國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

MENSA: 利用心理模擬進行大型語言模型代理的動態經驗檢索

MENSA: Leveraging Mental Simulation for Dynamic Experience Retrieval in LLM Agents

張仲喆

Chung-Che Chang

指導教授: 許永真 博士

傅立成 博士

Advisor: Jane Yung-jen Hsu Ph.D.

Li-Chen Fu Ph.D.

中華民國 113 年 11 月

November, 2024

doi:10.6342/NTU202404517

Acknowledgements

回頭看看這段碩士生活，能順利寫完這篇論文並投稿至國際會議，真的要感謝很多在我身邊協助、支持我的人。

首先，最感謝我的指導教授許永真老師。無論是選題方向、實驗設計，您總是耐心地幫我一步步釐清、引導我思考，讓我在研究上成長許多。不只是論文，很多我迷惘或卡關的時候，老師總是很有耐心地幫我釐清方向，討論時您的分享也讓我獲益良多，真的很幸運能成為您的學生。也非常感謝郭彥伶老師，您總是在技術上幫我把關，每一次投稿前的修改，您都給了非常精確又實用的建議。您的專業和細心真的讓我學到很多，也讓這份研究更完整。

同時也很感謝共同指導教授傅立成老師、口試委員孫紹華老師和詹詩涵博士對於我的研究給予了很多深刻的意見，讓我對研究有了更多層次的理解，也幫助我釐清不少盲點，真的獲益良多。

也謝謝我的家人、朋友、實驗室的夥伴，謝謝你們的陪伴、傾聽和鼓勵，尤其在我壓力爆炸、自我懷疑的時候，是你們讓我能撐過去。謝謝所有幫助過我的人，沒有你們就不會有現在的我。

最後，也謝謝自己，這一路走來竭盡所能、不怕挑戰，完成了原本難以想像的任務。雖然還有很多需要學習的地方，但請繼續保持這樣的勇氣，同時學會取捨，未來也要好好加油。

doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

摘要

當前以大型語言模型（LLM）驅動的 Agent 展現了在互動文本環境中執行序列決策任務的潛力。然而，由於這些 LLM Agent 難以意識到執行某個動作所需的前提條件，因此在執行任務的關鍵步驟時經常失敗。與此不同，人類則通過心理模擬，藉由曾獲得的經驗想像動作及其後果的過程，來判別能滿足前置條件所需的動作。為了解決這一問題，我們提出了一種名為**MEN**tal **S**imulation **A**gent（MENSA）的方法。MENSA 利用 LLM 生成未來的「動作-觀察對」以達到進行預測的效果，並根據對未來的預測查找過去相關的經驗提供給 Agent，從而在不進行模型參數微調的情況下提升 LLM Agent 的性能表現。我們在互動文本環境 ScienceWorld 中評估了我們提出的方法，結果顯示，MENSA 不僅在使用較大的模型（如 GPT-4o-mini[1]）時，相比之前的最先進方法提高了 **15.8** 分（**29%**），並且在不同尺寸 LLM 中，包括較小的模型（如 Phi-3-mini[2]）在內，也一樣獲得性能的提升，進步幅度達到 **11.9** 分（**57.5%**）。

關鍵字：大語言模型、大語言模型代理人、心智模擬、提示工程、少樣本上下文學習

[1]https://platform.openai.com/docs/models/gpt-4o-mini
[2]https://ai.azure.com/explore/models/Phi-3-mini-4k-instruct

doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

Abstract

Large language models (LLMs) powered agents have shown the potential to perform sequential decision-making tasks in interactive text environments. However, these LLM agents often fail at executing the key steps to achieve tasks because they are not aware of the preconditions required for an action. Unlike these agents, humans, on the other hand, use mental simulation, a process of imagining actions and their consequences from experience, to identify the actions needed to satisfy the preconditions. To address this, we propose a **MEN**tal **S**imulation **A**gent (MENSA). MENSA leverages LLMs to generate a forecast of action-observation pairs for future time steps. Based on the forecast, it then retrieves relevant past experiences to improve the performance of the LLM agent without fine-tuning. We evaluate our method in the interactive text environment ScienceWorld to show that MENSA not only outperforms previous state-of-the-art by **+15.8** points (**29%**) when using a larger model (e.g., `GPT-4o-mini`[1]) but also consistently improves the performance across different sizes of LLMs including smaller ones such as `Phi-3-mini`[2]

[1]https://platform.openai.com/docs/models/gpt-4o-mini
[2]https://ai.azure.com/explore/models/Phi-3-mini-4k-instruct

doi:10.6342/NTU202404517

with improvement of **+11.9** points (**57.5%**).

Keywords: Large Language Model, LLM-based Agent, Mental Simulation, Prompt Engineering, Few-shot In-context Learning

doi:10.6342/NTU202404517

Contents

doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

List of Figures

doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

List of Tables

doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

Chapter 1

Introduction

In this chapter, we begin by introducing the background and motivation for the research. We will then provide a brief summary of the proposed methods and key results, followed by an outline of the structure of the entire study.

1.1 Background

In recent years, the rapid development of artificial intelligence, particularly in the field of Large Language Models, has revolutionized numerous areas of study. Top-tier architecture design and a vast magnitude and extremely high quality data makes the close-source and open-source LLMs flourishing, such as GPT series [1, 18], Gemini series [42–44], Llama series [9, 45], Mistral series [19, 20], and others. The State-of-the-art language models perform impressively in language processing domains, such as text generation [10, 27, 57], code generation [21], machine translation [22, 37]. With the increasing capabilities the LLMs have demonstrated, tasks related to further application scenarios also start to get attention.

To successfully carry out tasks in complex environments, autonomous agents must

doi:10.6342/NTU202404517

engage in a sequence of decisions to select optimal actions. This usually requires an agent to have the ability to decompose tasks [15], identify relevant prior experience, and generate, evaluate, and refine plans. Recent progress in large language models (LLMs) has demonstrated the potential to leverage LLMs and in-context learning [3, 8, 51, 53] in various components of autonomous agents [49]. These LLM-based agents [52] have been deployed in interactive domains such as programming [13, 14], games [48], and robotics [4, 17].

1.2 Motivation

With the valuable research mentioned above, state-of-the-art LLMs have demonstrated impressive results in simple benchmarks. However, they still struggle with more complex and diverse tasks, such as functioning as an AI agent. Recent research on LLM-based agent architectures has explored several aspects of sequential decision making to optimize agent policies. Despite these efforts, they still frequently fail by selecting invalid actions within the environment. Figure 1.1a-1.1c shows the example failure modes in these LLM agents when they execute a complex task: creating sugar water. ReAct [54] prompts the LLM to generate reasons before outputting actions but it does not consider whether the object or action exists or not. Reflexion [39] revises the plan based on a reflection on the previous actions but it does not know how to break down a complex task. The most recent work, Skill Set Optimization (SSO) [33], can select a relevant subgoal from a skill set constructed from prior action sequences but it does not consider if the precondition of the actions satisfies.

Human decision making, on the other hand, is not just about identifying the next ac-

doi:10.6342/NTU202404517

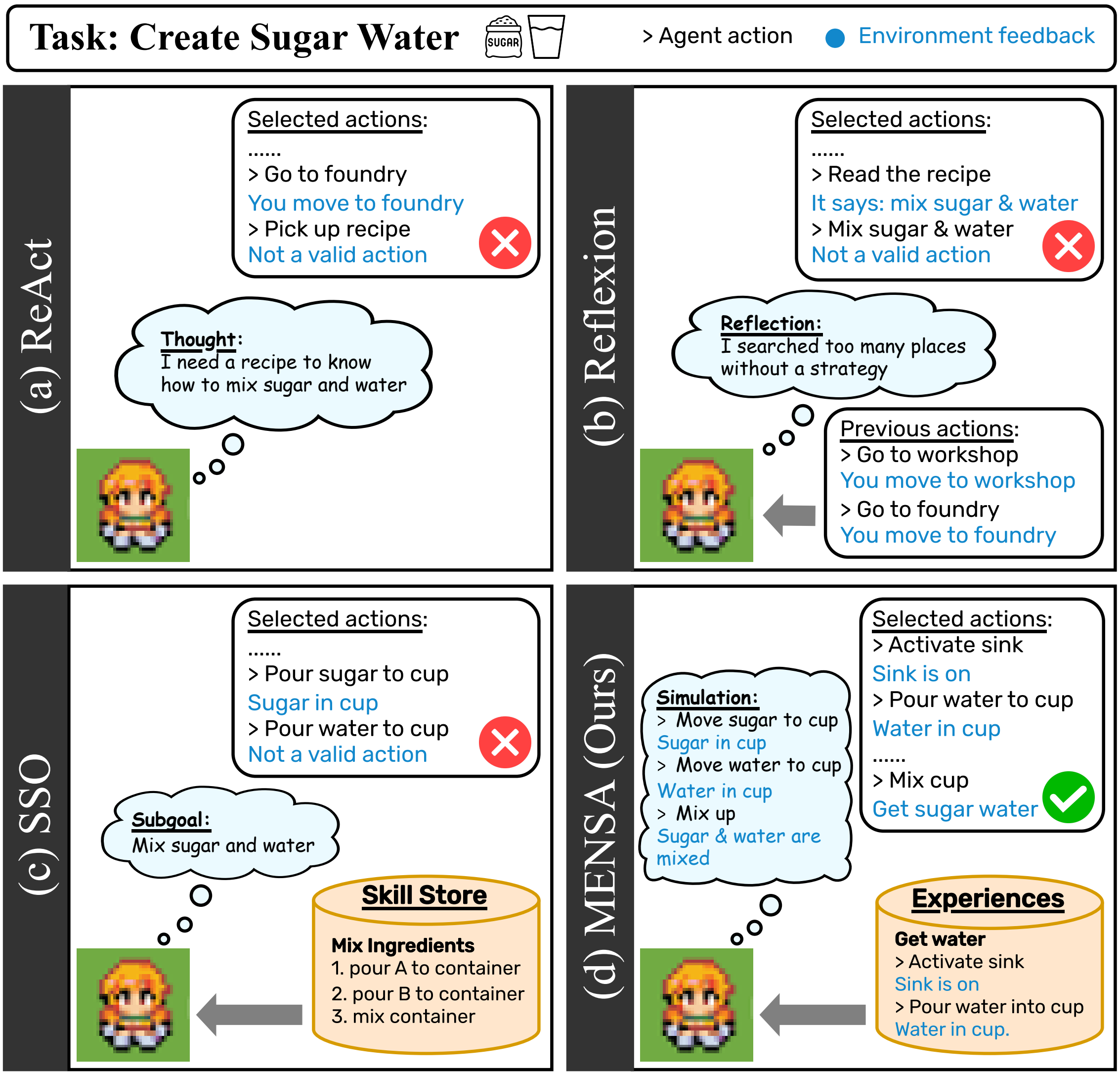


Figure 1.1: **Comparison of MENSA and other LLM agents for a task in chemistry experiments: creating sugar water.** (a) ReAct fails due to operational failure. (b) Reflexion addresses on past mistakes, but its ability to resolve problems remains limited. (c) Sometimes, SSO can retrieve useful skills, but without critical information, it still cannot complete the task. (d) MENSA leverages "mental simulation" to retrieve and utilize prior experiences more effectively.

doi:10.6342/NTU202404517

tion and subgoal. Before executing the actions in an environment, humans also imagine the needed steps, their outcomes, and how to synthesize them together for the successful execution of tasks. This mental simulation process enables us to hypothesize the preconditions and effects of actions so we can select the actions that help satisfy the preconditions of subgoals based on our prior experience. The ability to simulate future steps improves our efficiency when interacting with the world by avoiding constantly revising our plans.

In this research, we focus on leveraging the concept of mental simulation to design a novel autonomous agent framework without requiring model fine-tuning. The goal is to develop a capability-friendly approach adaptable to various LLMs, particularly smaller or less powerful models.

1.3 Proposed Method

In this study, we present the Mental Simulation Agent (MENSA), an LLM agent that uses the mental simulation process for policy improvement. At each time step, in addition to selecting the current action, MENSA prompts LLMs to generate action-state pairs (i.e., a forecast) for future time steps. The simulated actions are used to retrieve the relevant experience for the task. For example, in Figure 1.1, only the forecast in MENSA suggests that “move water to cup” is needed before mixing sugar and water so it can select the action “Activate sink” that helps satisfy the precondition of the task. An important aspect of effective decision making is to learn from previous experiences. MENSA employs a multi-episodic refinement framework to improve its forecasts. In the beginning, the agent uses the knowledge in pretrained LLMs to generate forecasts. After each episode of task execution, the LLM will incorporate the environmental feedback to update an experience

doi:10.6342/NTU202404517

set which is used to refine the agent's forecast to be more precise and task-specific.

1.4 Result and Contribution

We evaluate the effectiveness of MENSA in ScienceWorld [50], an interactive text-based environment where the agent is tasked to complete science experiments. We show that MENSA outperforms the top-performing SSO agent by **+15.8** points with a large model (`GPT-4o-mini`). Moreover, for agents that use smaller LLMs MENSA can improve the performance by a large margin (e.g., **+11.9** points for `Phi-3-mini`) demonstrating that mental simulation can consistently improve an agent's planning capability across different LLMs.

In summary, this study makes the following contributions:

- We propose MENSA, a novel agent architecture that incorporates mental simulation with multi-episodic policy improvements for sequential decision making tasks.
- We implement MENSA to perform complex decision making tasks (e.g., science experiments) in the ScienceWorld benchmark.
- Our results demonstrate that MENSA significantly improves the performance of LLM agents across different sizes of LLMs, reaching state-of-the-art performance in ScienceWorld.

1.5 Thesis Organization

The first chapter introduces the relevant research topics, provides the background, explains the study motivation, presents the proposed method, results and contribution. The

doi:10.6342/NTU202404517

structure of the thesis is also briefly provided. Chapter 2 presents a comprehensive review of the related work including topics such as sequential decision making with LLMs, mental simulation and capability evaluation in LLMs are included. In Chapter 3, we provide a detailed discussion and definition of the primary research topics, including the current challenges, unresolved issues, and the potential value these topics may offer. Chapter 4 describes the research methodology we propose, detailing the overall architecture, the main prompting design, and each of each module in MENSA. Chapter 5 presents the experimental details, benchmark environment setup, results and ablation studies. The results are compared with the existing methods. Extended topics are discussed in Chapter 6. In the final, Chapter 7 summarizes the main contribution of the thesis and provides a conclusion. Supplementary material, as well as more detail of experiment and results are provided in appendix.

doi:10.6342/NTU202404517

Chapter 2

Related Work

In this chapter, we delve into the key concepts involved in the problem studied in this study. In Section 2.1, we will first begin by reviewing existing methods for tackling sequential decision-making task, particularly when we employ LLMs. The discussion will focus on whether the actor, represented by LLMs, has been fine-tuned prior to evaluation. Next, in Section 2.2, we introduce the concept of mental simulation, which form the core of our methodology, examining it from both its original context and related perspective . Finally, in Section 2.3, we discuss the evaluation of LLM capabilities motivated by the notable differences we observed in how various solutions impacted models with differing capacities during our experiments.

2.1 Sequential Decision Making with LLMs

Since LLMs have excelled in natural language processing and commonsense-related tasks, numerous studies on more challenging extended tasks have followed. LLMs have started to be adopted as a crucial component for tackling complex decision-making task across various applications [15, 25]. Based on this concept, we will explore several com-

doi:10.6342/NTU202404517

mon implementation approaches.

Fine-tuning approaches

Previous work, *SwiftSage* [28], has proposed a framework consisting of two modules: the SWIFT module, responsible for quick intuitive responses, and the SAGE module, which handles slower, more deliberate reasoning. These models have shown that action planning can be achieved using domain-specific training data. However, fine-tuning LLMs while the agent interacts with the environment is often impractical or even impossible due to the large size or proprietary nature of some LLMs. The balance between training costs, uncertainty, and overall effectiveness presents significant challenges in both research and implementation.

In-context Learning approaches

To get a better use of the pretrained LLM, a technique called in-context learning is widely used where a LLMs is given with labeled demonstration or instruction. With the useful guidance integrated into the input prompt, the LLM can performs better in reasoning or even adapt to a varied task. This study focuses on refining LLM agent policies without gradient updates [16, 17, 39, 54]. Most of these prior works focus on employing the frozen LM as a decision maker with full prior knowledge.

Incorporating Long-term Memory Learning

Recent approaches like *CLIN* [32] and *ExpeL* [56] maintain and adapt a long-term memory pool that is useful across trials, tasks, and environments. In addition to external

doi:10.6342/NTU202404517

memory, *SSO* [33] and *Voyager* [48], identify subgoals and skills to achieve these subgoals to enhance the decision-making process. However, these existing methods do not anticipate future actions and their outcomes, which limits their capacity to make more informed decisions.

2.2 Mental Simulation

Mental simulation [41], a cognitive process that encompasses both the imaginative construction of hypothetical scenarios and the reflective reconstruction of real-life situations, allows humans to mentally explore and analyze different possibilities. It typically includes two categories: process simulation and outcome simulation. Process simulation helps humans envision the steps needed to reach a goal, while outcome simulation focuses on visualizing the results of achieving that goal. In search and AI planning, mental simulation has been implemented with several look-ahead approaches [47] or Monte Carlo simulations [5]. However, these approaches consider access to a world model which is not yet available to LLM agents. MENSA draws inspiration from these ideas but implements mental simulation via in-context learning to enhance the performance of LLM agents.

2.3 Capability Evaluation in LLMs

LLMs have been shown to have a diverse set of capabilities that enable them to tackle complex tasks across a wide range of domains. In the context of sequential decision-making, instruction following and planning are particularly critical. Instruction following ensures that the model accurately interprets and executes commands. Planning allows the model to anticipate future steps, evaluate potential outcomes, and make deci-

doi:10.6342/NTU202404517

sions that optimize the process. However, not all LLMs' capabilities are equal. Recent benchmarks have evaluated LLMs' single capability such as understanding [12], reasoning [7, 40, 46], instruction following [35, 58], and comprehensive ability by accomplish complex tasks [30, 34]. But the goal-oriented benchmark, focusing solely on evaluating the outcome, which is not quite enough for evaluating the influence of each capabilities. To better compare the differences in method effectiveness, In this study, we use *T-Eval* [6] to evaluate capabilities of the LLM in instruction following and planning to show how MENSA improves LLMs with different capabilities.

doi:10.6342/NTU202404517

Chapter 3

Problem Definition

In this study, we explore a sequential decision-making problem using an LLM-based agent, where the core challenge is to find the optimal action strategy. We attempt to define the problem framework using Markov decision process to help clarify the task objectives, as discussed in Section 3.1.

To clearly define this problem, we introduce a unified set of notations to describe the key elements of the process, including states, actions, and rewards, as listed in Section 3.2.

3.1 Sequential Decision Making

We consider the interactive decision-making problem as a Markov decision process (MDP), denoted by the tuple $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma >$, which satisfies the Markov property:

$$P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_0, a_0, ..., s_t, a_t)$$

indicating that the future state depends only on the current state and action, independent of the past sequence. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state of the environment and action spaces of

doi:10.6342/NTU202404517

the agent, respectively. At each time step t, the decision-maker (denoted as DM) observes the current state $s_t \in \mathcal{S}$ and makes a decisions $a_t \in \mathcal{A}$. After executing action a_t, the environment transitions to the next state s_{t+1} according to the transition probability

$$P = P(s_{t+1}|s_t, a_t)$$

. The agent also receives an observation $\{o_t\}_{t=0}^{T} \in \mathcal{O}$, and a corresponding reward $r_t = r(s_t, a_t)$ where $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function. Over an infinite horizon, the agent accumulates the discounted reward:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

where $\gamma \in (0, 1]$ is the discount factor. The objective of the decision-maker is to maximize the expected reward, expressed as:

$$V_\pi(s_t) = E_\pi[G_t|s_t]$$

by following an optimal policy $\pi^* = \arg\max_\pi V_\pi$ that maximizes the total reward and eventually reaches the target state s^*.

Goal:

The objective is that a decision-maker makes a sequence of decisions $\{a_t\}_{t=1}^{T}$ such that it reach the goal state (s^*). The decision-maker have to optimize its actions (a_t) in each time step (t) to maximize cumulative rewards and complete the task (T) within a limited number of attempts.

Input:

doi:10.6342/NTU202404517

A task (T) that need to be completed, along with a detailed description (TD) and the current state (s_t) of the environment.

Output:

The action (a_t) which the decision-maker chooses to take.

3.2 Notation

- DM: Decision-maker
- T: Main task
- TD: Description of the main task T
- s^*: Goal state of the main task T
- $\mathcal{A}$: Finite set of actions
- $\mathcal{S}$: Finite set of states
- $\mathcal{O}$: Finite set of observation
- $P(s_{t+1}|s_t, a_t)$: Transition probability
- $r(s_t, a_t)$: Environment reward function
- $V_\pi(s)$: State-value function from state s following policy π
- G_t: Total discounted reward from time-step t.
- γ: Discount factor, $\gamma \in (0, 1]$

doi:10.6342/NTU202404517

- $\pi(a_t|s_t)$: Decision policy of the decision-maker
- $\{s_t\}_{t=1}^{T}$: Sequence of environment states from time 1 to T
- $\{a_t\}_{t=1}^{T}$: Sequence of actions decided by the decision-maker from time 1 to T
- $\{o_t\}_{t=1}^{T}$: Sequence of observations, provided by the environment as each action a_t is executed from time 1 to T, reflects the change from S_t to S_{t+1} caused by the execution of a_t.
- $\{r_t\}_{t=1}^{T}$: Sequence of rewards from the environment after each action a is executed, from time 1 to T

doi:10.6342/NTU202404517

Chapter 4

Methodology

MENSA is composed of three key components: Mental Simulator, Executor, and Experience Learner. The **Mental Simulator** carries out mental simulations for a given task T, enabling the agent to plan actions and anticipate outcomes. In each step, the Experience Retriever dynamically extracts relevant experiences from the experience set, aligning them with the current task and time step. The **Executor** translates the actions generated by the LLM into admissible actions, ensuring that action execution failure is not caused by variations in semantic interpretation. The **Experience Learner** summarizes the executed trajectories into experiences at the end of each episode, and uses them to refine the decision-making capabilities of the agent. We detail each component in the following sections: the Mental Simulator in Section 4.1, the Executor in Section 4.2, the Experience Learner in Section 4.3. The diagram illustrates the overall architecture of our methodology is shown in Figure 4.1, and its comparison with other methods is presented in Figures 4.2.

doi:10.6342/NTU202404517

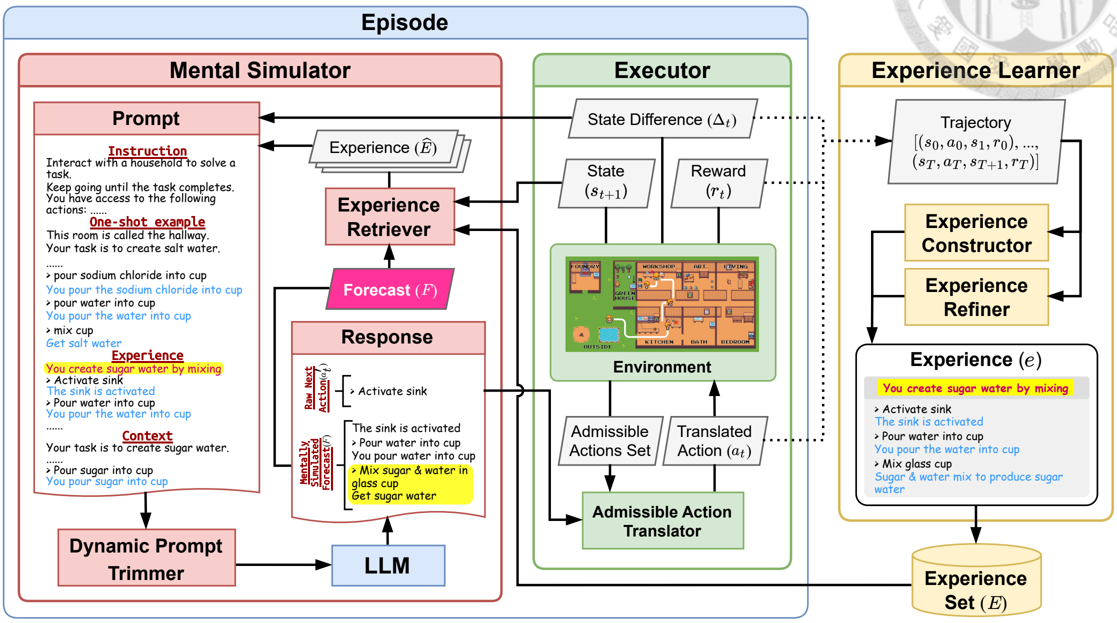


Figure 4.1: **MENSA architecture.** MENSA is allowed to learn experience from multiple episodes. In each episode, MENSA collects experience from its trajectory.

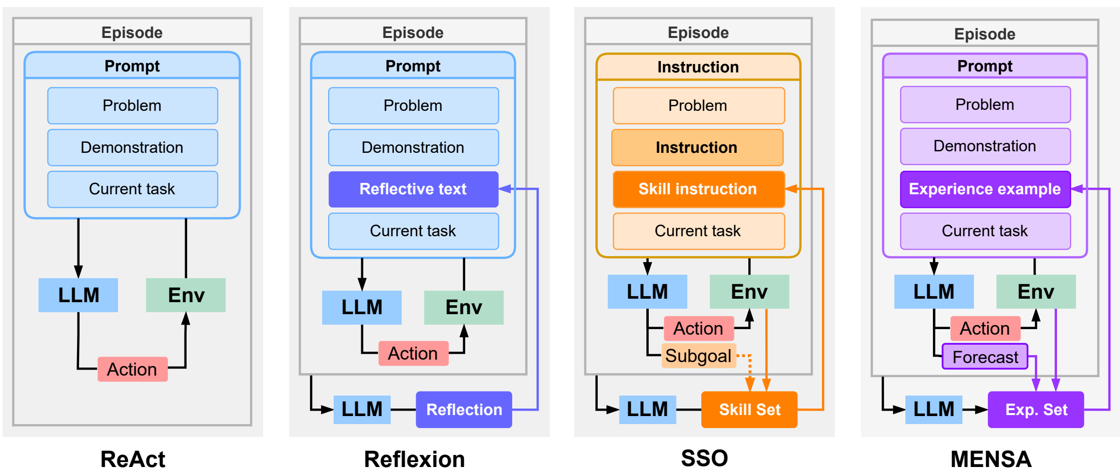


Figure 4.2: **Architecture comparison of MENSA and baselines.** Comparison diagram of high level architectures for four methods: ReAct, Reflexion, SSO, MENSA.

doi:10.6342/NTU202404517

4.1 Mental Simulator

During the decision-making phase using mental simulation, we aim to guide the LLM to plan a solution that leads to the target, decide the next action and foresee what happen in the upcoming steps can leads to success. Next, we execute action in the environment and use forecasting to extract useful trajectory from past experiences for the next decision-making process. Considering that different models have varying specifications, we specifically optimized the prompt trimmer to prioritize the retention of important information. The topics mentioned above are provided in Sections 4.1.1, 4.1.2, and 4.1.3, respectively. A demonstration of decision-making process is provided in Appendix A.1.

4.1.1 Prompt

Large Language Models with advanced capabilities can effectively handle more unconstrained and complex inputs. However, when working with smaller LLMs, which have limited capacity and less powerful abilities, it is often more effective to use simpler structures and language [2, 38]. This approach allows smaller models to perform better by reducing the cognitive load and ensuring that they can process and generate accurate responses within their constraints. Consequently, we adopt the prompt structure used in ReAct, where an action is prefixed with a triangle bracket symbol '>', and the subsequent line without the symbol denotes the observation. This format not only enhances the clarity of the input for smaller models but is also fully compatible with larger LLMs, ensuring consistent and effective processing across different model sizes.

As shown in Listing 4.1, our prompt consists of a one-shot example to facilitate the

doi:10.6342/NTU202404517

model to perform in-context learning and grasp the structure, along with selected relevant experience set $\hat{E}$, a task description TD, and the current trajectory

$$H = (a_0, o_0, a_1, o_1, ..., a_{t-1}, o_{t-1})$$

. We provide the instruction "`Keep going until the task completes.`" in the prompt to engage the LLM to perform mental simulation by forecasting the next action-observation pairs

$$F = (a'_{t+1}, o'_{t+1}, ..., a'_{T'}, o'_{T'})$$

, in addition to generating the next action a'_t. In scenarios where prior experiences are absent (i.e., episode-0), the LLM is encouraged to simulate outcomes based on commonsense reasoning. However, as experiences accumulate (i.e., episode-1 and beyond), the LLM is guided to utilize this information to make more informed and task-specific decisions. Additionally, target-based experiences are prioritized due to their greater relevance to the execution objective, and the internal order of both target-based and state-based experiences is sorted in ascending order due to recency bias. These aspects are further discussed in Section 5.4.3. The response of the LLM is then parsed into raw action and forecast, where the raw action is forwarded to the admissible action translator, while the forecast is sent to the experience retriever for next iteration.

Listing 4.1: **Prompt template for Actor in MENSA**

```
Interact with a household to solve a task. Keep going until the task
    completes.

You have access to the following actions: look around, show task,  show
     inventory, read OBJ, activate OBJ, deactivate OBJ, open OBJ, close
     OBJ, pick up OBJ, look in OBJ, look at OBJ, focus on OBJ, flush
    OBJ, eat OBJ, dunk OBJ in OBJ, move OBJ to CONTAINER, mix CONTAINER
    , teleport LOCATION, go LOCATION, wait.
```


doi:10.6342/NTU202404517

```
Here is an example:
<Environment Description>
<Task Description>
<Trajectory>

The following examples contain potentially useful information:
<State-base Experience Examples>
<Target-base Experience Examples>

Now it is your turn:
<Current Environment Description>
<Current Task Description>
<Current Trajectory>
```

4.1.2 Experience Retriever

MENSA learns experience from the trajectory in the past episode. We store the experience in the experience set E. Each of the experience contains useful subtrajectories (action-observation pairs) and subgoal description, and we additionally save the first partial state in the subtrajectories for use in retrieval. Experience construction mechanism is described in Section 4.3.

To enhance the quality of retrieved experience, as shown in Figure 4.3, we compare the forecast with the initial partial states and subgoals for each experience in the experience set. For initial partial state (partial-state-based) comparison, we use a mechanism similar to that in SSO, where the cosine similarity of text embeddings is calculated to identify the top-k experiences that most closely resemble the current partial state. This ensures that the retrieved experiences are well-suited to continue from the present partial state. For subgoal (target-based) comparison, which aims to provide experiences that have similar goals to the current goal, we use a generative text classifier [55]. The classifier assesses whether the current forecast, F, which is set to be 3 steps long and discussed further in

doi:10.6342/NTU202404517

Section 6.1, aligns with the experience goal. A similarity score is then derived from the classifier's probability logits. We then sort these scores and select the top-k experience instances to guide the decision-making process.

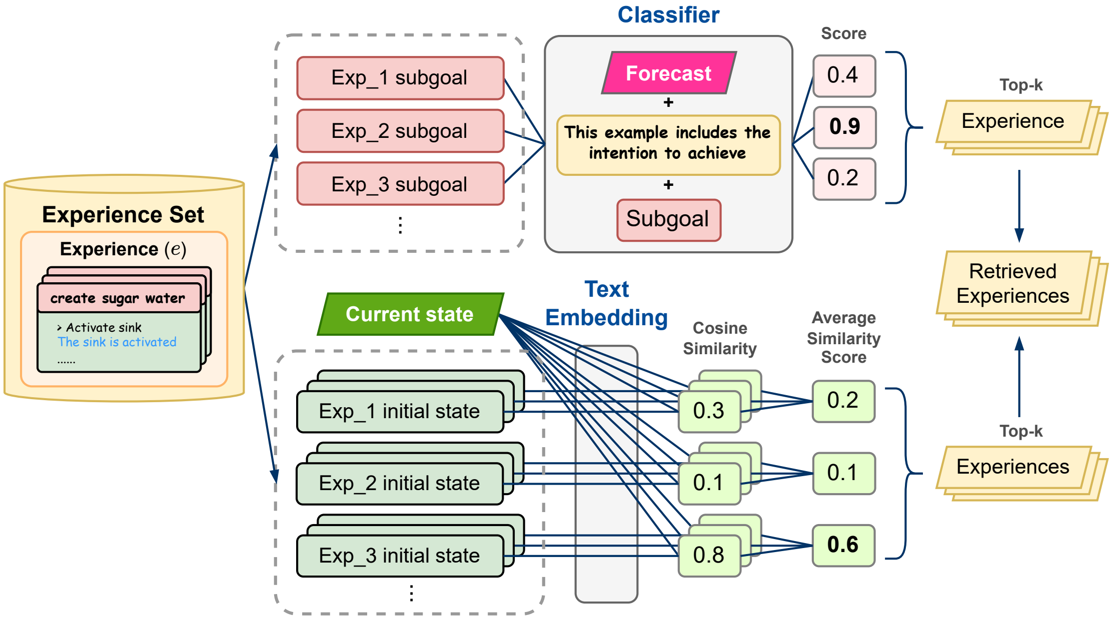


Figure 4.3: **Illustration of Experience Retriever.** The experiences selected in the final stage are combined after being separately sorted using target-based comparison and state-based comparison.

4.1.3 Dynamic Prompt Trimmer

Current LLMs encounter significant challenges when dealing with long in-context learning, especially with extended input sequences [26, 31]. Although more advanced models show promising results with inputs up to 20K tokens, their ability to effectively process and understand longer sequences declines substantially. Models with fewer than 10 billion parameters face even greater difficulties, with performance plateauing around 8K tokens and dropping sharply beyond 10K tokens. Therefore, following [26], we focus on optimizing the model's performance for sequences of 8K tokens or less, ensuring that the LLM operates at peak efficiency before any degradation sets in.

doi:10.6342/NTU202404517

When working within an 8K token limit, it is possible for length of the prompt comprising one-shot example, selected experience set, and current trajectory, to surpass this threshold. Existing work typically address this by naively trimming the trajectory within the one-shot example [28], or by taking only 3 latest steps from the current trajectory [39]. However, this poses a significant drawback where the important historical trajectory information is potentially misaligned with the ongoing context, thus reducing its overall utility. Therefore, we introduce a dynamic prompt trimmer that strategically remove portions of the text starting with the least relevant segments.

In the event that the prompt length exceeds 8K tokens, we first consider trimming the tail end of the one-shot example trajectory $\hat{H}$. Our priority is to remove parts that are less relevant to the current time step, such as distant future steps. We establish minimum thresholds, τ^H for the current trajectory and $\tau^{\hat{H}}$ for the example trajectory, to ensure the LLM retains sufficient context for accurate decision making. Once the current trajectory exceeds τ^H steps, we begin trimming the head of the current trajectory, simultaneously sliding the effective window of $\hat{H}$ to maintain alignment between the current and example trajectories. If neither the current nor the example trajectory can be further trimmed, we then reduce the experience set $\hat{E}$. Thresholds are only adjusted downward once all possible trimming has been exhausted, and the experience set has been fully pruned. The detailed algorithm is outlined in Algorithm 1.

4.2 Executor

Executor is responsible for translating and executing the intended action derived from the LLM's response, ensuring that these actions are both admissible and comply to the for-

doi:10.6342/NTU202404517

Algorithm 1: Dynamic Prompt Trimming

Require: A prompt p (tokenized into p_{tok}) containing one-shot example's trajectory $\hat{H}$, selected experience set $\hat{E}$, current trajectory H, token length threshold τ_{tok}. $\triangleright$ $i^{C}_{\text{start/end}}$ indicates the start or end index of the component C.

Ensure: A trimmed prompt with $\textsc{len}(p_{\text{tok}}) <= \tau_{\text{tok}}$

while $\textsc{len}(p_{\text{tok}}) > \tau_{\text{tok}}$ **do**
    if $\textsc{len}(H) > \tau^{H}$ **then**
        $i^{H}_{\text{start}} \leftarrow i^{H}_{\text{start}} + 1$ $\triangleright$ *Trim current trajectory*
        if $i^{\hat{H}}_{\text{end}} < i^{\hat{H}}_{\text{original\_end}}$ **then**
            $i^{\hat{H}}_{\text{start}} \leftarrow i^{\hat{H}}_{\text{start}} + 1$ $\triangleright$ *Window sliding*
            $i^{\hat{H}}_{\text{end}} \leftarrow i^{\hat{H}}_{\text{end}} + 1$
        end if
    else if $\textsc{len}(\hat{H}) > \tau^{\hat{H}}$ **then**
        $i^{\hat{H}}_{\text{end}} \leftarrow i^{\hat{H}}_{\text{end}} - 1$ $\triangleright$ *Trim example's trajectory*
    else if $\textsc{len}(\hat{E}) > 0$ **then**
        $\hat{E} \leftarrow \hat{E} \setminus \hat{E}_{-1}$ $\triangleright$ *Trim experience*
    else
        $\tau^{H} \leftarrow \tau^{H} - 1$ $\triangleright$ *Decrease the thresholds*
        $\tau^{\hat{H}} \leftarrow \tau^{\hat{H}} - 1$
    end if
end while

doi:10.6342/NTU202404517

mat required by the environment. Upon successful execution, the environment generates an observation, which is then appended to the current trajectory in the prompt. The observation is also used in constructing trajectories in the experience learner. We detail our admissible action translator component as follows.

4.2.1 Admissible Action Translator

The text-based environment we utilize, ScienceWorld, requires inputs to adhere to a strict format, posing a challenge for LLMs which typically generate human-friendly free-form text. To address this issue, previous work [15] matches raw actions with admissible actions provided by the environment, thereby minimizing syntactic errors and ensuring that actions are executable. We adopt this approach by using *SentenceBERT* [36] to translate raw actions into the most similar admissible actions based on semantic similarity using cosine distance. However, we introduce an additional threshold to prevent the execution of actions that deviate too far from the original intent. If this similarity score is below the threshold, the raw action is executed as is to preserve the agent's intended behavior.

4.3 Experience Learner

The Experience Learner module integrates both construction and refinement processes, enhanced from SSO's skill learner and customized for MENSA's experience structure. We will discuss the process of constructing experiences in Section 4.3.1, and the method for refining experiences in Section 4.3.2.

doi:10.6342/NTU202404517

4.3.1 Experience Set Construction

In the construction phase, our goal is to build reusable and transferable experience set. As shown in Figure 4.4, we begin by extracting potential subtrajectories, which are scored using heuristics and later sampled through beam search, ultimately leading to subgoal generation to form the experience. The extraction process considers multiple similar subtrajectories, with frequency indicating a high likelihood of reuse and transferability. Scoring heuristics are applied according to the formula ($w_1 \cdot$ similarity score $+ w_2 \cdot$ reward score $+ w_3 \cdot$ length score), using weights of 1, 0.1, and 0.01, respectively. These subtrajectory pair scores guide the beam search, ensuring that no overlapping subtrajectories exist within an experience. Finally, we employ the same LLM used by the actor to generate generalizable subgoals that maintain a format similar to the original observation. We design the prompt with a simple few-shot example per task, as displayed in Listing 4.2. Unlike SSO, which summarizes subtrajectories into higher-level instructions through a complex procedure, MENSA preserves the original action-observation pairs. This approach reduces cognitive load on the LLM actor by maintaining consistency in expression between instructions and trajectories, enabling the model to process and utilize experience data more effectively.

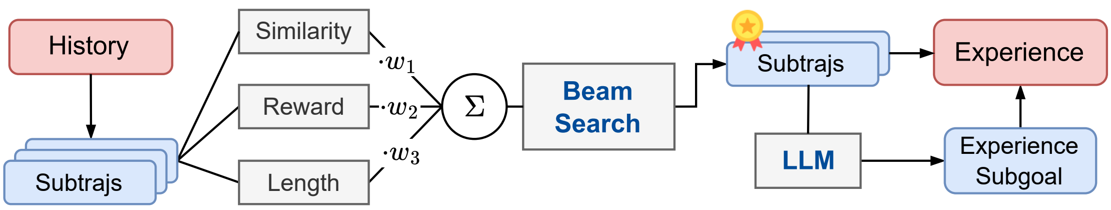


Figure 4.4: **Illustration of Experience Constructor.** Extract high-value segments from past trajectories, curate them into experiences, and define corresponding subgoals to facilitate experience retrieval.

doi:10.6342/NTU202404517

4.3.2 Experience Set Refinement

Experience refinement process focuses on filtering out less impactful experiences. This process involves tracking the sum of discount future reward for each experience, which is calculated based on the trajectory from past episodes. Similar to SSO, we apply the reward to an experience by recording the instances when it was retrieved. Only useful experiences are retained for future use.

Listing 4.2: **Prompt template for Experience Constructor in MENSA**.

```
You are an expert analyst system. Consider the trajectories that
   include the states and actions below. Generate a single TARGET
   OBSERVATION that indicates the end of the trajectories.

Here are some examples:
<Experience Trajectories_1>
<Experience Subgoal_1>

<Experience Trajectories_2>
<Experience Subgoal_2>

<Experience Trajectories_3>
<Experience Subgoal_3>

Now it is your turn:
<Experience Trajectories>
```


doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

Chapter 5

Experiments and Results

In this chapter, we will present the details of the benchmark, experiment setup, implementation specifics and results. First, we introduce *ScienceWorld*[1] [50], the environment used to evaluate the proposed methodology. In the following section, we describe the experimental setup in detail, including the baseline approaches and implementation specifics. Subsequently, we present the experimental results, highlighting the improvements achieved over previous methodologies. Finally, in the last section, we conduct ablation studies to analyze the impact of forecasting steps, as well as the ordering and selection of experiences, on the overall performance.

5.1 Benchmark Environment

To assess the performance of our method, we employ the ScienceWorld benchmark, a complex text-based interactive environment that tests agents' abilities in scientific reasoning. The ScienceWorld environment encompasses a network of interconnected spaces with 10 distinct locations, such as a workshop, a kitchen, a green house. Similar to the ap-

[1] https://sciworld.apps.allenai.org

doi:10.6342/NTU202404517

proaches taken in CLIN and SSO, we selected 18 task classes to evaluate MENSA against existing baselines, as listed in Table B.1. We compiled several practical examples with varying difficulty as reference in Table 5.1. To ensure a diverse evaluation, we randomly select five variants for each of the 18 tasks (as shown in Table B.1), with these variants ranging between 10 and 1400, affecting critical objects (e.g., water, sugar), the starting location (e.g., hallway, workshop), and the environmental content (e.g., table, apple, stopwatch). Rewards are assigned on a scale from 0 to 100 based on the completion of the subtasks, which divide the main goal into 2 to 15 subgoals, with the maximum allowed steps set to 1.5 times the length of the gold trajectory.

Task	Task description
Find Living Thing (S)	Your task is to find a(n) living thing. First, focus on the thing. Then, move it to the orange box in the living room.
Lifespan, Longest (S)	Your task is to find the animal with the longest life span. The animals are in the 'outside' location. Focus on the animal with the longest life span.
Chemistry (M)	Your task is to use chemistry to create the substance 'sugar water'. A recipe and some of the ingredients might be found near the kitchen. When you are done, focus on the sugar water.
Life Stage, Plant (M)	Your task is to focus on the life stages of the lemon plant, starting from earliest to latest. The plants are located outside.
Grow Fruit (L)	Your task is to grow a orange. This will require growing several plants, and them being cross-pollinated to produce fruit. Seeds can be found in the bathroom. To complete the task, focus on the grown orange.
Genetics, Know (L)	Your task is to determine whether round seed shape is a dominant or recessive trait in the pea plant. If the trait is dominant, focus on the red box. If the trait is recessive, focus on the green box.

Table 5.1: **Task Descriptions for Varying Complexity Levels in ScienceWorld.** Taking task "Create Sugar Water (Chemistry)" as an example, the agent must follow the task description to first find the recipe and ingredients, then successfully perform the experiment. The task is only considered complete once the agent has obtained the sugar water and focuses on it.

doi:10.6342/NTU202404517

5.2 Experiment Setup

MENSA is evaluated on in two settings: An adaptation setting to showcase its capability for continuous learning within the same task, and a transfer setting to assess its ability to generalize acquired experiences to new tasks.

- **Adaptation** setting allows agent to have 5 attempts for each test variants, with memory being learned between episodes. At the beginning, every agent start with an empty memory and a same initial starting state which determined by the variant.
- **Transfer** setting allows agent to learn on over 15 episodes, comprising 5 different train variants, each with 3 trajectories, to evaluate its capacity to generalize acquired experiences to new tasks. Following training, we assess performance on the same test variants used in the adaptation setting.

Intermediate rewards are given by the environment when completing a subtasks. A task is considered successfully completed when it achieves a cumulative reward or final score of 100. We adopt the same one-shot example like in SwiftSage for ReAct-based approaches including ReAct, Reflexion, and MENSA. Additionally, to maintain consistency of the true maximum action steps across methods, the "`think`" steps in ReAct, Reflexion, and MENSA that have not actually been executed are excluded from the step counting.

5.2.1 Baselines

To ensure a fair comparison, we include finetuning-free baseline methods that employ few-shot in-context learning approaches and previous state-of-the-art. The following methods are selected for this comparison:

doi:10.6342/NTU202404517

- **ReAct** [54] synergizes between reasoning and action traces, helping agents effectively solve complex tasks by generating additional '`think`' steps for reasoning and the intended next actions.

- **Reflexion** [39] instructs the ReAct-based LLM to generate verbal reflections between episodes, emphasizing the analysis of previous unsuccessful attempts. This process aims to facilitate learning past mistakes to improve performance in the subsequent trials.

- **SSO** [33] adopts comprehensive instructions to gather skill set from past trajectories over long-term episodes. This approach allows for the accumulation and refinement of skills over episodes.

5.2.2 Implementation Details

To highlight our contributions, we evaluate LLMs across a spectrum of capabilities, from larger, more powerful models to smaller, more constrained ones. For our experiments, we select a highly capable closed-source model, `GPT-4o-mini`, accessed via the OpenAI API[1]. In addition, we utilize open-source models with parameter counts ranging from 2B to 9B, as detailed in Table 5.2. We use the base (pre-trained) LLMs across all methods, except for `Phi3-small`, `Phi3-mini`, and all SSO experiments, which employ instruction-tuned models due to the unavailability of a base model and the method's incompatibility using base-model approach. We further group these open-source LLMs into two categories: mini and small-sized models. For open-source models, we utilize vLLM[2] [23] to manage the API.

[1] https://openai.com/index/openai-api/
[2] https://docs.vllm.ai/

doi:10.6342/NTU202404517

For the text embeddings in experience retrieval (state-based) and experience construction, we utilize SentenceBERT `paraphrase-MiniLM-L6-v2`[1] model to extract the embeddings. This approach is consistently applied across all methods involved in text similarity measurement. For generative text classifier used in experience retrieval (target-based), `BART-large-mnli`[2] model [24] is used.

5.3 Experiment Results

We report the result of MENSA compared with the existing baselines and previous SOTA in Table 5.2, depict them visually in Figure 5.1a and provide detailed performance for each approach combined with different LLMs across tasks in Tables C.1 to C.4. On both adaptation and transfer tasks, MENSA outperforms the all the existing methods using LLMs with various capabilities and sizes. In adaptation setting, MENSA outperforms previous SOTA (SSO), achieving a maximum improvement of **32.4** (122.7%) points with the moderately capable small-sized LLM, `Phi-3-small`. On the other hand, MENSA demonstrates gains of **11.9** (57.5%) and **15.8** (29%) on mini-sized model (`Phi-3-mini`) and the more powerful closed-source LLM (`GPT-4o-mini`), respectively. For transfer setting, we conduct experiments using two representative models, which are `Llama-3-8B` and `Gemma-2-9B`. MENSA surpasses previous SOTA by an improvement of **18.6** (89.4%) using `Llama-3-8B`, and **19.9** (125.2%) using `Gemma-2-9B`.

We report the detailed results using adaptation setting on the ScienceWorld tasks, which are categorized into short (S), medium (M), and long (L) based on the ground-truth number of steps required to complete the task. Following the category splitting in

[1] https://huggingface.co/sentence-transformers/paraphrase-MiniLM-L6-v2
[2] https://huggingface.co/facebook/bart-large-mnli

doi:10.6342/NTU202404517

LLM	ReAct	Reflexion	SSO	MENSA	Δ	Δ(%)
Closed-Source Large						
`GPT-4o-mini`	22.4*	25.9*	54.5*	**70.3***	+15.8	+29%
Open-Source Small						
`Gemma-2-9B`	29.9	31.2	30.7*	**42.0**	+10.8	+34.6%
`Llama-3-8B`	26.7	31.9	41.6*	**45.0**	+3.4	+8.2%
`Phi-3-small` (7B)	18.1*	23.6*	26.4*	**58.8***	+32.4	+122.7%
`Mistral-7B`	24.5	26.4	13.1*	**44.3**	+17.9	+68%
Open-Source Mini						
`Phi-3-mini` (4B)	6.5*	7.2*	20.7*	**32.6***	+11.9	+57.5%
`Gemma-2-2B`	17.9	20.6	10.5*	**29.9**	+9.3	+45.1%

* Using instruction-tuned models
Δ: Difference between MENSA and top-2 method

Table 5.2: **Main Results under Adaptation Setting**. Performance of different methods across different LLMs in adaptation setting.

LLM	SSO	MENSA	Δ	Δ(%)
`Gemma-2-9B`	15.9*	**35.8**	+19.9	+125.2%
`Llama-3-8B`	20.8*	**39.4**	+18.6	+89.4%

* Using instruction-tuned models
Δ: Difference between MENSA and top-2 method

Table 5.3: **Main Results under Transfer Setting.** Performance of MENSA and SSO across `Gemma-2-9B` and `Llama-3-8B` in transfer setting.

doi:10.6342/NTU202404517

SwiftSage, short tasks consist of fewer than 20 steps, medium tasks range between 20 and 50 steps, and long tasks involve more than 50 steps. We evaluated three representative models of varying sizes, from large to mini, including `GPT-4o-mini`, `Phi-3-small`, and `Phi-3-mini`. As depicted in Figures 5.1b, 5.1c and 5.1d, ReAct and Reflexion perform well on short tasks but struggle with longer ones. SSO excels at long tasks but requires a larger, more capable model such as GPT-4o-mini to operate effectively. However, its performance declines significantly with weaker models. On the other hand, MENSA outperforms all existing methods across all task categories, emphasizing superior performance with minimal degradation even when using a smaller model.

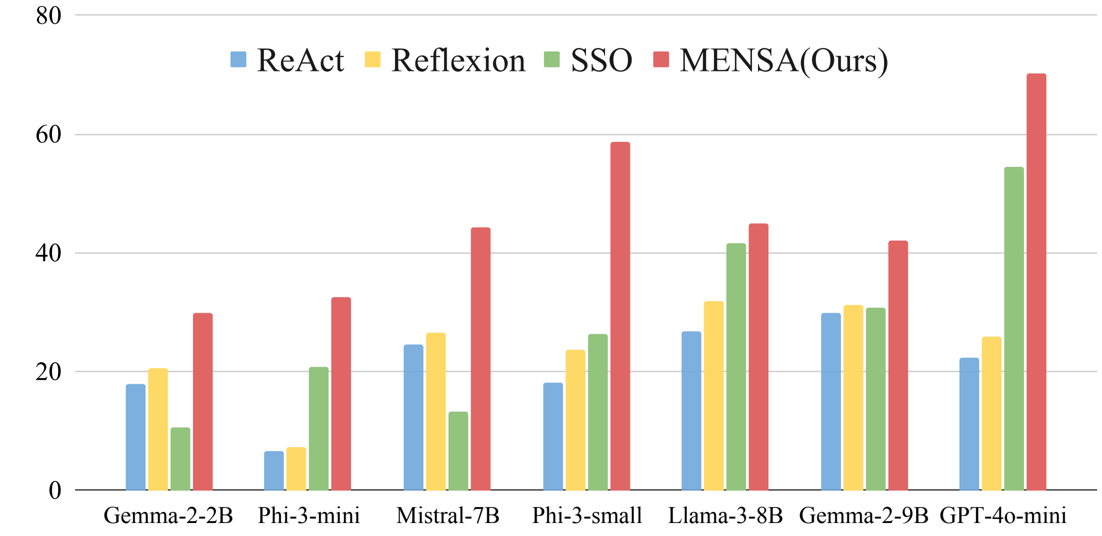
5.4 Ablation Study

We conduct several ablation experiments to evaluate the impact of various factors on experience retrieval within MENSA. These factors include the number of forecast steps considered, the ordering in which experiences are presented in the prompt, and the comparison between state-based and target-based retrieval methods. For consistency, we utilized the representative for smaller LLM and widely-recognized `Llama-3-8B` model across all ablation experiments.

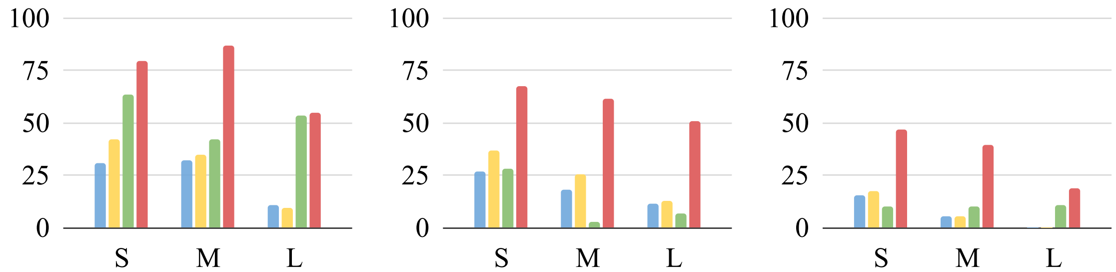
5.4.1 Forecast Steps in Experience Retrieval.

In the experience retrieval process within MENSA, an initial segment of the forecast is used to match with entries in the experience set. To determine the optimal number of forecast steps, we conducted an ablation study, testing a range of step counts from 0 to 8. When the step count is 0, that means no experience will be retrieved. As shown in

doi:10.6342/NTU202404517



(a) Performance under all models and methods.



(b) `GPT-4o-mini` (c) `Phi-3-small` (d) `Phi-3-mini`

Figure 5.1: **Visualization of Main Results under Adaptation Setting.** (a) Performance of different methods using different LLMs on ScienceWorld. The x-axis is organized by model group and sorted in ascending order according to the number of parameters in each model. The results show that MENSA outperforms other methods using different LLMs. (b), (c), (d) Detailed results on adaptation setting, categorized into short (S), medium (M), and long (L) tasks.

doi:10.6342/NTU202404517

Figure 5.2, the best performance was achieved with a step count of 3, after which the effectiveness declined. The results suggest that retrieved experiences are most beneficial when they are closely aligned with the current step, aiding in the decision-making process for the next action. Additionally, we find insights related to token usage costs, which are explored further in Section 6.1.

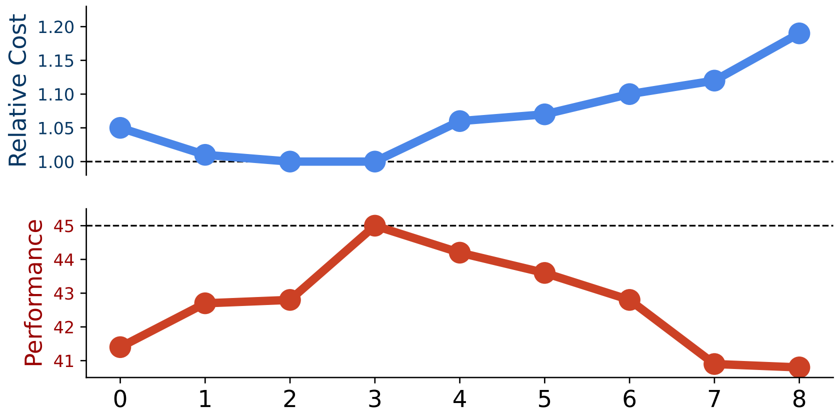


Figure 5.2: **Performance and Relative Cost for Different Forecast Step Counts.** We found that forecasting the next 3 steps yields the best results while minimizing overall token consumption.

5.4.2 Experience Ordering.

When processing longer contexts, LLMs are known to exhibit primacy and recency biases [11, 29]. Our prompts, which include a set of retrieved experiences, may also be subject to these biases. To investigate, we conducted an ablation study that compares different ordering strategies based on the initial list configuration [T_Exp, S_Exp], including reversed, non-reversed, and shuffled sequences when presented to the LLM. In this context, a sequence is considered reversed if it is ordered from the least to the most relevant experience based on the similarity score relative to the current target or state. Our results in Table 5.4 indicate that reversed ordering yields the highest performance, scoring **45.0**, which is **2.2** points higher than the next best approach (Shuffled), while non-reversed

doi:10.6342/NTU202404517

ordering has the poorest performance. This result suggests that recency bias plays a role in our scenario, making reverse ordering a beneficial strategy. For that reason, we adopt reverse ordering for all experiments discussed in the main results in Section 5.3.

Method	Category order	Internal order	Score
Shuffled	-	-	42.8
T_Exp prioritized, non-reversed	T, S	Descending	42.4
T_Exp prioritized, reversed	S, T	Ascending	**45.0**

T, T_Exp: Target-based experiences.
S, S_Exp: State-based experiences

Table 5.4: **Results of Experience-Ordering Methods.**

5.4.3 Experience Retrieval Approaches.

In our comparison of approaches for retrieving relevant experiences during mental simulation, we evaluated several strategies: state-based similarity comparison, target-based comparison, and a combination of both, with varying orderings. Specifically, we examined whether placing state-based experiences (S_Exp) before or after target-based experiences (T_Exp) would yield better results and whether primacy and recency biases affect all experimental configurations. Building on our earlier findings regarding experience ordering, we reversed the sequence within each category. As shown in Table 5.5, the best performance was achieved when target-based experiences were prioritized. This result underscores the value of including both state-based and target-based experiences, with a notable advantage when target-based experiences are treated as the primary basis.

doi:10.6342/NTU202404517

Method	Category order	Internal order	Score
T_Exp only	T	Descending	43.4
S_Exp only	S	Descending	39.1
T_Exp prioritized	T, S	Descending	42.9
S_Exp prioritized	S, T	Descending	42.4
T_Exp only, reversed	T	Ascending	43.9
S_Exp only, reversed	S	Ascending	41.4
T_Exp prioritized, reversed	S, T	Ascending	**45.0**
S_Exp prioritized, reversed	T, S	Ascending	42.9

T, T_Exp: Target-based experiences.
S, S_Exp: State-based experiences

Table 5.5: **Performance Comparison of Experience Example Ordering Approaches.** The elements in experience set S_Exp, T_Exp are ranked in descending order of relevance by default.

doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

Chapter 6

Discussion

In this chapter, we extend the previous ablation study in Section 5.4.1 to further analyze the impact of the forecast step in the Experience Retriever on overall cost, as presented in Section 6.1. Additionally, in Section 6.2, we explore the differences in capability requirements between MENSA and the previous SOTA approach, SSO, and assess whether our method is indeed more applicable to LLMs of varying sizes compared to SSO.

6.1 How does the number of the forecast step affects performance and token cost?

To understand how the number of forecast steps affects the agent, we plot the performance score and the token cost for different numbers of forecast steps as shown in Figure 5.2. The cost calculation is based on the sum of input tokens and three times the output tokens, reflecting a similar pricing structure to the OpenAI API. We show the relative cost compared to that of the 3 forecast steps. We find that the performance increases and the cost decreases for the number of forecast steps ≤ 3. This shows that the forecast not only improves the agent's score but also helps reduce the cost because the agent can

doi:10.6342/NTU202404517

complete their tasks sooner. However, a further increase in the number of forecast steps showed a negative impact on the performance and token cost. We identify two major failure cases when increasing the forecast steps. First, LLMs often struggle to generate the most relevant actions and outcomes for the later steps in the forecast because they are still imperfect world models. Second, even when the forecast is correct, the LLMs may fail to select a good next action because of the overload of information from future steps and LLMs often treat the additional information as noise. This often results in a higher incidence of erroneous actions, which not only lowers success rates but also introduces additional steps to correct these errors. Consequently, token usage increases as more forecast steps are used.

6.2 How does LLM capability affect performance?

Large Language Models possess a diverse range of abilities that enable them to perform complex tasks across various domains. These abilities include instruction following, planning sequences of actions, reasoning through problems, and understanding. Each of these abilities contributes to the model's effectiveness in applications such as decision-making, and interactive systems. The capabilities and the inherent strengths are not uniform across different LLMs, and they can vary depending on the model size and training data. A method that uses LLMs for a specific task, i.e. sequential decision making, is supposed to be favorable if the performance does not suffer from dramatic degradation when using models with weaker abilities.

Recent research on the evaluation of LLM abilities has particularly focused on tool utilization such as T-Eval [6], which shares common ground with sequential decision-

doi:10.6342/NTU202404517

making tasks, especially in instruction-following and planning. We assess the performance of various models on instruction-following and planning abilities defined in T-Eval, aiming to understand how differences in these abilities impact the effectiveness of different methods. We report the result of the ability score of different LLMs, as shown in Table 6.1. The following analysis examines the impact of MENSA and existing methods in adaptation settings when applied to LLMs with varying capabilities.

	Instruct	*Plan*	ReAct	Reflexion	SSO	MENSA
`GPT-4o-mini`	100.0	84.2	22.4	25.9	54.5	70.3
`Gemma-2-9B`	99.8	65.0	29.9	31.2	30.7	42.0
`Llama-3-8B`	100.0	60.5	26.7	31.9	41.6	45.0
`Phi-3-small`	87.2	64.8	18.1	23.6	26.4	58.8
`Mistral-7B`	99.2	56.1	24.5	26.4	13.1	44.2
`Phi-3-mini`	95.1	61.6	6.5	7.2	20.7	32.6
`Gemma-2-2B`	99.9	44.4	17.9	20.6	10.5	29.9

Table 6.1: **Comparison of *Instruct* and *Plan* Abilities to Performance.** The *Instruct* and *Plan* abilities of each LLM are evaluated on T-Eval, along with the performance of each LLM using methods within the adaptation setting.

The results show that for ReAct and Reflexion, a noticeable performance decline occurs when using `Phi-3-mini`, which has weaker abilities. However, when either the instruction-following or planning ability is relatively strong, the decline in performance is less pronounced. In contrast, SSO’s performance is severely impacted when the planning ability falls below 60%, and weaker instruction-following ability also negatively affects its performance, as observed in `Phi-3-mini` and `Phi-3-small`. This suggests that SSO heavily relies on planning, consistent with its dependence on inferring subgoals for skill retrieval. Moreover, more complex instructions also hinder the performance of models that are not proficient in instruction-following. MENSA, however, demonstrates better resilience, maintaining stabler performance regardless of the varying strengths of these abilities. This indicates that MENSA’s mental simulation mechanism effectively lever-

doi:10.6342/NTU202404517

ages basic capabilities, enabling smaller LLMs to achieve greater success in sequential decision-making tasks.

doi:10.6342/NTU202404517

Chapter 7

Conclusion

In this final chapter, the findings of the study are summarized, key contributions are highlighted, and the future study of the research are discussed.

7.1 Conclusion

This study introduces MENSA, an LLM agent that leverages mental simulation to dynamically retrieve relevant experiences from a continuously constructed experience set across episodes. MENSA outperforms the existing state-of-the-art methods in the text-based interactive environment, ScienceWorld, not only with a strong model but also with much smaller LLMs. Our analysis on T-Eval shows that MENSA can effectively leverage basic capabilities, enabling smaller LLMs to achieve greater success in sequential decision-making tasks.

doi:10.6342/NTU202404517

7.2 Limitation and Future Work

The following presents the current limitations of our proposed approach and discusses opportunities for further improvement.

- **Imperfect World Models in LLMs.** Due to the default configuration of LLM-based agent, the LLM typically plays a leading role. One limitation of MENSA is its reliance on the encoded world models in LLMs to simulate actions and outcomes. As a result, the quality of the implicit world model will affect the performance of the agent. Future work on improving knowledge and world models in LLMs can further improve agents.

- **Long Prompt Condensing.** In this study, we employ a dynamic prompt trimming method that eliminates the most temporally distant segments at each step to streamline the prompt. However, future work explore more sophisticated method to achieve more precise context condensing without compromising performance.

- **Selection of Experience Example.** Currently, we randomly select previously executed sub-trajectories from the experience to guide the decision maker. Future research can explore ways to improve the efficiency of information usage, creating more transferable examples with a clear and comprehensible structure.

doi:10.6342/NTU202404517

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.

[2] S. Arora, A. Narayan, M. F. Chen, L. Orr, N. Guha, K. Bhatia, I. Chami, and C. Re. Ask me anything: A simple strategy for prompting language models. In The Eleventh International Conference on Learning Representations, 2023.

[3] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk, and T. Hoefler. Graph of thoughts: Solving elaborate problems with large language models. Proceedings of the AAAI Conference on Artificial Intelligence, 38(16):17682–17690, Mar. 2024.

[4] K. Bousmalis, G. Vezzani, D. Rao, C. M. Devin, A. X. Lee, M. B. Villalonga, T. Davchev, Y. Zhou, A. Gupta, A. Raju, et al. Robocat: A self-improving generalist agent for robotic manipulation. Transactions on Machine Learning Research, 2024.

[5] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree

doi:10.6342/NTU202404517

search methods. IEEE Transactions on Computational Intelligence and AI in games, 4(1):1–43, 2012.

[6] Z. Chen, W. Du, W. Zhang, K. Liu, J. Liu, M. Zheng, J. Zhuo, S. Zhang, D. Lin, K. Chen, and F. Zhao. T-eval: Evaluating the tool utilization capability of large language models step by step. In L.-W. Ku, A. Martins, and V. Srikumar, editors, Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9510–9529, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics.

[7] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

[8] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, T. Liu, B. Chang, X. Sun, L. Li, and Z. Sui. A survey on in-context learning, 2024.

[9] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models, 2024.

[10] M. Gao, X. Hu, J. Ruan, X. Pu, and X. Wan. Llm-based nlg evaluation: Current status and challenges, 2024.

[11] X. Guo and S. Vosoughi. Serial position effects of large language models, 2024.

[12] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding, 2021.

[13] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, J. Wang, Z. Wang,

doi:10.6342/NTU202404517

S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2023.

[14] D. Huang, J. M. Zhang, M. Luck, Q. Bu, Y. Qing, and H. Cui. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation, 2024.

[15] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 9118–9147. PMLR, 17–23 Jul 2022.

[16] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In K. Liu, D. Kulic, and J. Ichnowski, editors, Proceedings of The 6th Conference on Robot Learning, volume 205 of Proceedings of Machine Learning Research, pages 1769–1782. PMLR, 14–18 Dec 2023.

[17] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, et al. Do as i can, not as i say: Grounding language in robotic affordances. In K. Liu, D. Kulic, and J. Ichnowski, editors, Proceedings of The 6th Conference on Robot Learning, volume 205 of Proceedings of Machine Learning Research, pages 287–318. PMLR, 14–18 Dec 2023.

[18] R. Islam and O. M. Moushi. Gpt-4o: The cutting-edge advancement in multimodal llm. Authorea Preprints, 2024.

[19] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas,

doi:10.6342/NTU202404517

F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. arXiv preprint arXiv:2310.06825, 2023.

[20] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand, et al. Mixtral of experts. arXiv preprint arXiv:2401.04088, 2024.

[21] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim. A survey on large language models for code generation. arXiv preprint arXiv:2406.00515, 2024.

[22] W. Jiao, W. Wang, J. tse Huang, X. Wang, S. Shi, and Z. Tu. Is chatgpt a good translator? a preliminary study. In ArXiv, 2023.

[23] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles, 2023.

[24] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[25] S. Li, X. Puig, C. Paxton, Y. Du, C. Wang, L. Fan, T. Chen, D.-A. Huang, E. Akyürek, A. Anandkumar, et al. Pre-trained language models for interactive decision-making. Advances in Neural Information Processing Systems, 35:31199–31212, 2022.

doi:10.6342/NTU202404517

[26] T. Li, G. Zhang, Q. D. Do, X. Yue, and W. Chen. Long-context llms struggle with long in-context learning, 2024.

[27] X. Liang, H. Wang, Y. Wang, S. Song, J. Yang, S. Niu, J. Hu, D. Liu, S. Yao, F. Xiong, et al. Controllable text generation for large language models: A survey. arXiv preprint arXiv:2408.12599, 2024.

[28] B. Y. Lin, Y. Fu, K. Yang, F. Brahman, S. Huang, C. Bhagavatula, P. Ammanabrolu, Y. Choi, and X. Ren. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, Advances in Neural Information Processing Systems, volume 36, pages 23813–23825. Curran Associates, Inc., 2023.

[29] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, 12:157–173, 2024.

[30] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, et al. Agentbench: Evaluating llms as agents. arXiv preprint arXiv:2308.03688, 2023.

[31] T. Lu, M. Gao, K. Yu, A. Byerly, and D. Khashabi. Insights into llm long-context failures: When transformers know but don't tell, 2024.

[32] B. P. Majumder, B. D. Mishra, P. Jansen, O. Tafjord, N. Tandon, L. Zhang, C. Callison-Burch, and P. Clark. Clin: A continually learning language agent for rapid task adaptation and generalization, 2023.

[33] K. Nottingham, B. P. Majumder, B. Dalvi Mishra, S. Singh, P. Clark, and R. Fox. Skill set optimization: Reinforcing language model behavior via transferable skills.

doi:10.6342/NTU202404517

In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, Proceedings of the 41st International Conference on Machine Learning, volume 235 of Proceedings of Machine Learning Research, pages 38409–38425. PMLR, 21–27 Jul 2024.

[34] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, L. Hong, R. Tian, R. Xie, J. Zhou, M. Gerstein, dahai li, Z. Liu, and M. Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In The Twelfth International Conference on Learning Representations, 2024.

[35] Y. Qin, K. Song, Y. Hu, W. Yao, S. Cho, X. Wang, X. Wu, F. Liu, P. Liu, and D. Yu. InFoBench: Evaluating instruction following ability in large language models. In L.-W. Ku, A. Martins, and V. Srikumar, editors, Findings of the Association for Computational Linguistics ACL 2024, pages 13025–13048, Bangkok, Thailand and virtual meeting, Aug. 2024. Association for Computational Linguistics.

[36] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 11 2019.

[37] N. Robinson, P. Ogayo, D. R. Mortensen, and G. Neubig. ChatGPT MT: Competitive for high- (but not low-) resource languages. In P. Koehn, B. Haddow, T. Kocmi, and C. Monz, editors, Proceedings of the Eighth Conference on Machine Translation, pages 392–418, Singapore, Dec. 2023. Association for Computational Linguistics.

[38] T. Schick and H. Schütze. It's not just size that matters: Small language models are also few-shot learners. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou,

doi:10.6342/NTU202404517

editors, Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2339–2352, Online, June 2021. Association for Computational Linguistics.

[39] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, Advances in Neural Information Processing Systems, volume 36, pages 8634–8652. Curran Associates, Inc., 2023.

[40] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv:2206.04615, 2022.

[41] S. E. Taylor and S. K. Schneider. Coping and the simulation of events. Social Cognition, 7(2):174–194, 1989.

[42] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, et al. Gemini: A family of highly capable multimodal models, 2024.

[43] G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, A. Gulati, G. Tanzer, D. Vincent, Z. Pan, S. Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.

[44] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al. Gemma 2: Improving open language models at a practical size. arXiv preprint arXiv:2408.00118, 2024.

doi:10.6342/NTU202404517

[45] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

[46] P. Veličković, A. P. Badia, D. Budden, R. Pascanu, A. Banino, M. Dashevskiy, R. Hadsell, and C. Blundell. The clrs algorithmic reasoning benchmark, 2022.

[47] V. Vidal et al. A lookahead strategy for heuristic search planning. In ICAPS, pages 150–160, 2004.

[48] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. Voyager: An open-ended embodied agent with large language models. Transactions on Machine Learning Research, 2024.

[49] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J. Wen. A survey on large language model based autonomous agents. Frontiers of Computer Science, 18(6), Mar. 2024.

[50] R. Wang, P. Jansen, M.-A. Côté, and P. Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 11279–11298, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[51] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, Advances in Neural Information Processing Systems, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

doi:10.6342/NTU202404517

[52] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, et al. The rise and potential of large language model based agents: A survey. arXiv preprint arXiv:2309.07864, 2023.

[53] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. R. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In Thirty-seventh Conference on Neural Information Processing Systems, 2023.

[54] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In The Eleventh International Conference on Learning Representations, 2023.

[55] W. Yin, J. Hay, and D. Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach, 2019.

[56] A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang. Expel: Llm agents are experiential learners. Proceedings of the AAAI Conference on Artificial Intelligence, 38(17):19632–19642, Mar. 2024.

[57] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. arXiv preprint arXiv:2303.18223, 2023.

[58] J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou. Instruction-following evaluation for large language models. arXiv preprint arXiv:2311.07911, 2023.

doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

Appendix A — Examples

Below, we present a practical example illustrating the decision-making process of the actor in our proposed approach, including the LLM input prompt, output, parsed action, and forecast.

A.1 Decision-Making Example

Task Description:

```
Your task is to use chemistry to create the substance 'sugar water
    '. A recipe and some of the ingredients might be found near the
    workshop. When you are done , focus on the sugar water.
```

LLM Input:

```
Interact with a household to solve a task. Keep going until the task completes.
```

```
You have access to the following actions: look around, show task, show
inventory, read OBJ, activate OBJ, deactivate OBJ, open OBJ, close OBJ, pick up
OBJ, look in OBJ, look at OBJ, focus on OBJ, flush OBJ, eat OBJ, dunk OBJ in
OBJ, move OBJ to CONTAINER, mix CONTAINER, teleport LOCATION, go LOCATION,
wait.
```

```
Here is an example:
This room is called the hallway. In it, you see:   the agent  a
    substance called air  a picture You also see:  A door to the art
    studio (that is open)  A door to the bedroom (that is open)  A door
     to the greenhouse (that is open)  A door to the kitchen (that is
    open)  A door to the living room (that is open)  A door to the
    workshop (that is open)
```


doi:10.6342/NTU202404517

```
Your task is to use chemistry to create the substance 'salt water'. A
   recipe and some of the ingredients might be found near the kitchen.
    When you are done, focus on the salt water.
> think: To solve the task, I need to move to a new location and be in
   same location as sodium chloride
OK.
> teleport to kitchen
You teleport to the kitchen.
> think: Now I move to a new location and am in same location as sodium
    chloride. Next, I need to read recipe
OK.
> open cupboard
The cupboard is now open.
> pick up cup containing nothing in table
You move the glass cup to the inventory.
> pick up recipe
You move the recipe to the inventory.
> read recipe in inventory
The recipe reads: To make salt water, you need to mix sodium chloride,
   water.
> think: Now I read recipe. Next, I need to be in same location as
   water and have all ingredients alone in a single container
OK.
> open freezer
The freezer is now open.
> open oven
The oven is now open.
> open fridge
The fridge is now open.
> open drawer in cupboard
The drawer is now open.
> open drawer in counter
The drawer is now open.
> open glass jar
The glass jar is now open.
> pick up sodium chloride
You move the sodium chloride to the inventory.
> move sodium chloride to cup containing nothing in inventory
You move the sodium chloride to the glass cup.
> move cup containing sodium chloride to sink
You move the glass cup to the sink.
> activate sink
The sink is now activated.
> think: Now I am in same location as water and have all ingredients
   alone in a single container. Next, I need to be in same location as
    salt water
OK.
> deactivate sink
The sink is now deactivated.
> pick up cup containing sodium chloride and water
You move the glass cup to the inventory.
> pour cup containing sodium chloride and water into cup containing
   sodium chloride and water
You can't move something into itself.
> examine cup containing sodium chloride and water
a glass cup (containing a substance called sodium chloride, a substance
    called water)
> mix cup containing sodium chloride and water
```


doi:10.6342/NTU202404517

```
Sodium chloride and water mix to produce salt water.
> think: Now I am in same location as salt water. Next, I need to focus
    on result chemical (salt water)
OK.
> focus on salt water
You focus on the salt water.
> think: Now I focus on result chemical (salt water)
OK.
> wait1
You decide to wait for 1 iterations.
```

The following examples contain potentially useful information:

```
You move the sugar to the inventory. This room is called the workshop.
   In it, you see:    the agent   a substance called air  A recipe
   titled instructions to make sugar water  a table. On the table is:
   a battery, a black wire, a blue wire, a green light bulb, which is
   off, a orange light bulb, which is off, a orange wire, a red light
   bulb, which is off, a switch, which is off.   a ultra low
   temperature freezer. The ultra low temperature freezer door is
   closed.  You also see:   A door to the hallway (that is open)  In
   your inventory, you see:   an orange   a substance called sugar
> pick up recipe
You move the recipe to the inventory.
> read recipe on inventory
The recipe reads: To make sugar water, you need to mix sugar, water.

You teleport to the workshop. This room is called the workshop. In it,
   you see:   the agent   a substance called air  A recipe titled
   instructions to make sugar water  a substance called sugar  a table
   . On the table is: a battery, a black wire, a blue wire, a green
   light bulb, which is off, a orange light bulb, which is off, a
   orange wire, a red light bulb, which is off, a switch, which is off
   .   a ultra low temperature freezer. The ultra low temperature
   freezer door is closed.  You also see:   A door to the hallway (
   that is open)  In your inventory, you see:   an orange
> look around
This room is called the workshop.
> think: now that i'm in the workshop, i need to read the recipe for
   sugar water.
OK.

Sugar and water mix to produce sugar water. This room is called the
   kitchen. In it, you see:    the agent   a substance called air  a
   chair. On the chair is: nothing.  a counter. On the counter is: a
   bowl (containing a red apple, a banana, an orange, a potato), a
   drawer.   a cupboard. The cupboard door is closed.    a freezer.
   The freezer door is closed.    a fridge. The fridge door is closed.
       a glass jar (containing a substance called sodium chloride)   a
    lighter   a oven, which is turned off. The oven door is closed.
   a painting  a sink, which is turned on. In the sink is: nothing.  a
    substance called soap   a stopwatch, which is deactivated.    a
   stove, which is turned off. On the stove is: nothing.   a table. On
    the table is: nothing.  a thermometer, currently reading a
   temperature of 10 degrees celsius You also see:  A door to the
   bathroom (that is open)   A door to the hallway (that is open)  A
   door to the outside (that is open)  In your inventory, you see:   a
    glass cup (containing a substance called sugar water)   an orange
```


doi:10.6342/NTU202404517

```
> think: now that i have created sugar water, i need to focus on the
   result chemical (sugar water).
OK.
> focus on sugar water
You focus on the sugar water.

OK. This room is called the kitchen. In it, you see:    the agent   a
   substance called air  a chair. On the chair is: nothing.  a counter
   . On the counter is: a bowl (containing a red apple, a banana, an
   orange, a potato), a drawer.   a cupboard. The cupboard door is
   closed.    a freezer. The freezer door is closed.    a fridge. The
   fridge door is closed.    a glass jar (containing a substance
   called sodium chloride)   a lighter   a oven, which is turned off.
   The oven door is closed.   a painting  a sink, which is turned on.
   In the sink is: a glass cup (containing a substance called water).
    a substance called soap   a stopwatch, which is deactivated.    a
   stove, which is turned off. On the stove is: nothing.   a table. On
    the table is: nothing.  a thermometer, currently reading a
   temperature of 10 degrees celsius You also see:  A door to the
   bathroom (that is open)   A door to the hallway (that is open)  A
   door to the outside (that is open)  In your inventory, you see:
   an orange   a substance called sugar
> think: i need to find a way to fill the glass cup with water from the
    sink. since the sink is activated, i should be able to collect
   water without issues. perhaps i should try using the cup directly
   without any additional commands.
OK.
> put glass cup to sink
You move the glass cup to the sink.
```

Now it is your turn:

```
This outside location is called the outside. Here you see:   the agent
    a substance called air  an axe  a fire pit (containing nothing)  a
    fountain (containing nothing)  the ground  a shovel  a substance
   called wood You also see:  A door to the foundry (that is open)  A
   door to the greenhouse (that is open)  A door to the kitchen (that
   is open) In your inventory, you see:  an orange
Your task is to use chemistry to create the substance 'sugar water'. A
   recipe and some of the ingredients might be found near the workshop
   . When you are done, focus on the sugar water.
> think: to solve the task, i need to move to a new location and find
   the sugar as well as a recipe for sugar water.
OK.
> teleport workshop
You teleport to the workshop.
> look around
This room is called the workshop. In it, you see:   the agent  a
   substance called air  A recipe titled instructions to make sugar
   water  a substance called sugar  a table. On the table is: a
   battery, a black wire, a blue wire, a green light bulb, which is
   off, a orange light bulb, which is off, a orange wire, a red light
   bulb, which is off, a switch, which is off.  a ultra low
   temperature freezer. The ultra low temperature freezer door is
   closed.  You also see:  A door to the hallway (that is open) In
   your inventory, you see:  an orange
> think: now that i am in the workshop, i need to pick up the sugar and
    the recipe for sugar water to proceed with the task.
OK.
```


doi:10.6342/NTU202404517

```
> pick up sugar
You move the sugar to the inventory.
> pick up recipe
You move the recipe to the inventory.
> read recipe on inventory
The recipe reads: To make sugar water, you need to mix sugar, water.
> think: now i have the recipe, but i need to find water to create
    sugar water. i should teleport back to the kitchen where i know
    there's a sink with water.
OK.
> teleport kitchen
You teleport to the kitchen.
> look around
This room is called the kitchen. In it, you see:   the agent  a
    substance called air  a chair. On the chair is: nothing.  a counter
    . On the counter is: a bowl (containing a red apple, a banana, an
    orange, a potato), a drawer.  a cupboard. The cupboard door is
    closed.   a freezer. The freezer door is closed.   a fridge. The
    fridge door is closed.   a glass jar (containing a substance called
     sodium chloride)  a lighter  a oven, which is turned off. The oven
     door is closed.   a painting  a sink, which is turned off. In the
    sink is: nothing.  a substance called soap  a stopwatch, which is
    deactivated.   a stove, which is turned off. On the stove is:
    nothing.  a table. On the table is: a glass cup (containing nothing
    ).  a thermometer, currently reading a temperature of 10 degrees
    celsius You also see:  A door to the bathroom (that is open)  A
    door to the hallway (that is open)  A door to the outside (that is
    open) In your inventory, you see:  an orange  A recipe titled
    instructions to make sugar water  a substance called sugar
> think: now that i'm in the kitchen, i need to activate the sink to
    get water for the sugar water recipe.
OK.
>
```

LLM Output:

```
> activate sink
The sink is now activated.
> think: Now that the sink is activated, I can fill the glass cup with
    water. I should use the cup directly to collect the water.
OK.
> put glass cup to sink
You move the glass cup to the sink.
> think: The glass cup should be filling with water now. After that, I
    need to mix the sugar and water together.
OK.
> wait
You wait for a moment as the glass cup fills with water.
```

Parsed Action:

```
activate sink
```


doi:10.6342/NTU202404517

Parsed Forecast:

```
The sink is now activated.
> think: Now that the sink is activated, I can fill the glass cup with
    water. I should use the cup directly to collect the water.
OK.
> put glass cup to sink
You move the glass cup to the sink.
> think: The glass cup should be filling with water now. After that, I
    need to mix the sugar and water together.
OK.
```

A.2 Experience Example

Subgoal Description:

```
You activate the sink to get water for the sugar water recipe
```

Subtrajectories 1:

```
> think: now that i'm in the kitchen, i need to activate the sink to
    get water for the sugar water recipe.
OK.
> activate sink
The sink is now activated.
> think: now that the sink is activated, i need to fill the glass cup
    with water from the sink.
OK.
> put glass cup to sink
You move the glass cup to the sink.
```

Subtrajectories 2:

```
> think: i need to find water to mix with the sugar. i will check the
    sink for water.
OK.
> activate sink
The sink is now activated.
> think: now that the sink is activated and water is flowing, i need to
     fill the glass cup with water from the sink to mix it with the
    sugar.
OK.
> put glass cup to sink
You move the glass cup to the sink.
```


doi:10.6342/NTU202404517

Appendix B — Task Selection and Environment Details

Task Selection: We follow previous work by selecting 2 tasks from each topic, except for the ”Electricity” tasks, due of the inaccuracy of the evaluation criteria.

Variants: Column in Table B.1 lists the variation IDs for each task used during training and testing. We use 5 test variants for all experiments.

Task	Task ID	Length Level	Train Variants	Test Variants
Boil	1-1	L	(8, 12, 7, 6, 13)	(24, 27, 28, 25, 29)
Freeze	1-3	L	(8, 12, 7, 6, 13)	(24, 27, 28, 25, 29)
Use Thermometer	2-1	M	(138, 48, 66, 225, 181)	(428, 468, 538, 414, 533)
Boiling Point	2-2	M	(78, 170, 36, 134, 176)	(388, 336, 328, 377, 430)
Find Living Thing	4-1	S	(63, 133, 109, 134, 75)	(271, 243, 261, 233, 232)
Find Plant	4-3	S	(63, 133, 109, 134, 75)	(271, 243, 261, 233, 232)
Grow Plant	5-1	L	(56, 21, 19, 16, 18)	(120, 98, 104, 109, 119)
Grow Fruit	5-2	L	(56, 21, 19, 16, 18)	(120, 98, 104, 109, 119)
Chemistry	6-1	M	(7, 9, 5, 6, 14)	(27, 28, 30, 31, 26)
Color Mix	6-2	S	(13, 12, 4, 9, 5)	(30, 33, 34, 31, 35)
Lifespan, Longest	7-1	S	(56, 21, 19, 16, 18)	(119, 98, 103, 108, 118)
Lifespan, Shortest	7-2	S	(56, 21, 19, 16, 18)	(119, 98, 103, 108, 118)
Life Stage, Plant	8-1	M	(3, 1, 2, 4, 0)	(12, 10, 11, 13, 9)
Life Stage, Animal	8-2	S	(2, 1, 3, 0)	(8, 7, 9, 6)
Force	9-1	L	(61, 57, 39, 30, 56)	(135, 129, 136, 152, 163)
Friction	9-2	L	(8, 412, 264, 531, 39)	(1185, 1182, 1334, 1348, 1301)
Genetics, Know	10-1	L	(38, 23, 54, 11, 16)	(109, 104, 100, 116, 112)
Genetics, unKnow	10-2	L	(45, 113, 9, 85, 105)	(369, 444, 455, 474, 452)

Table B.1: **Task Details of Benchmark.** Train and Test Variants are randomly selected using *seed*=42. Short (S) tasks consist of fewer than 20 steps, medium (M) tasks range between 20 and 50 steps, and long (L) tasks involve more than 50 steps.

doi:10.6342/NTU202404517

doi:10.6342/NTU202404517

Appendix C — ScienceWorld Performance By Task

We evaluate MENSA and existing baselines on ScienceWorld using 18 selected tasks using various LLMs. The performance of React, Reflexion, SSO, and MENSA are detailed in Tables C.1 to C.4, respectively.

	LLM model						
Task	GPT-4o mini	Gemma-2 9B	Llama-3 8B	Phi-3 small	Mistral 7B	Phi-3 mini	Gemma-2 2B
Boil	2.4	0.4	0.0	0.6	0.0	0.0	0.4
Freeze	2.0	0.0	0.0	8.0	0.0	0.0	0.0
Use Thermometer	29.8	43.8	32.4	4.2	63.6	4.2	10.2
Boiling Point	33.6	52.0	47.0	39.6	47.0	0.0	47.0
Find Living	38.6	50.2	28.6	23.6	45.2	23.8	13.6
Find Plant	8.8	23.6	25.2	8.8	12.0	10.4	15.2
Grow Plant	23.8	42.4	25.4	4.4	8.2	0.0	3.8
Grow Fruit	9.6	57.0	57.0	8.8	18.0	0.0	17.4
Chemistry	23.4	21.8	20.2	20.0	35.2	18.4	35.2
Color Mix	40.0	36.0	50.0	22.0	18.0	8.0	8.0
Lifespan, Longest	40.0	60.0	60.0	50.0	60.0	30.0	50.0
Lifespan, Shortest	40.0	40.0	40.0	40.0	40.0	20.0	40.0
Life Stage, Plant	41.0	35.6	44.0	9.6	51.6	0.0	44.6
Life Stage, Animal	19.0	8.0	9.0	17.0	8.0	2.0	8.0
Force	5.0	11.0	3.0	12.0	12.0	0.0	3.0
Friction	10.0	13.0	13.0	28.0	13.0	0.0	9.0
Genetics, Know	14.2	27.2	18.0	19.6	6.4	0.0	6.6
Genetics, Unknow	21.8	17.0	8.4	10.2	3.4	0.0	9.8
Average	22.4	29.9	26.7	18.1	24.5	6.5	17.9

Table C.1: **Performance of LLMs with "ReAct" Across Tasks.**

doi:10.6342/NTU202404517

	LLM model						
Task	GPT-4o mini	Gemma-2 9B	Llama-3 8B	Phi-3 small	Mistral 7B	Phi-3 mini	Gemma-2 2B
Boil	1.2	0.4	0.0	2.4	0.0	0.0	0.4
Freeze	4.0	8.6	0.0	10.0	2.0	0.0	0.0
Use Thermometer	10.4	61.4	50.8	17.0	64.2	4.2	10.8
Boiling Point	47.2	47.2	51.6	46.6	47.0	0.0	51.2
Find Living	30.2	40.2	41.8	43.6	60.2	23.8	22.0
Find Plant	8.8	25.2	25.2	8.8	12.0	12.0	15.2
Grow Plant	5.6	9.4	42.4	5.0	9.2	0.0	10.2
Grow Fruit	15.6	39.0	21.0	12.4	18.0	0.0	20.2
Chemistry	21.8	23.4	38.4	25.0	35.2	18.4	35.2
Color Mix	54.0	42.0	40.0	34.0	32.0	8.0	10.0
Lifespan, Longest	70.0	60.0	70.0	50.0	60.0	20.0	60.0
Lifespan, Shortest	50.0	40.0	40.0	60.0	40.0	30.0	40.0
Life Stage, Plant	59.4	46.0	35.0	12.2	52.6	0.0	47.6
Life Stage, Animal	41.3	23.3	32.8	24.3	8.0	12.3	8.0
Force	7.0	31.0	23.0	14.0	12.0	0.0	4.0
Friction	10.0	32.0	32.0	28.0	13.0	0.0	9.0
Genetics, Know	14.2	19.4	20.2	19.4	7.2	0.0	13.0
Genetics, Unknow	16.0	12.2	9.2	12.6	3.4	0.0	13.8
Average	25.9	31.2	31.9	23.6	26.4	7.2	20.6

Table C.2: **Performance of LLMs with "Reflexion" Across Tasks.**

doi:10.6342/NTU202404517

	LLM model						
Task	GPT-4o mini	Gemma-2 9B	Llama-3 8B	Phi-3 small	Mistral 7B	Phi-3 mini	Gemma-2 2B
Boil	53.0	0.8	23.4	2.4	0.0	8.6	0.0
Freeze	43.2	0.0	2.0	4.0	2.0	0.0	0.0
Use Thermometer	24.4	6.6	31.0	8.6	2.4	1.2	0.0
Boiling Point	56.2	50.8	57.8	25.6	1.0	24.2	16.2
Find Living	36.8	28.6	53.4	47.0	5.2	22.0	13.4
Find Plant	12.0	13.6	13.6	22.0	18.6	8.6	10.2
Grow Plant	34.0	12.8	31.2	4.6	1.2	4.8	4.0
Grow Fruit	36.8	48.6	36.0	14.2	0.0	27.6	40.2
Chemistry	28.2	25.0	23.2	21.8	8.4	16.8	21.6
Color Mix	64.0	6.0	18.0	16.0	8.0	26.0	10.0
Lifespan, Longest	90.0	90.0	70.0	50.0	70.0	75.0	15.0
Lifespan, Shortest	90.0	60.0	80.0	45.0	50.0	50.0	10.0
Life Stage, Plant	59.6	15.0	4.2	28.2	0.0	7.4	1.6
Life Stage, Animal	89.5	11.0	32.5	21.5	16.3	12.0	3.0
Force	80.0	61.0	62.0	25.0	1.0	21.0	21.0
Friction	66.0	26.0	82.0	26.0	4.0	5.0	1.0
Genetics, Know	67.2	14.4	66.8	83.4	23.4	30.4	0.4
Genetics, Unknow	49.4	81.6	61.6	29.0	25.0	31.6	21.2
Average	54.5	30.7	41.6	26.4	13.1	20.7	10.5

Table C.3: **Performance of LLMs with "SSO" Across Tasks.**

doi:10.6342/NTU202404517

	LLM model						
Task	GPT-4o mini	Gemma-2 9B	Llama-3 8B	Phi-3 small	Mistral 7B	Phi-3 mini	Gemma-2 2B
Boil	39.0	8.6	0.0	16.2	0.0	1.4	14.8
Freeze	61.2	0.0	0.0	21.2	0.0	4.0	2.0
Use Thermometer	100.0	82.4	90.4	64.0	95.2	29.0	28.2
Boiling Point	80.2	51.4	52.2	63.6	52.4	58.6	48.0
Find Living	58.6	45.2	43.6	73.6	66.8	26.8	38.6
Find Plant	86.8	40.2	50.2	58.6	70.2	15.2	41.8
Grow Plant	63.0	47.8	47.8	48.2	47.8	43.0	42.8
Grow Fruit	31.6	67.8	67.8	42.2	67.8	27.0	41.2
Chemistry	85.0	36.8	35.2	60.0	38.4	26.6	36.8
Color Mix	72.0	54.0	54.0	74.0	36.0	42.0	18.0
Lifespan, Longest	90.0	70.0	70.0	60.0	80.0	60.0	40.0
Lifespan, Shortest	70.0	40.0	40.0	40.0	40.0	60.0	40.0
Life Stage, Plant	82.0	70.6	93.0	58.6	87.0	44.4	54.4
Life Stage, Animal	100.0	91.5	100.0	100.0	91.5	75.5	51.5
Force	82.0	16.0	32.0	32.0	10.0	42.0	19.0
Friction	26.0	13.0	16.0	64.0	13.0	24.0	9.0
Genetics, Know	68.0	0.0	0.0	100.0	0.0	4.0	3.6
Genetics, Unknow	69.4	21.4	18.0	81.6	1.4	3.0	8.0
Average	70.3	42.0	45.0	58.8	44.3	32.6	29.9

Table C.4: **Performance of LLMs with "MENSA" Across Tasks.**

doi:10.6342/NTU202404517

Appendix D — Computing Infrastructure Specifications

We use the following hardware to run all of our experiments.

- CPU: Intel Core i5-12400
- GPU: NVIDIA GeForce RTX 3090
- Amount of memory: 16 GB
- Operating system: Ubuntu 22.04 LTS

We use the following software:

- Simulator: ScienceWorld 1.2.0 on ”*exhaustivevalidaction*” branch[1]
- Framework: PyTorch 2.3.1
- Local LLM API: vLLM v0.5.2[2]

[1] https://github.com/allenai/ScienceWorld/tree/exhaustivevalidactions
[2] https://github.com/vllm-project/vllm

doi:10.6342/NTU202404517