ODYSSEY: Empowering Minecraft Agents with Open-World Skills

Shunyu Liu[1], **Yaoru Li**[1], **Kongcheng Zhang**[1], **Zhenyu Cui**[1], **Wenkai Fang**[1], **Yuxuan Zheng**[1], **Tongya Zheng**[2,3] and **Mingli Song**[3,4*]
[1]Zhejiang University
[2]Zhejiang Provincial Engineering Research Center for Real-Time SmartTech in Urban Security Governance, School of Computer and Computing Science, Hangzhou City University
[3]State Key Laboratory of Blockchain and Data Security, Zhejiang University
[4]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security
{liushunyu, liyaoru, zhangkc, zhenyucui, wenkfang, zyxuan}@zju.edu.cn, doujiang_zheng@163.com, brooksong@zju.edu.cn

Abstract

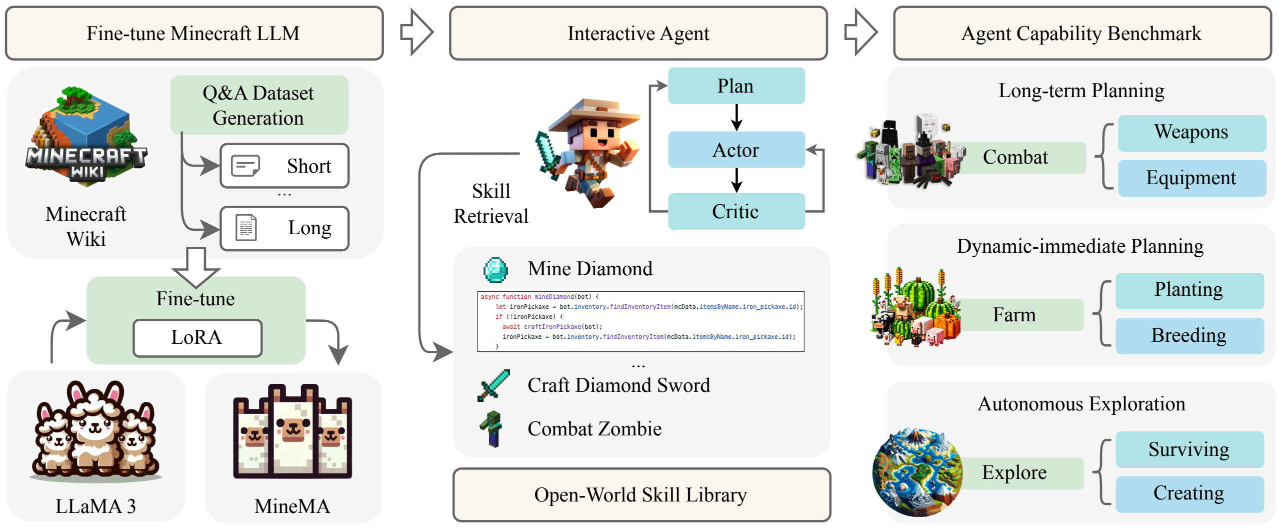
Recent studies have delved into constructing generalist agents for open-world environments like Minecraft. Despite the encouraging results, existing efforts mainly focus on solving basic programmatic tasks, *e.g.*, material collection and tool-crafting following the Minecraft tech-tree, treating the `ObtainDiamond` task as the ultimate goal. This limitation stems from the narrowly defined set of actions available to agents, requiring them to learn effective long-horizon strategies from scratch. Consequently, discovering diverse gameplay opportunities in the open world becomes challenging. In this work, we introduce ODYSSEY, a new framework that empowers Large Language Model (LLM)-based agents with open-world skills to explore the vast Minecraft world. ODYSSEY comprises three key parts: (1) An interactive agent with an *open-world skill library* that consists of 40 primitive skills and 183 compositional skills. (2) A fine-tuned LLaMA-3 model trained on a *large question-answering dataset* with 390k+ instruction entries derived from the Minecraft Wiki. (3) A *new agent capability benchmark* includes the long-term planning task, the dynamic-immediate planning task, and the autonomous exploration task. Extensive experiments demonstrate that the proposed ODYSSEY framework can effectively evaluate different capabilities of LLM-based agents. All datasets, model weights, and code are publicly available to motivate future research on more advanced autonomous agent solutions.

1 Introduction

Developing autonomous agents capable of performing open-world tasks represents a significant milestone towards achieving artificial general intelligence [Reed *et al.*, 2022; Driess *et al.*, 2023]. These open-world tasks necessitate that agents interact with complex and dynamic environments, make decisions based on incomplete information, and adapt to unexpected events. Early reinforcement learning agents [Tessler *et al.*, 2017; Oh *et al.*, 2017] have demonstrated limited knowledge in such open-world setting. Furthermore, these agents often struggle with long-term planning, which is crucial for the fulfillment of intricate goals. Recent breakthrough of Large Language Models (LLMs) [Hu *et al.*, 2021; Achiam *et al.*, 2023; Touvron *et al.*, 2023] have shown the potential to revolutionize various fields such as healthcare [Zhang *et al.*, 2023b; Yang *et al.*, 2024b], robotics [Huang *et al.*, 2022; Singh *et al.*, 2023], and web services [Deng *et al.*, 2023; Iong *et al.*, 2024], attributed to its capability on endowing agents with expansive knowledge and sophisticated planning akin to human reasoning [Wei *et al.*, 2022; Wang *et al.*, 2024a; Liang *et al.*, 2023]. However, the development of LLMs in open-world tasks remains challenging due to the need for well-defined environments and measurable benchmarks [Wang *et al.*, 2023a; Qin *et al.*, 2023].

The popular Minecraft game features a vast and diverse world with various biomes, terrains, and resources, making it an ideal testbed for evaluating the capabilities of autonomous agents in the open-world setting. To facilitate the development of generalist agents in this setting, MineRL [Guss *et al.*, 2019] and MineDojo [Fan *et al.*, 2022] introduced simulation benchmarks built upon the sandbox Minecraft environment. The seminal work, Voyager [Wang *et al.*, 2023a], proposed an LLM-based agent to drive exploration in Minecraft. Subsequently, there has been a surge of efforts to leverage the superior performance of LLMs to extend the capabilities of such Minecraft agents [Wang *et al.*, 2023b; Zhou *et al.*, 2024; Wang *et al.*, 2023c; Qin *et al.*, 2023]. Despite recent advancements, existing works mainly focus on solving basic programmatic tasks, often considering the `ObtainDiamond` task as the ultimate challenge [Guss *et al.*, 2019]. Basic programmatic tasks refer to those constrained by the explicit dependencies following the Minecraft tech-tree, such as collecting materials and crafting tools. Such tasks inherently only assess the ability of LLMs to prioritize crafting steps within a limited task space, rather than their potential for complicated and diverse solutions. This limitation arises from

*Corresponding author.



```
async function mineDiamond(bot) {
    let ironPickaxe = bot.inventory.findInventoryItem(mcData.itemsByName.iron_pickaxe.id);
    if (!ironPickaxe) {
      await craftIronPickaxe(bot);
      ironPickaxe = bot.inventory.findInventoryItem(mcData.itemsByName.iron_pickaxe.id);
    }
```

Figure 1: An overview of the proposed ODYSSEY framework. Odyssey consists of three key components: (1) a fine-tuned LLaMA-3 model trained on a large-scale question-answering dataset; (2) an interactive agent equipped with an extensive open-world skill library; (3) a novel agent capability benchmark encompassing a variety of tasks.

the narrowly defined set of actions available to agents (*e.g.*, mouse and keyboard), which necessitates learning skills from scratch. Since Minecraft is fundamentally resource-based, an agent must first learn to collect adequate resources and tools to engage in creative play, which limits the exploration of diverse gameplay options. Moreover, methods like Voyager [Wang *et al.*, 2023a] heavily rely on the powerful GPT-4 for high-quality solutions, imposing a substantial cost burden on researchers who prefer open-source models.

In this work, we introduce ODYSSEY[1], a novel framework that equips LLM-based agents with advanced open-world skills, enabling efficient interaction and exploration within the Minecraft environment. ODYSSEY allows agents to move beyond basic programmatic tasks and focus more on complex open-world challenges. As shown in Fig. 1, ODYSSEY comprises three key contributions:

1. We develop an LLM-based interactive agent with an *open-world skill library*, encompassing 40 primitive skills that serve as underlying interfaces and 183 compositional skills tailored for complex and diverse tasks in an open-world setting. A recursive method improves skill execution by checking prerequisites. The ODYSSEY agent consists of a planner for goal decomposition, an actor for skill retrieval and subgoal execution, and a critic for feedback and strategy refinement.
2. We fine-tune the LLaMA-3 model [Touvron *et al.*, 2023] for Minecraft agents using a *comprehensive question-answering dataset*. This involves generating a large-scale training dataset with 390k+ instruction entries from Minecraft Wikis, fine-tuning various sizes of the LLaMA-3 models using LoRA [Hu *et al.*, 2021], and evaluating them with a custom multiple-choice dataset.
3. We introduce a *new agent capability benchmark* to evaluate different aspects of agent performance in Minecraft, including the long-term planning task, the dynamic-immediate planning task, and the autonomous exploration task. Extensive experiments demonstrate that the proposed ODYSSEY framework provides a robust measure of agent effectiveness, showcasing the practical advantages of our framework using the open-source models.

It is worth noting that our focus is *not to design a new LLM-based agent architecture*. Instead, this work aims to provide a comprehensive framework for *developing and evaluating autonomous agents in open-world environments*, enabling them to explore the vast and diverse Minecraft world.

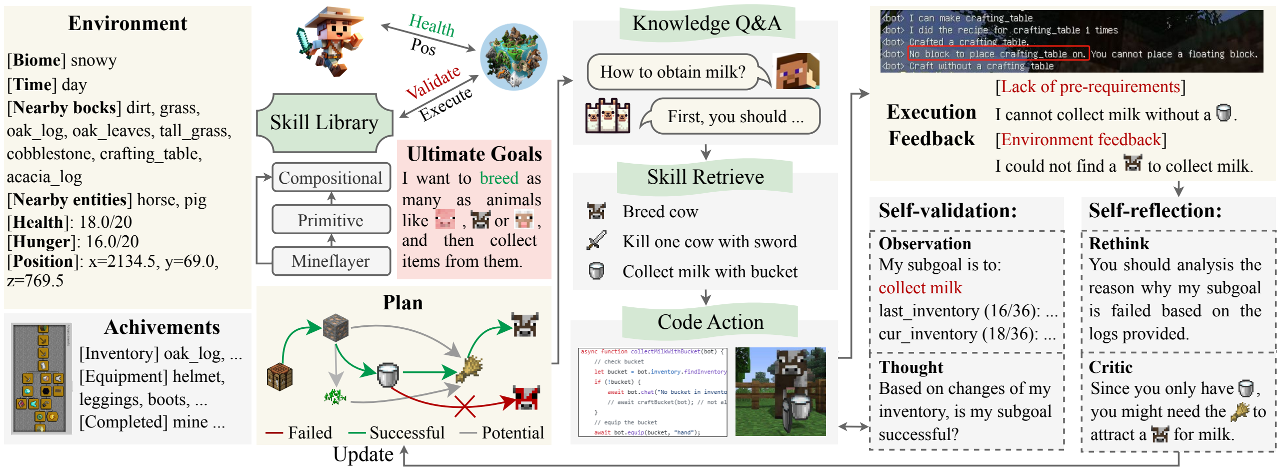
2 Open-World Skill-based Interactive Agent

ODYSSEY develops an LLM-based interactive agent with an open-world skill library, aiming to enhance the efficiency and adaptability of agents in complex Minecraft environments. The skill library comprises 40 primitive skills and 183 compositional skills, while the LLM-based agent employs a planner-actor-critic architecture to facilitate task decomposition, skill execution, and performance feedback. The architecture of the interactive agent is depicted in Fig. 2. Full skill and prompt details used in the LLM-based interactive agent are given in Appendix C.

2.1 Open-World Skill Library

Primitive skills. ODYSSEY encompass a series of underlying interfaces on top of Mineflayer JavaScript APIs [PrismarineJS, 2023], divided into two main categories: 32 operational skills and 8 spatial skills. This suite of skills exceeds the 18 primitive skills (all are operational skills) delineated in Voyager [Wang *et al.*, 2023a]. Operational skills serve as foundational interfaces with parameterized input, such as `mine(·)` for material collection and `craft(·)` for

[1]The Odyssey is a great ancient Greek epic poem attributed to Homer, which is now often used metaphorically to describe *a long adventurous journey* (Oxford English Dictionary).



[**Biome**] snowy
[**Time**] day
[**Nearby bocks**] dirt, grass, oak_log, oak_leaves, tall_grass, cobblestone, crafting_table, acacia_log
[**Nearby entities**] horse, pig
[**Health**]: 18.0/20
[**Hunger**]: 16.0/20
[**Position**]: x=2134.5, y=69.0, z=769.5

I want to breed as many as animals like , or , and then collect items from them.

[Inventory] oak_log, ...
[Equipment] helmet, leggings, boots, ...
[Completed] mine ...

```
async function collectMilkWithBucket(bot) {
    // check bucket
    let bucket = bot.inventory.findInventory
    if (!bucket) {
        await bot.chat("No bucket in invento
        // await craftBucket(bot); // not al
    }
    // equip the bucket
    await bot.equip(bucket, "hand");
```

```
<bot> I can make crafting_table
<bot> I did the recipe for crafting_table 1 times
<bot> Crafted a crafting table.
<bot> No block to place crafting_table on. You cannot place a floating block.
<bot> Craft without a crafting table
```

[Lack of pre-requirements]
I cannot collect milk without a .
[Environment feedback]
I could not find a to collect milk.

Observation
My subgoal is to: collect milk
last_inventory (16/36): ...
cur_inventory (18/36): ...

Thought
Based on changes of my inventory, is my subgoal successful?

Rethink
You should analysis the reason why my subgoal is failed based on the logs provided.

Critic
Since you only have , you might need the to attract a for milk.

Figure 2: An illustrative diagram of the interactive agent following a planner-actor-critic architecture based on the skill library. The Planner decomposes ultimate goals into specific subgoals, while the Actor sequentially executes code actions for each subgoal using the skill library. The Critic evaluates these actions through self-validation and reflection, enabling the agent to update its plan based on execution feedback.

tool crafting. Additionally, we pioneer 8 spatial skills that Voyager [Wang *et al.*, 2023a] lacks, allowing for environmental interactions based on the agent coordinates. Given that our work is conducted within a text-based Minecraft environment [Wang *et al.*, 2023a; Fan *et al.*, 2022], spatial skills are crucial for handling tasks that require precise positioning and orientation, especially in the absence of visual input.

Compositional skills. We encapsulate primitive skills into higher-level ones, functioning to address a variety of basic programmatic tasks, such as `mineDiamond` and `craftIronPickaxe`. ODYSSEY classifies 183 compositional skills into types like `mineX`, `craftX`, `plantX`, `breedX`, `cookX`, *etc.* We use a recursive method to construct the skill library, simplifying complex task decomposition by ensuring prerequisites are met before skill execution. Taking `mineDiamond` as an example, if the agent lacks an iron pickaxe, it will recursively execute `craftIronPickaxe`. This indicates that our program internally manages the construction and execution order of skills through its recursive method, thereby avoiding the need for the agent to engage in additional planning.

To facilitate efficient retrieval of skills in the skill library, we first generate a description for each skill by calling the LLM and using the complete program code as a prompt. We then employ Sentence Transformer [Reimers and Gurevych, 2019] to encode the skill description. This method transforms text information into vector representations, facilitating semantic retrieval and enabling the agent to find the most relevant skill description based on the context provided.

2.2 Planner-Actor-Critic Architecture

LLM Planner. The LLM Planner is responsible for developing a comprehensive plan, facilitating efficient exploration through long-term goal decomposition. The LLM Planner breaks down high-level goals into specific low-level subgoals, each corresponding to a particular skill outlined in Sec. 2.1. By addressing each subgoal in the plan, the ultimate goal can be progressively achieved. The input prompt to the planner consists of several components: **(1) Ultimate goals and behavioral constraints.** For example, "My ultimate goal is to ... Propose the current task only when you ensure that you have all the necessary dependent items in inventory". **(2) States of the agent.** This reflects the interaction between the agent and environment, such as hunger and health values, position and nearby entities, *etc.* **(3) Achievements of the agent.** This includes the current inventory and unlocked equipment, as well as previously successful and failed tasks.

LLM Actor. In the execution phase, the LLM actor is invoked to sequentially execute the subgoals generated by the LLM planner within the Minecraft environment. This process utilizes the open-world skill library to achieve these subgoals. The mapping from high-level subgoals to executable skill code is accomplished through query context encoding and skill similarity retrieval. This process includes: **(1) Query context.** The text-based subgoals generated by the LLM planner are encoded by Sentence Transformer [Reimers and Gurevych, 2019] to vector representations as the query context. **(2) Similarity matching.** The vector similarity between the query context and the skill descriptions in the skill library is computed to determine semantic closeness. **(3) Skill selection.** The top-5 relevant skills with the highest scores are identified, and the actor selects the most appropriate code for execution within the environment based on their descriptions.

LLM Critic. During action execution, it is critical for an agent to document its experiences, especially noting successful outcomes and failure points. This is crucial in open-world planning to establish a feedback-informed system, which corrects initial plan discrepancies that can cause execution errors. For instance, achieving the animal breeding goal requires prerequisite crops for feed. The LLM critic can assess action effectiveness by comparing expected and actual outcomes, providing insights for refining future strategies. We categorize feedback into three types: **(1) Execution feedback.** This captures the progress of skill execution. For example, "No hoe in inventory. Craft a hoe first!" not only highlights the reason for failure in hoeing farmland but also provides a guideline to address this problem. **(2) Self-validation.** By presenting inventory changes post-action to the LLM critic, we empower it to validate whether the skill has achieved its subgoal, eliminating the need for manual checks. **(3) Self-reflection.** Simply confirming the completion of a subgoal is often inadequate for correcting planning errors. The LLM critic also serves as an analyst, deducing the cause of task failure by evaluating the current state of the agent and its environment. It then offers a critique, suggesting a more efficient strategy for task completion.

3 Fine-tune Minecraft LLM

To improve agent performance in Minecraft, we fine-tune the LLaMA-3 model [Touvron *et al.*, 2023] using a large-scale Question-Answering (Q&A) dataset with 390k+ instruction entries sourced from the Minecraft Wiki. ODYSSEY presents an effective procedure for converting a foundation model into a domain-specific model, which involves dataset generation, model fine-tuning, and model evaluation. The detailed descriptions can be found in Appendix D.

Dataset Generation. We develop a GPT-assisted method to generate a Minecraft instruction dataset. First, we crawl relevant content from the Minecraft Wiki, excluding non-essential sections like history. The collected data is then categorized and separated into different files based on their content type. Then we use GPT-3.5-Turbo [OpenAI, 2023] with different customized prompts to generate diverse Q&A pairs. These Q&A pairs are categorized into four types based on the nature of the answers: short, normal, long, and boolean, yielding 390k+ entries. In contrast, the Wiki dataset released by MineDojo [Fan *et al.*, 2022] only collects Minecraft Wiki pages, without refining the content and generating Q&A pairs for model training. STEVE [Zhao *et al.*, 2023] introduces a non-public dataset with 20k+ Q&A pairs, which is smaller than our dataset in terms of scale and diversity.

Model Fine-tuning. We employ LoRA [Hu *et al.*, 2021] for model fine-tuning, which is a parameter-efficient training technique. LoRA introduces small, trainable low-rank matrices to adapt a pre-trained neural network, enabling targeted updates without the need to retrain the entire model. Using LoRA, we fine-tune the LLaMA-3-8B-Instruct and LLaMA-3-70B-Instruct models with our Minecraft dataset, resulting in the new models termed MineMA-8B and MineMA-70B.

Model Evaluation. In Minecraft, questions are often open-ended and can yield diverse answers; therefore, conventional evaluation metrics [Papineni *et al.*, 2002; Lin, 2004] may fall short. Meanwhile, common benchmarks [Wang *et al.*, 2018; Wang *et al.*, 2019; Hendrycks *et al.*, 2021] are not suitable for assessing the capabilities of expert models. Thus, we employed GPT-4 [Achiam *et al.*, 2023] to generate two Multiple-Choice Question (MCQ) datasets based on different themes and keywords related to Minecraft. These datasets can quantitatively evaluate the domain-specific expertise of models.

4 Agent Capability Benchmark

ODYSSEY presents a new benchmark for evaluating agent capabilities within Minecraft, offering three task types: long-term planning, dynamic-immediate planning, and autonomous exploration. It is notable that these tasks cannot be solved by any single skill but demand a sophisticated combination of multiple skills. These tasks are set in various Minecraft scenarios, with different tasks in the same scenario testing different agent capabilities. For example, in the cooking scenario, long-term planning requires formulating a complete plan to locate and hunt a specific animal, whereas dynamic-immediate planning involves selecting which nearby animal to cook based on the immediate environment. Please refer to Appendix E for more details.

Long-term Planning Task. We design a suite of combat scenarios to assess the long-term planning capability of agents, requiring them to craft appropriate weapons and equipment to defeat various monsters. These combat scenarios can be divided into single-type and multi-type monster scenarios. Agents must generate a comprehensive long-term plan, detailing the sequence of crafting the necessary weapons and equipment for the assigned combat task. Performance is measured by remaining health and time consumed during combat. After each battle, agents can iteratively optimize their plan, learning from previous outcomes to improve performance in subsequent rounds. To extend the scope of the long-term planning task beyond combat, we also adopt animal husbandry and cooking scenarios, where agents are required to formulate detailed plans for completing tasks related to specific animals.

Dynamic-immediate Planning Task. This task requires agents to dynamically generate and execute plans based on immediate environmental feedback. Thus, we design a suite of farming scenarios, where agents engage in activities like planting, cooking, and animal husbandry. Although some scenarios are similar to the long-term planning task, the dynamic-immediate planning task emphasizes reacting to real-time feedback like available resources and nearby animals. Performance is evaluated through task completion time and success rates.

Autonomous Exploration Task. To further test the exploratory capability of agents within open-world settings, we design an autonomous exploration task. In this task, agents are required to determine their subsequent objectives and execute the appropriate skills based on the game context. This task involves discovering and utilizing resources, while adapting to unexpected events such as encounters with hostile monsters. Agents must adapt to these challenges by developing strategies for resource management and task prioritization. The performance metrics include the number of distinct items obtained, the total items crafted, the recipes and advancements (R&A) unlocked, and the distance traveled.

5 Experiments

To demonstrate the effectiveness of the proposed ODYSSEY framework, we conduct experiments on basic programmatic

Task	Time (min)	2min	5min	10min	15min
🟫	0.59 ± 0.79	95.8%	99.2%	100.0%	100.0%
⛏	0.95 ± 0.80	92.5%	99.2%	100.0%	100.0%
⛏	1.48 ± 0.96	85.0%	97.5%	100.0%	100.0%
⛏	4.43 ± 1.48	0.0%	76.7%	100.0%	100.0%
💎	6.48 ± 2.02	0.0%	21.7%	92.5%	100.0%

Table 1: Average execution time and success rate in different time on 5 basic programmatic tasks in Minecraft: 🟫 Crafting Table, ⛏ Wooden Tool, ⛏ Stone Tool, ⛏ Iron Tool, and 💎 Obtain Diamond.

tasks and the agent capability benchmark. Our simulation environment is built on top of Voyager [Wang *et al.*, 2023a], providing a text-based interface for agents to interact with Minecraft. We only use GPT for initial data generation, but all experiments are conducted with the open-source LLaMA-3 model, significantly reducing costs compared to GPT-4-based skill generation methods [Wang *et al.*, 2023a; Wang *et al.*, 2023b]. Notably, we do not employ GPT-4 in Voyager due to the high cost, which we estimate would be in the thousands of dollars per experiment. Instead, we reproduce Voyager using GPT-4o-mini and GPT-3.5 for comparison. More details are provided in Appendix F. We aim to answer the following questions: (1) Can the open-world skill library improve the efficiency of agents in Minecraft? (Sec. 5.1). (2) How well do agents with different LLMs perform on the agent capability benchmark tasks? (Sec. 5.2). (3) What is the contribution of different components of the ODYSSEY agent to its overall performance? (Sec. 5.3).

5.1 Open-World Skill Library

To demonstrate the superior capability of our open-world skill library in Minecraft, we first tested it on 5 basic programmatic tasks. We conducted 120 repeated experiments on each task and recorded the average completion time for each task as well as the success rates at different time points. The results in Table 1 demonstrate that our open-world skill library efficiently handles basic programmatic tasks. Simple tasks achieve near-perfect success within five minutes. Even for difficult tasks like obtaining a diamond, success rates rise from 21.7% at five minutes to 92.5% at ten minutes, highlighting the effectiveness of the skill library.

5.2 Agent Capability Benchmark

We evaluate the LLM-based agent on the long-term planning task, the dynamic-immediate planning task, and the autonomous exploration task from the ODYSSEY benchmark. These tasks cover a variety of complex gaming scenarios and require diverse solutions.

Long-term Planning Task. The long-term planning task assesses the agent capability to directly formulate and execute comprehensive plans over extended periods. For example, in the combat scenarios, the agent is required to plan a list of weapons and equipment to craft based on the strength of different monsters, with the goal of defeating the monster in as short a time as possible. We compared the performance of our agent with both the fine-tuned MineMA-8B and the original LLaMA-3-8B, and also the performance of Voyager [Wang *et al.*, 2023a] with GPT-4o-mini across these tasks. Moreover, we also evaluate the performance of single-round and multi-round planning. The single-round test results in Tab. 2 demonstrate that the fine-tuned MineMA-8B surpasses the original LLaMA-3-8B in terms of success rate and time efficiency, albeit at the cost of more LLM iterations. Moreover, our agent with MineMA-8B can outperform Voyager

Task	Model	SR	Health	Time	Iters
1 zombie	Voyager	**3 / 3**	**20.0**	9.9	67.3
	LLaMA-3-8B	4 / 8	**20.0**	**8.3**	**6.1**
	MineMA-8B	**8 / 8**	19.4	*8.8*	*10.0*
1 spider	Voyager	**3 / 3**	10.8	*9.4*	19.0
	LLaMA-3-8B	4 / 8	**19.4**	12.1	**8.4**
	MineMA-8B	**8 / 8**	*19.3*	**8.3**	*15.2*
1 skeleton	Voyager	*2 / 3*	*16.5*	**7.4**	46.0
	LLaMA-3-8B	4 / 8	**17.6**	*8.1*	**8.9**
	MineMA-8B	**8 / 8**	13.6	8.6	*12.1*
1 zombified piglin	Voyager	**3 / 3**	*19.0*	14.5	50.3
	LLaMA-3-8B	4 / 8	**19.9**	*9.2*	**10.0**
	MineMA-8B	**8 / 8**	18.7	**8.5**	*11.7*
1 enderman	Voyager	**2 / 3**	11.0	22.8	28.0
	LLaMA-3-8B	2 / 8	*15.1*	*13.0*	**6.8**
	MineMA-8B	*4 / 8*	**19.8**	**10.4**	*12.5*
1 zombie villager	Voyager	2 / 3	**20.0**	*12.6*	50.0
	LLaMA-3-8B	*7 / 8*	19.6	12.7	**11.0**
	MineMA-8B	**8 / 8**	**20.0**	**9.0**	*12.8*
1 cave spider	Voyager	2 / 3	16.5	*10.0*	79.2
	LLaMA-3-8B	*6 / 8*	*19.5*	12.0	*19.5*
	MineMA-8B	**7 / 8**	**20.0**	**3.6**	**8.6**
1 wither skeleton	Voyager	1 / 3	**20.0**	20.9	100.0
	LLaMA-3-8B	*6 / 8*	13.2	*11.7*	**12.3**
	MineMA-8B	**7 / 8**	*17.3*	**11.0**	*12.6*
1 zombie, 1 spider	Voyager	*1 / 3*	*17.5*	**5.9**	21.0
	LLaMA-3-8B	1 / 8	**20.0**	*8.5*	**6.0**
	MineMA-8B	**5 / 8**	16.4	10.6	*12.0*
1 zombie, 1 skeleton	Voyager	**2 / 3**	**19.0**	15.0	40.5
	LLaMA-3-8B	1 / 8	0.2	**13.5**	**9.0**
	MineMA-8B	*3 / 8*	*12.8*	*14.0*	*10.3*
3 zombies	Voyager	**2 / 3**	**7.8**	**8.2**	61.0
	LLaMA-3-8B	*1 / 8*	3.7	14.3	**8.0**
	MineMA-8B	*1 / 8*	*5.2*	*11.1*	*14.0*
cook meat	Voyager	0 / 3	-	N/A	N/A
	LLaMA-3-8B	*1 / 8*	-	**20.3**	**19.0**
	MineMA-8B	**2 / 8**	-	*21.4*	*30.0*
animal husbandry	Voyager	*1 / 3*	-	19.0	**12.0**
	LLaMA-3-8B	2 / 8	-	**15.3**	31.0
	MineMA-8B	**3 / 8**	-	*16.8*	*26.7*

Table 2: Performance comparison of different models on the single-round long-term planning task. "SR" refers to success rate. "Health" refers to the remaining health points. "Time" refers to the minutes spent in both gathering materials and crafting equipment to defeat different monsters. "Iters" is the number of LLM iterations (calling LLM) required to complete the task. All metrics are calculated only for successful tasks. **Bold** and *italics* mean the best and the second-best results. "-" indicates that health is not a relevant metric. "N/A" indicates that all tasks fail. Please refer to Appendix F.4 for standard deviation and visual inspection.

Task	Model	SR	Time	Iters
Collect Seeds	GPT-4o	5 / 5	1.2	1.0
	Baichuan2-7B	2 / 5	1.8	3.0
	Qwen2-7B	2 / 5	3.8	4.5
	MineMA-8B	**5 / 5**	**1.3**	*1.4*
	MineMA-70B	**5 / 5**	*1.4*	**1.0**
Hoe Farmland	GPT-4o	5 / 5	3.9	5.8
	Baichuan2-7B	0 / 5	N/A	N/A
	Qwen2-7B	*2 / 5*	*15.7*	*19.5*
	MineMA-8B	*2 / 5*	17.2	26.5
	MineMA-70B	**4 / 5**	**10.2**	**11.8**
Shear Sheep	GPT-4o	5 / 5	4.7	5.6
	Baichuan2-7B	1 / 5	26.0	30.0
	Qwen2-7B	*2 / 5*	*11.0*	**10.8**
	MineMA-8B	*2 / 5*	15.7	13.0
	MineMA-70B	**3 / 5**	**6.9**	*11.0*
Milk Cow	GPT-4o	3 / 5	17.9	20.3
	Baichuan2-7B	0 / 5	N/A	N/A
	Qwen2-7B	*1 / 5*	26.1	30.0
	MineMA-8B	*1 / 5*	**7.2**	**7.0**
	MineMA-70B	**2 / 5**	*8.6*	*10.0*
Cook Meat	GPT-4o	3 / 5	5.5	5.0
	Baichuan2-7B	0 / 5	N/A	N/A
	Qwen2-7B	0 / 5	N/A	N/A
	MineMA-8B	*1 / 5*	*25.6*	*38.0*
	MineMA-70B	**2 / 5**	**20.2**	**24.0**
Obtain Leather	GPT-4o	5 / 5	14.8	13.0
	Baichuan2-7B	0 / 5	N/A	N/A
	Qwen2-7B	1 / 5	*14.9*	*16.0*
	MineMA-8B	*4 / 5*	15.0	17.8
	MineMA-70B	**5 / 5**	**7.4**	**8.8**
Make Sugar	GPT-4o	5 / 5	5.5	7.0
	Baichuan2-7B	2 / 5	16.2	22.0
	Qwen2-7B	2 / 5	15.4	15.5
	MineMA-8B	**5 / 5**	**4.3**	**7.0**
	MineMA-70B	**5 / 5**	*4.3*	*7.8*
Collect Water	GPT-4o	5 / 5	11.4	27.3
	Baichuan2-7B	0 / 5	N/A	N/A
	Qwen2-7B	1 / 5	*10.0*	10.0
	MineMA-8B	*4 / 5*	10.4	**8.8**
	MineMA-70B	**5 / 5**	**9.3**	*9.4*

Table 3: Performance comparison of different models on the dynamic-immediate planning task. All evaluation metrics are calculated only for successful tasks. **Bold** and *italics* mean the best and the second-best results of all open-sourced LLMs (excluding GPT-4o). "N/A" indicates that all tasks fail. Please refer to Appendix F.4 for standard deviation and visual inspection.

with GPT-4o-mini in most scenarios, indicating the effectiveness of our fine-tuning strategy. The multi-round test results in Fig. 3 show that the multi-round planning strategy significantly improves time efficiency. This improvement suggests that the agent is capable of iteratively refining its plans based on the outcomes of previous encounters, thereby boosting its performance in subsequent rounds.

Dynamic-immediate Planning Task. For the dynamic-immediate planning task, the agent is required to dynamically generate and execute plans based on immediate environmental feedback. We compared our MineMA model with different open-sourced LLMs, including GPT-4o, Qwen2-7B [Yang *et al.*, 2024a] and Baichuan2-7B [Yang *et al.*, 2023]. Moreover, we evaluate the performance of the MineMA-8B and the MineMA-70B model to investigate the impact of model size on task performance. As shown in Tab. 3, the MineMA-8B model outperforms the Baichuan2-7B and Qwen2-7B models in terms of success rate and time efficiency. Moreover, the MineMA-70B model shows superior performance compared with the MineMA-8B model. Across all open-sourced LLMs, MineMA-70B demonstrates higher success rates and generally lower average execution times and LLM iterations. Additionally, our MineMA can achieve performance similar to that of GPT-4o.

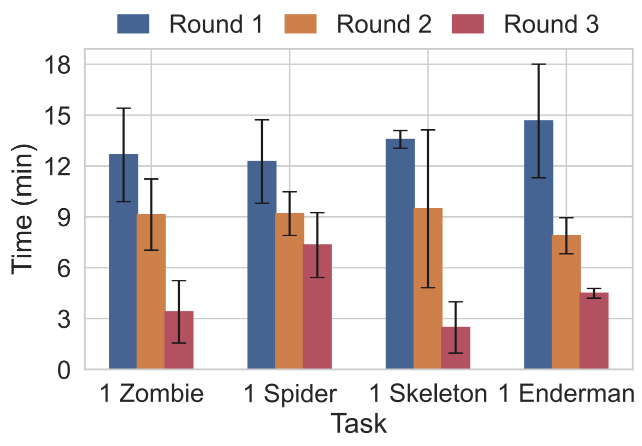
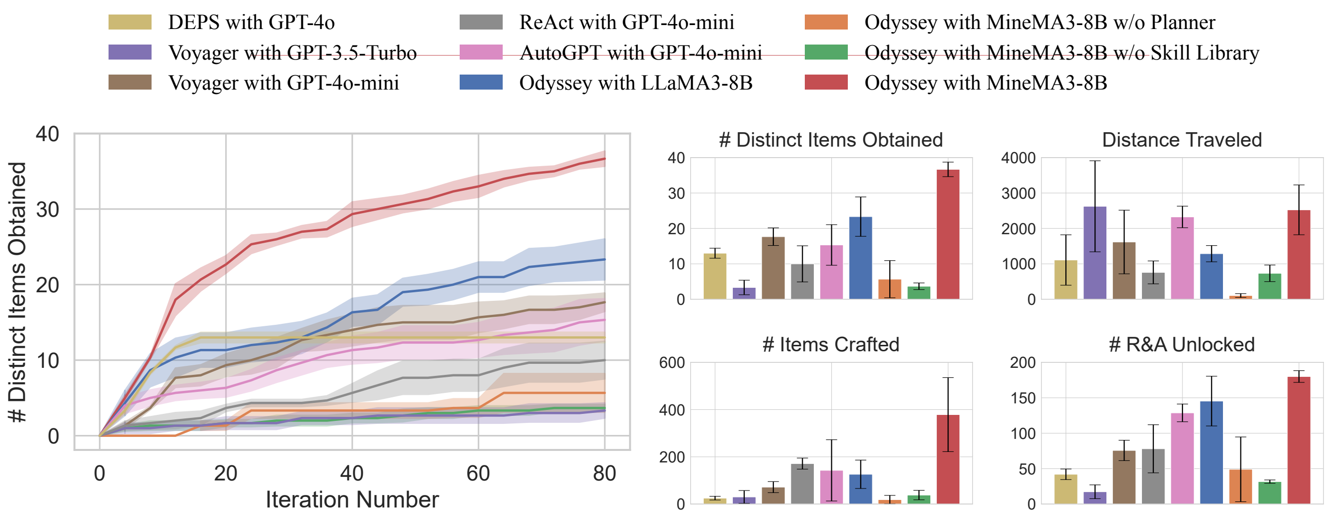


Figure 3: Performance on the multi-round long-term planning task. Note that all data are from successful tasks.

Autonomous Exploration Task. In the autonomous exploration task, the agent is required to explore the Minecraft world freely without any specific goals. We compare our agent with different Minecraft agents (Voyager [Wang *et al.*, 2023a] and DEPS [Wang *et al.*, 2023b]) and different LLM-based agent techniques (ReAct [Yao *et al.*, 2023] and AutoGPT [Significant-Gravitas, 2023]) on this task. Note that we reproduced different LLM-based agent techniques following the same settings as in Voyager [Wang *et al.*, 2023a]. As shown in Fig. 4, our agent with the MineMA-8B model can achieve superior performance compared with all baselines, indicating that the agent can autonomously explore the Minecraft world without specific goals. It is notable that our agent with the MineMA-8B model can outperform Voyager [Wang *et al.*, 2023a] with GPT-4o-mini or GPT-3.5.

5.3 Ablation Study

We conduct ablation studies on two core components of the ODYSSEY agent, including the LLM planner and the open-world skill library. The results are shown in Fig. 4. In the autonomous exploration task, the LLM planner is responsible for generating a comprehensive plan based on the open-world skill library. The ablation study demonstrates that the planner is indispensable for the agent to effectively navigate the complex Minecraft environment. Additionally, our experimental results indicate that the absence of the open-world skill library significantly degrades performance. Without the open-world skill library, the 8B LLM model alone is largely



(a) Exploration Curves

(b) Evaluation Metrics

Figure 4: Performance comparison of different models on autonomous exploration tasks. To make the results in figures clearer for readers, we adopt a 50% confidence interval to plot the error region.

incapable of generating executable codes for the agent. This underscores the critical role of the open-world skill library in enabling the agent to perform complex tasks within the open-world setting of Minecraft.

6 Related Works

Minecraft Agents. Minecraft agents have been widely studied in recent years to test the capabilities of autonomous agents in open-world environments. Previous works focused on training Minecraft agents with reinforcement learning [Tessler *et al.*, 2017; Oh *et al.*, 2017; Lin *et al.*, 2022; Mao *et al.*, 2022; Hafner *et al.*, 2023] or imitation learning [Baker *et al.*, 2022; Cai *et al.*, 2023; Lifshitz *et al.*, 2023], which are extensively used in the MineRL [Guss *et al.*, 2019] competition to solve the `ObtainDiamond` task. With the rapid development of LLMs, numerous studies leverage LLMs to enhance agent capabilities [Zhang *et al.*, 2023a; Zhu *et al.*, 2023; Feng *et al.*, 2023; Zhao *et al.*, 2023; Zheng *et al.*, 2023; Zhou *et al.*, 2024; Li *et al.*, 2024; Yu and Lu, 2024; Wang *et al.*, 2024b; Cai *et al.*, 2024]. Among these, several works [Li *et al.*, 2023; Yuan *et al.*, 2023; Wang *et al.*, 2023c; Qin *et al.*, 2023; Ding *et al.*, 2023] employ LLMs to guide skill learning in Minecraft, enabling agents to act in a human-like way. However, these methods mainly focus on learning primitive skills from scratch, lacking a reusable skill library. Voyager [Wang *et al.*, 2023a] builds a skill library by allowing the LLM to write its own skills. However, Voyager must rely on GPT-4 for high-quality skill generation, incurring substantial costs. This expense can be prohibitive for many researchers. In contrast, ODYSSEY provides an open-world skill library that agents can call upon, achieving performance comparable to Voyager with GPT-4, but using only 8B LLMs. This makes ODYSSEY significantly more accessible and cost-effective, enabling LLM-based agents to efficiently generate complex policies for broader exploration.

Open-world Tasks. Open-world tasks have gained attention from research communities [Chevalier-Boisvert *et al.*, 2018; Juliani *et al.*, 2019; Shen *et al.*, 2021; Du *et al.*, 2023; Liu *et al.*, 2023; Liu *et al.*, 2024; Jiang *et al.*, 2021; Jiang *et al.*, 2022]. Minecraft, with its diverse tasks and mature game mechanics, has emerged as an ideal test-bed for open-world tasks. Built on Minecraft, MineRL [Guss *et al.*, 2019] implements a simulation environment for agent learning. MineDojo [Fan *et al.*, 2022] further extends MineRL with thousands of diverse tasks. MCU [Lin *et al.*, 2023] collects a variety of atom tasks, offering a method to generate infinite tasks by combining the atom tasks. However, existing benchmarks mainly focus on providing basic programmatic tasks to evaluate agents learned from scratch. Our benchmark is built on top of the skill library, enabling the agents to bypass basic programmatic tasks and focus on complex challenges.

7 Conclusion

This work proposes ODYSSEY to empower agents with open-world skills in the Minecraft environment. The proposed open-world skill library enables the use of open-source LLMs as the foundation for agents to call upon skills, avoiding the high costs associated with previous work using GPT-4 [Wang *et al.*, 2023a; Li *et al.*, 2023; Qin *et al.*, 2023]. The public availability of all datasets, model weights, and code will facilitate future research in the development of autonomous agents. We hope that ODYSSEY will inspire further innovation and progress in the field of autonomous agent. For future work, the open-source LLMs are now prone to generating hallucinations, leading to a decrease in agent performance. Thus, we will focus on employing retrieval-augmented generation to improve LLMs in Minecraft. Additionally, this work focuses on developing text-based LLMs in the context of Minecraft, with visual aspects currently out of scope. Looking ahead, we plan to integrate visual understanding into the skill library to enhance the agent capabilities.

Acknowledgments

This work was supported in part by the CCF-Baidu Open Fund under Grant No. CCF-BAIDU OF202410, in part by the Hangzhou Joint Funds of the Zhejiang Provincial Natural Science Foundation of China under Grant No. LHZSD24F020001, in part by the Zhejiang Province High-Level Talents Special Support Program "Leading Talent of Technological Innovation of Ten-Thousands Talents Program" under Grant No. 2022R52046, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant No. LMS25F020012, and in part by the advanced computing resources provided by the Supercomputing Center of Hangzhou City University.

Contribution Statement

Shunyu Liu, Yaoru Li, Kongcheng Zhang, Zhenyu Cui, and Wenkai Fang made equal contributions.

References

[Achiam *et al.*, 2023] Josh Achiam, Steven Adler, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[Baker *et al.*, 2022] Bowen Baker, Ilge Akkaya, et al. Video pretraining (vpt): Learning to act by watching unlabeled online videos. In *NeurIPS*, 2022.

[Cai *et al.*, 2023] Shaofei Cai, Zihao Wang, et al. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. In *CVPR*, 2023.

[Cai *et al.*, 2024] Shaofei Cai, Zihao Wang, et al. Rocket-1: Master open-world interaction with visual-temporal context prompting. *arXiv preprint arXiv:2410.17856*, 2024.

[Chevalier-Boisvert *et al.*, 2018] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, et al. Babyai: A platform to study the sample efficiency of grounded language learning. In *ICLR*, 2018.

[Deng *et al.*, 2023] Xiang Deng, Yu Gu, et al. Mind2web: Towards a generalist agent for the web. In *NeurIPS*, 2023.

[Ding *et al.*, 2023] Ziluo Ding, Hao Luo, et al. Clip4mc: An rl-friendly vision-language model for minecraft. *arXiv preprint arXiv:2303.10571*, 2023.

[Driess *et al.*, 2023] Danny Driess, Fei Xia, et al. Palm-e: An embodied multimodal language model. In *ICML*, 2023.

[Du *et al.*, 2023] Yuqing Du, Olivia Watkins, et al. Guiding pretraining in reinforcement learning with large language models. In *ICML*, 2023.

[Fan *et al.*, 2022] Linxi Fan, Guanzhi Wang, et al. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *NeurIPS*, 2022.

[Feng *et al.*, 2023] Yicheng Feng, Yuxuan Wang, et al. Llama rider: Spurring large language models to explore the open world. *arXiv preprint arXiv:2310.08922*, 2023.

[Guss *et al.*, 2019] William H Guss, Brandon Houghton, et al. Minerl: a large-scale dataset of minecraft demonstrations. In *IJCAI*, 2019.

[Hafner *et al.*, 2023] Danijar Hafner, Jurgis Pasukonis, et al. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[Hendrycks *et al.*, 2021] Dan Hendrycks, Collin Burns, et al. Measuring massive multitask language understanding. In *ICLR*, 2021.

[Hu *et al.*, 2021] Edward J Hu, Phillip Wallis, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2021.

[Huang *et al.*, 2022] Wenlong Huang, Fei Xia, et al. Inner monologue: Embodied reasoning through planning with language models. In *CoRL*, 2022.

[Iong *et al.*, 2024] Iat Long Iong, Xiao Liu, et al. OpenWebAgent: An open toolkit to enable web agents on large language models. In *ACL*, 2024.

[Jiang *et al.*, 2021] Haobo Jiang, Jin Xie, and Jian Yang. Action candidate based clipped double q-learning for discrete and continuous action tasks. In *AAAI*, 2021.

[Jiang *et al.*, 2022] Haobo Jiang, Guangyu Li, et al. Action candidate driven clipped double q-learning for discrete and continuous action tasks. *TNNLS*, 2022.

[Juliani *et al.*, 2019] Arthur Juliani, Ahmed Khalifa, et al. Obstacle tower: A generalization challenge in vision, control, and planning. In *IJCAI*, 2019.

[Li *et al.*, 2023] Hao Li, Xue Yang, et al. Auto mc-reward: Automated dense reward design with large language models for minecraft. *arXiv preprint arXiv:2312.09238*, 2023.

[Li *et al.*, 2024] Zaijing Li, Yuquan Xie, et al. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *arXiv preprint arXiv:2408.03615*, 2024.

[Liang *et al.*, 2023] Jacky Liang, Wenlong Huang, et al. Code as policies: Language model programs for embodied control. In *ICRA*, 2023.

[Lifshitz *et al.*, 2023] Shalev Lifshitz, Keiran Paster, et al. Steve-1: A generative model for text-to-behavior in minecraft. In *NeurIPS*, 2023.

[Lin *et al.*, 2022] Zichuan Lin, Junyou Li, et al. Juewu-mc: Playing minecraft with sample-efficient hierarchical reinforcement learning. In *IJCAI*, 2022.

[Lin *et al.*, 2023] Haowei Lin, Zihao Wang, et al. Mcu: A task-centric framework for open-ended agent evaluation in minecraft. *arXiv preprint arXiv:2310.08367*, 2023.

[Lin, 2004] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 2004.

[Liu *et al.*, 2023] Shunyu Liu, Yihe Zhou, et al. Contrastive identity-aware learning for multi-agent value decomposition. In *AAAI*, 2023.

[Liu *et al.*, 2024] Shunyu Liu, Jie Song, et al. Interaction pattern disentangling for multi-agent reinforcement learning. *TPAMI*, 2024.

[Mao *et al.*, 2022] Hangyu Mao, Chao Wang, et al. Seihai: A sample-efficient hierarchical ai for the minerl competition. In *DAI*, 2022.

[Oh *et al.*, 2017] Junhyuk Oh, Satinder Singh, et al. Zero-shot task generalization with multi-task deep reinforcement learning. In *ICML*, 2017.

[OpenAI, 2023] OpenAI. Introducing chatgpt. 2023.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, et al. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[PrismarineJS, 2023] PrismarineJS. Mineflayer: Create minecraft bots with a powerful, stable, and high level javascript api. https://github.com/PrismarineJS/mineflayer, 2023.

[Qin *et al.*, 2023] Yiran Qin, Enshen Zhou, et al. Mp5: A multi-modal open-ended embodied system in minecraft via active perception. *arXiv preprint arXiv:2312.07472*, 2023.

[Reed *et al.*, 2022] Scott E. Reed, Konrad Zolna, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[Reimers and Gurevych, 2019] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019.

[Shen *et al.*, 2021] Bokui Shen, Fei Xia, et al. igibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *IROS*, 2021.

[Significant-Gravitas, 2023] Significant-Gravitas. Autogpt: Build & use ai agents. https://github.com/Significant-Gravitas/AutoGPT, 2023.

[Singh *et al.*, 2023] Ishika Singh, Valts Blukis, et al. Progprompt: Generating situated robot task plans using large language models. In *ICRA*, 2023.

[Tessler *et al.*, 2017] Chen Tessler, Shahar Givony, et al. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*, 2017.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[Wang *et al.*, 2018] Alex Wang, Amanpreet Singh, et al. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2018.

[Wang *et al.*, 2019] Alex Wang, Yada Pruksachatkun, et al. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, 2019.

[Wang *et al.*, 2023a] Guanzhi Wang, Yuqi Xie, et al. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[Wang *et al.*, 2023b] Zihao Wang, Shaofei Cai, et al. Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents. In *NeurIPS*, 2023.

[Wang *et al.*, 2023c] Zihao Wang, Shaofei Cai, et al. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *arXiv preprint arXiv:2311.05997*, 2023.

[Wang *et al.*, 2024a] Lei Wang, Chen Ma, et al. A survey on large language model based autonomous agents. *FCS*, 2024.

[Wang *et al.*, 2024b] Zihao Wang, Shaofei Cai, et al. Omnijarvis: Unified vision-language-action tokenization enables open-world instruction following agents. In *NeurIPS*, 2024.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, et al. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[Yang *et al.*, 2023] Aiyuan Yang, Bin Xiao, et al. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*, 2023.

[Yang *et al.*, 2024a] An Yang, Baosong Yang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[Yang *et al.*, 2024b] Songhua Yang, Hanjie Zhao, et al. Zhongjing: Enhancing the chinese medical capabilities of large language model through expert feedback and real-world multi-turn dialogue. In *AAAI*, 2024.

[Yao *et al.*, 2023] Shunyu Yao, Jeffrey Zhao, et al. React: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

[Yu and Lu, 2024] Shu Yu and Chaochao Lu. Adam: An embodied causal agent in open-world environments. *arXiv preprint arXiv:2410.22194*, 2024.

[Yuan *et al.*, 2023] Haoqi Yuan, Chi Zhang, et al. Skill reinforcement learning and planning for open-world long-horizon tasks. *arXiv preprint arXiv:2303.16563*, 2023.

[Zhang *et al.*, 2023a] Chi Zhang, Penglin Cai, et al. Creative agents: Empowering agents with imagination for creative tasks. *arXiv preprint arXiv:2312.02519*, 2023.

[Zhang *et al.*, 2023b] Hongbo Zhang, Junying Chen, et al. Huatuogpt, towards taming language model to be a doctor. In *EMNLP*, 2023.

[Zhao *et al.*, 2023] Zhonghan Zhao, Wenhao Chai, et al. See and think: Embodied agent in virtual environment. *arXiv preprint arXiv:2311.15209*, 2023.

[Zheng *et al.*, 2023] Sipeng Zheng, Jiazheng Liu, et al. Steve-eye: Equipping llm-based embodied agents with visual perception in open worlds. In *ICLR*, 2023.

[Zhou *et al.*, 2024] Enshen Zhou, Yiran Qin, et al. Minedreamer: Learning to follow instructions via chain-of-imagination for simulated-world control. *arXiv preprint arXiv:2403.12037*, 2024.

[Zhu *et al.*, 2023] Xizhou Zhu, Yuntao Chen, et al. Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*, 2023.