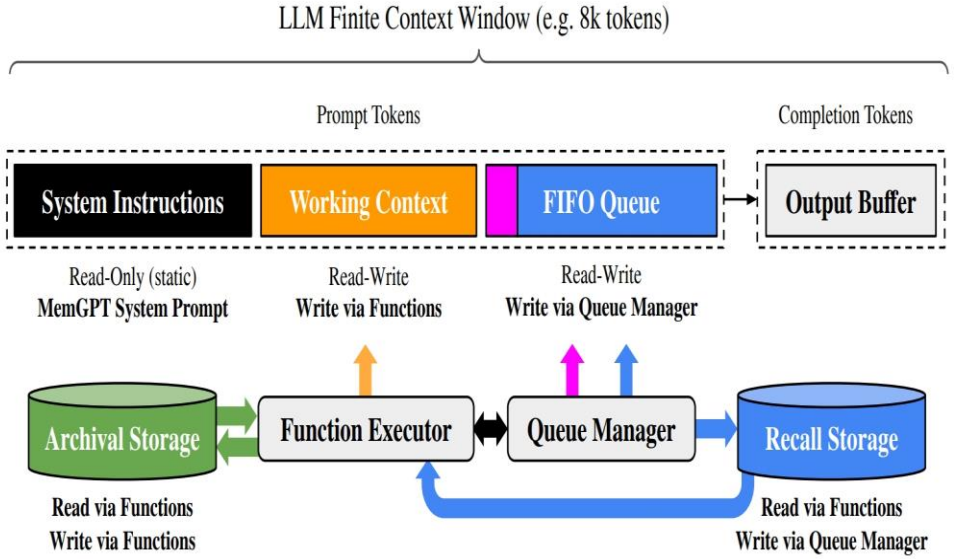


Figure 3. In MemGPT, a fixed-context LLM processor is augmented with a hierarchical memory system and functions that let it manage its own memory. The LLM's prompt tokens (inputs), or *main context*, consist of the system instructions, working context, and a FIFO queue. The LLM completion tokens (outputs) are interpreted as function calls by the function executor. MemGPT uses functions to move data between main context and *external context* (the archival and recall storage databases). The LLM can request immediate follow-up LLM inference to chain function calls together by generating a special keyword argument (`request_heartbeat=true`) in its output; function chaining is what allows MemGPT to perform multi-step retrieval to answer user queries.

(Referred from `MemGPT: Towards LLMs as Operating Systems`)

4-2 Memory
Aspects of Memory

- Memory write: reflexion,repeats,and rethink
- Memory Retrieval:recency,relevance,and importance
- Memory Overflow: first in first out
- Memory Depulicate: abstract or filter out

Resource of Memory:

Paper: Understanding ACT-R an Outsider's Perspective
Book: The Oxford Handbook of Cognitive Science

4-3 Action 2W1H

- When to act

  LLM-based agent should know when to call tools.

- What to act

  LLM-based agent should make it clear that which tools need to call.

- How to act

  LLM-based agent should how to decompose difficult tasks into sub-tasks and how to plan the execution sequence of sub-tasks.

(Referred from `Toolformer: Language Models Can Teach Themselves to Use Tools`)

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: x

Output:

Figure 3: An exemplary prompt $P(\mathbf{x})$ used to generate API calls for the question answering tool.

4-3 Action
Aspacts of Action

- Action Production

  Action via Memory Recollection.The action is generated by extracting information from the agent memory according to the current task. The task and the extracted memories are used as prompts to trigger the agent actions.

  (Paper:Generative Agents)

  Action via Plan Following.

- Action Space:The space of tool callable.

  APIs

  Databases & Knowledge Bases

  External Models:Other LLMs or Models

  Internal Knowledge

4-4 Environment

(1)Opterating System

(2)Simulation
- Driving
- 3D Reconstruction/3D Points
- Shopping
- Household
- Web browsing
- ...

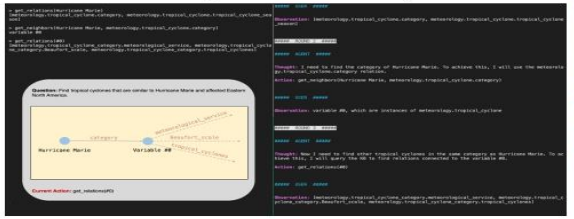
(3)DataBase

(4)Human FeedBack

(Referred from `AGENTBENCH: EVALUATING LLMS AS AGENTS`)

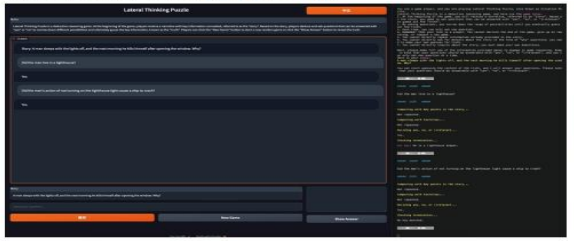
(a) **Operating System (OS)**
Task: "Find the total number of non-empty directories inside the '/etc' directory."
Action Space: Any valid bash commands
Observation: System standard output

(b) **Database (DB)**
Task: "What was the total number of medals won by United States?", given the table 'Olympic Medals'
Action space: Any valid SQL commands
Observation: MySQL CLI interface output



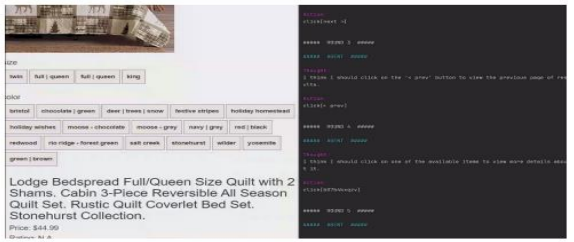
(c) **Knowledge Graph (KG)**
Task: "Find tropical cyclones that are similar to Hurricane Marie and affected Eastern North America."
Action space: Basic KG-querying tools
Observation: Query results

(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'

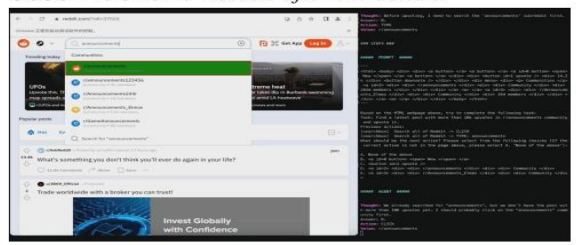


(e) **Lateral Thinking Puzzles (LTP)**
Task: "A man sleeps with the lights off, and the next morning he suicides after opening windows. Why?"
Action Space: Any binary questions
Observation: 'Yes', 'No', or 'Irrelevant'

(f) **House-holding (HH)**
Task: "Clean some soapbar and put it in coutertop"
Action space: A list of allowed actions in the room, or other accessible rooms
Observation: Results after the action.



(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



(h) **Web Browsing (WB)**
Task: "Find a latest post with more than 10k upvotes in r/announcements community and upvote it."
Action space: 1) Choose one out of all HTML elements in the webpage; 2) Click, Type, or Select Options
Observation: Page HTML (optional: screenshot)

5.Training or Tuning
Prompt VS Tuning

- Generative Tuning

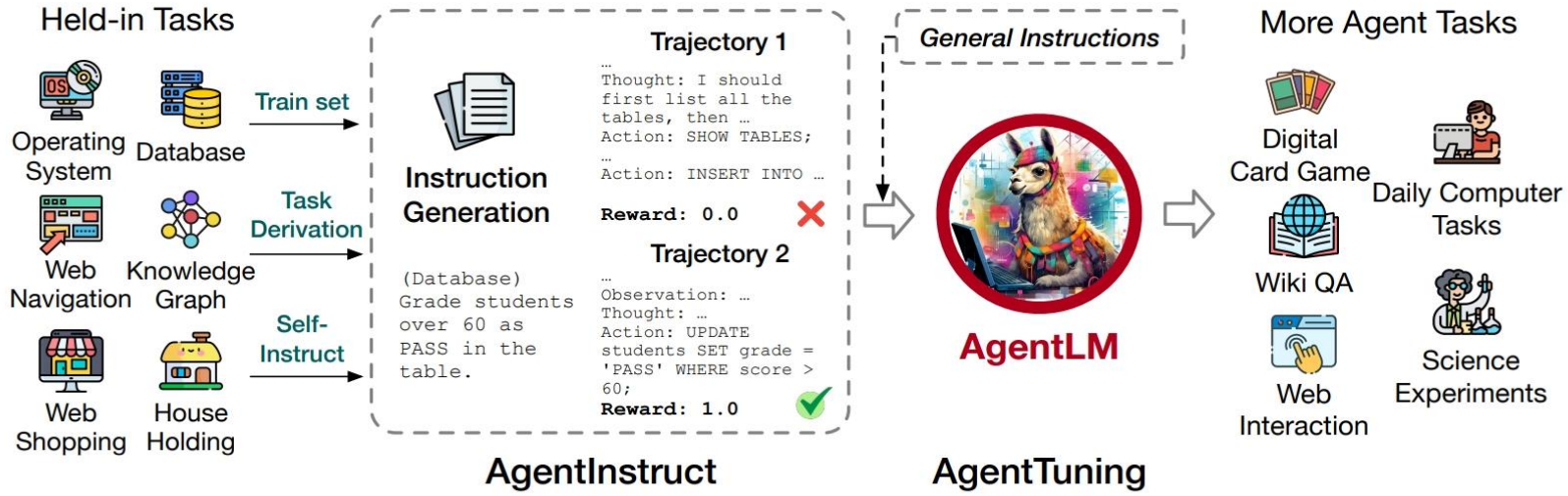


Figure 2: **An overview of AgentInstruct and AgentTuning.** The construction of AgentInstruct, consisting of instruction generation, trajectory interaction, and trajectory filter. AgentLM is fine-tuned using a mixture of AgentInstruct and general-domain instructions.

(Referred from `AGENTTUNING: ENABLING GENERALIZED AGENT ABILITIES FOR LLMS`)

5.Training or Tuning
Prompt VS Tuning

- Prompt for closed-source LLLM-based Agent(Verbal reinforcement learning)

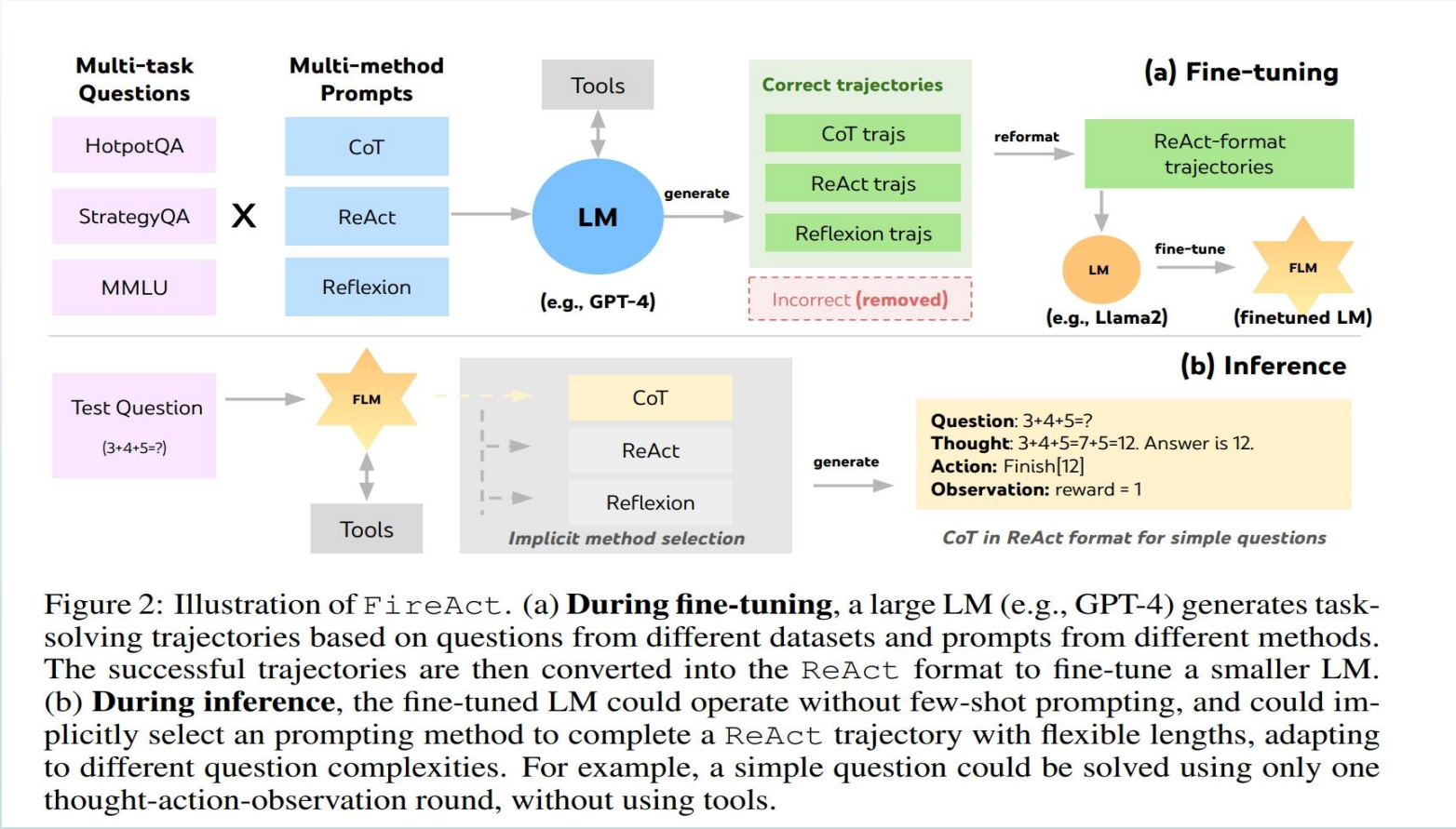
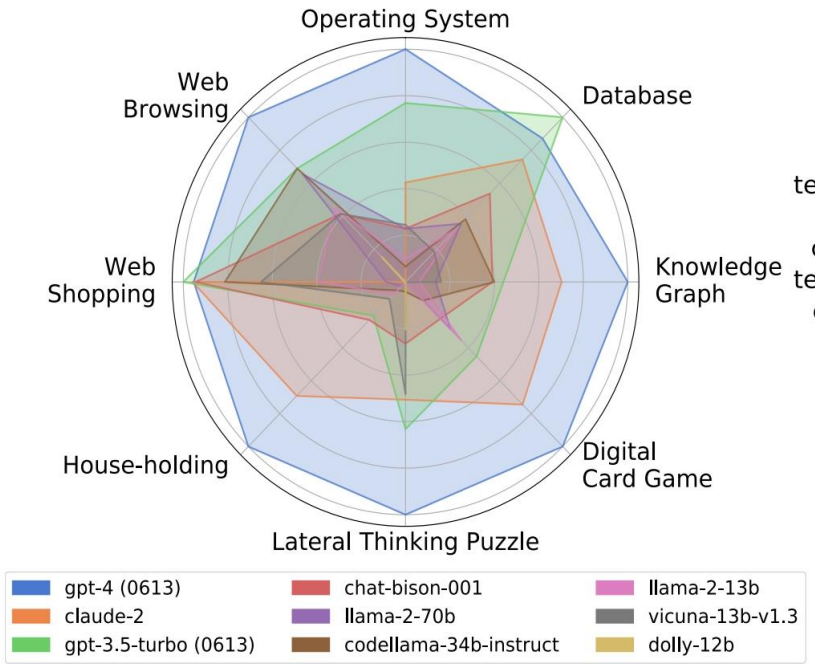


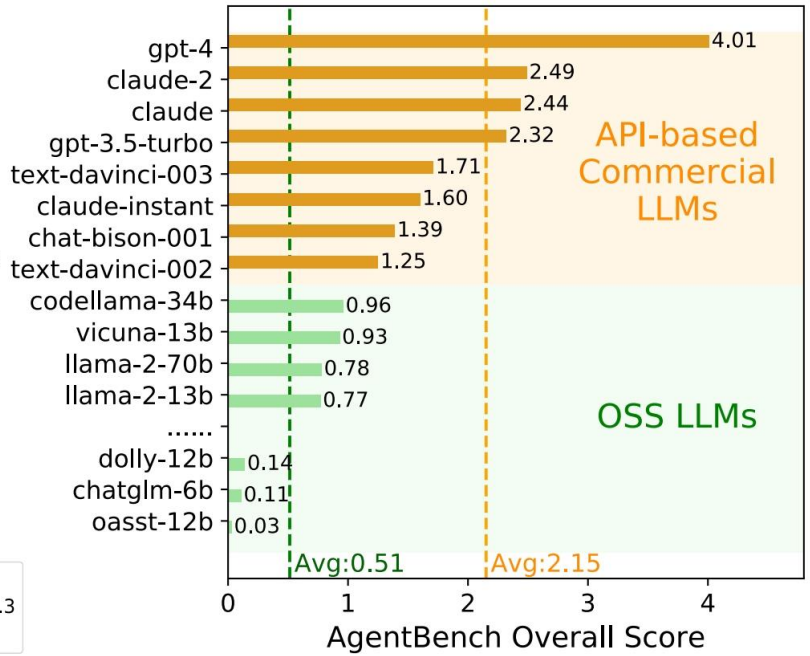
Figure 2: Illustration of FireAct. (a) **During fine-tuning**, a large LM (e.g., GPT-4) generates task-solving trajectories based on questions from different datasets and prompts from different methods. The successful trajectories are then converted into the ReAct format to fine-tune a smaller LM. (b) **During inference**, the fine-tuned LM could operate without few-shot prompting, and could implicitly select an prompting method to complete a ReAct trajectory with flexible lengths, adapting to different question complexities. For example, a simple question could be solved using only one thought-action-observation round, without using tools.

(Referred from `FIREACT:TOWARD LANGUAGE AGENT FINE-TUNING`)

6.Evaluation



(a) Typical LLMs' AgentBench performance (Relative) against the best in each environment



(b) Overall scores of AgentBench across 8 environ -ments. Dashed lines for two LLM types' average.

(Referred from `AGENTBENCH: EVALUATING LLMS AS AGENTS`)

6.Evaluation
Aspacts of Evaluation

Different Sources of Feedbacks

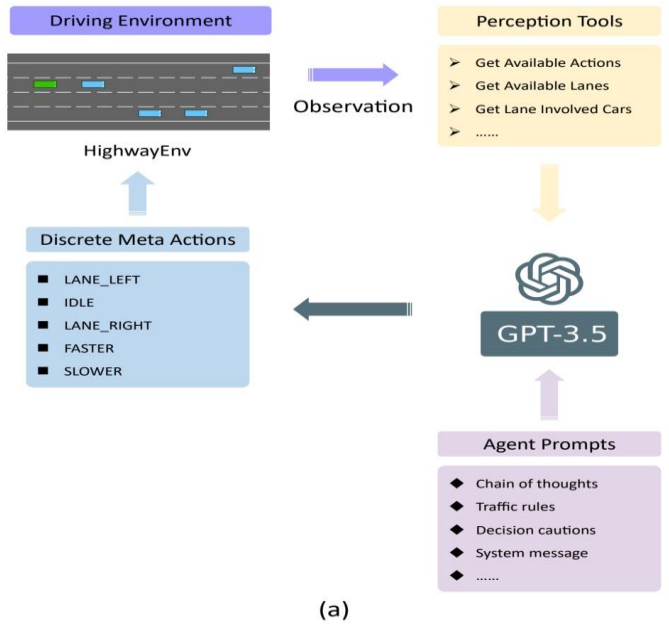
- From Expriment:Turing Test
- From Labeled Datasets:EvalDatasets/MMLU
- From Simulation:Such as paper `Agent bench`
- From human OR Chatgpt4/Claude3

Method:

- Reinforcement learning
- Student-Teacher Mode
- Multi-Agent vote
- Multi-Agent conversations

7.Prospect
Applications of Autonomous Agent

- 1.Drive Car



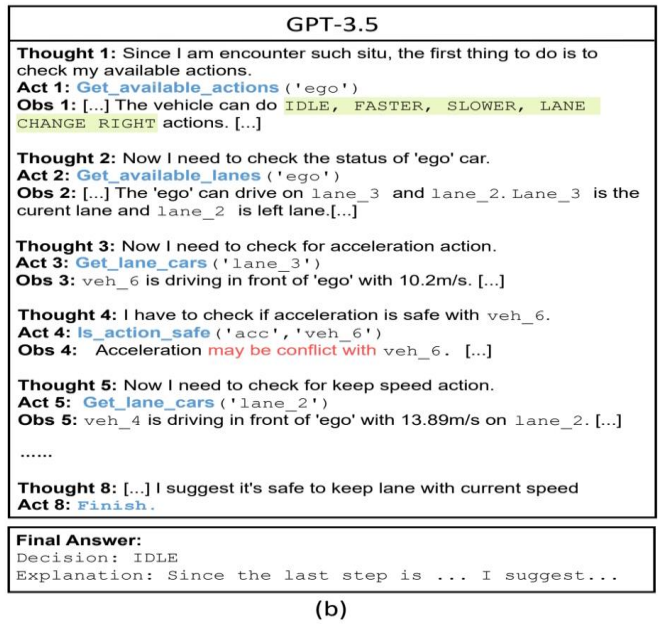
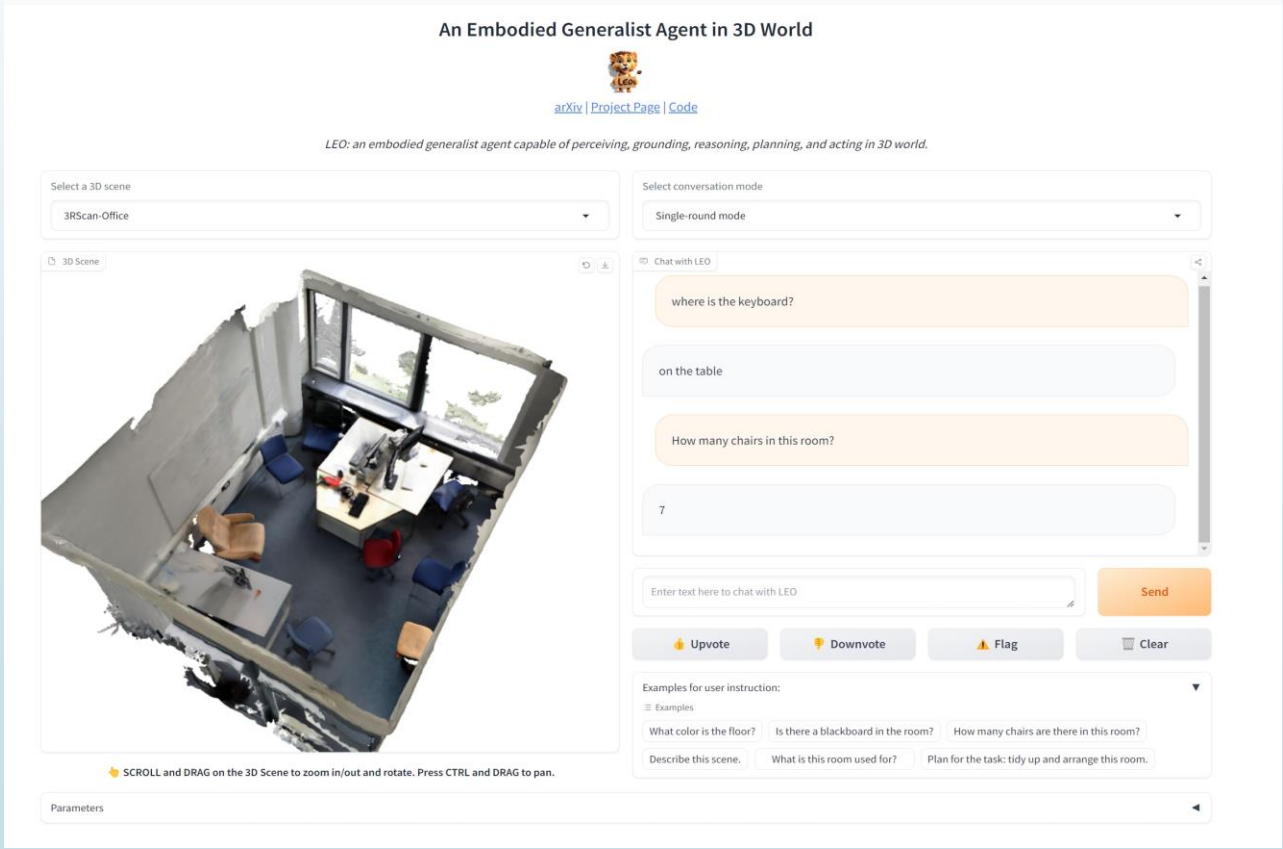


Figure 2: GPT-3.5 closed-loop driving in HighwayEnv: (a) GPT-3.5 observes its environment in HighwayEnv using perception tools and makes decisions to control vehicles, forming a closed loop. (b) GPT-3.5 employs the ReAct strategy to plan actions and use tools, while perceiving its surroundings through a cycle of thought, action, and observation

(Referred from `Drive Like a Human: Rethinking Autonomous Driving with Large Language Models`)

7.Prospect
Applications of Autonomous Agent

- 2.Embodied Game



(Referred from `AN EMBODIED GENERALIST AGENT IN 3D WORLD`)

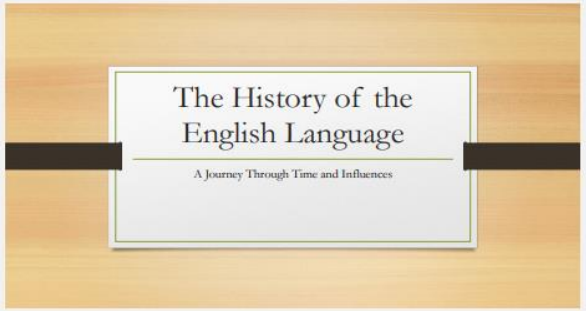
7.Prospect
Applications of Autonomous Agent

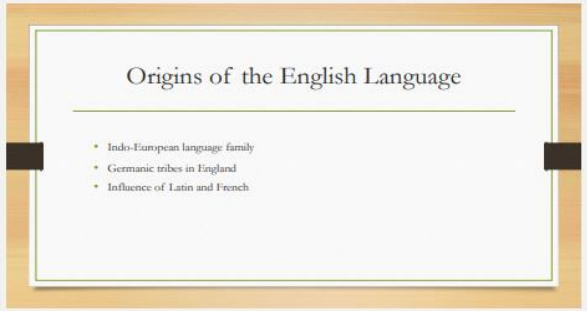
- 3.Create PPT

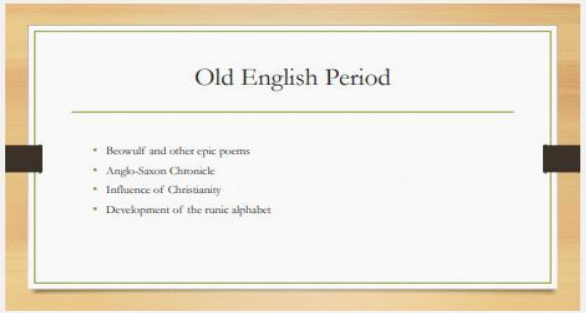
Action Input: {}
Observation: submitted file

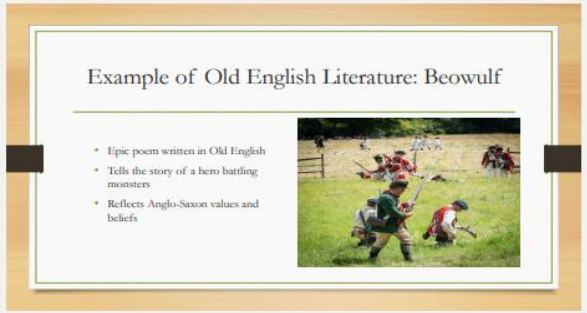
Query:
Design a 20-slide PowerPoint presentation on the history of the English language for an upper-level high school literature class. Focus on key developments and influences, such as Old English, Middle English, and Modern English; include important historical.

Trace:
Thought: We need to create a PPT presentation with a wooden theme.
Action: `create_file`
Action Input: `{"theme": "wooden"}`
Observation: created ppt file
Thought: We need to add the first page with a title and subtitle.
Action: `add_first_page`
Action Input:
`{"title": "The History of the English Language", "subtitle": "A Journey Through Time and Influences"}`
Observation: added first page
Thought: We need to add a text page outlining the origins of the English language.
Action: `add_text_page`
Action Input:
`{"title": "Origins of the English Language", "bullet_items": ["Indo-European language family", "Germanic tribes in England", "Influence of Latin and French"]}`
Observation: added text page
Thought: We need to add a text page outlining the Old English period.
Action: `add_text_page`
Action Input:
`{"title": "Old English Period", "bullet_items": ["Beowulf and other epic poems", "Anglo-Saxon Chronicle", "Influence of Christianity", "Development of the runic alphabet"]}`
Observation: added text page









(Referred from `Tool Learning with Foundation Models`)

7.Prospect
Applications of Autonomous Agent

- 4.Coding

Multiple agents develop software together through cooperation, communication, and feedback.

(Referred from `Communicative Agents for Software Development`)

THANKS FOR YOUR ATTENTION

- **Wish my sharing could serve as a catalyst for further discussion and inspiration!**

Thank you!