EscapeBench: Pushing Language Models to Think Outside the Box

Anonymous ARR submission

Abstract

Language model agents excel in long-session planning and reasoning, but existing benchmarks primarily focus on goal-oriented tasks with explicit objectives, neglecting creative adaptation in unfamiliar environments. To address this, we introduce **EscapeBench**—a benchmark suite of room escape game environments designed to challenge agents with creative reasoning, unconventional tool use, and iterative problem-solving to uncover implicit goals. Our results show that current LM models, despite employing working memory and Chain-of-Thought reasoning, achieve only 15% average progress without hints, highlighting their limitations in creativity. To bridge this gap, we propose **EscapeAgent**, a framework designed to enhance creative reasoning through *Foresight* (innovative tool use) and *Reflection* (identifying unsolved tasks). Experiments show that EscapeAgent can execute action chains over 1,000 steps while maintaining logical coherence. It navigates and completes games with up to 40% fewer steps and hints, performs robustly across difficulty levels, and achieves higher action success rates with more efficient and innovative puzzle-solving strategies.

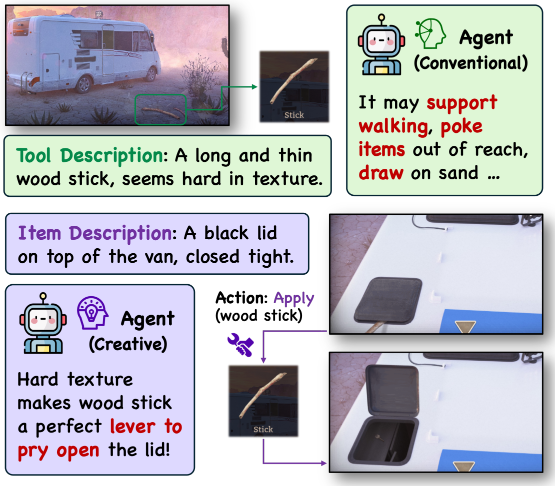

Figure 1: An agent with creative thinking should adapt its observation (e.g. hard texture of wood stick) into a novel tool-use strategy (e.g. prying objects open).

1 Introduction

Building robust language model (LM) agents to perform planning and reasoning has always been a challenging task. Recent efforts have explored how agents could compress and utilize memory (Wang et al., 2023a; Hu et al., 2023; Liu et al., 2023b; Liang et al., 2023b; Wang et al., 2024c; Zhong et al., 2024), perform complex reasoning (Wei et al., 2022; Kojima et al., 2022; Zhou et al., 2023a; Lin et al., 2024; Yao et al., 2023), planning (Wang et al., 2023b; Liu et al., 2023a; Hao et al., 2023; Yao et al., 2024; Zhou et al., 2024a), and reflection (Madaan et al., 2024; Zhang et al., 2024a,b; Miao et al., 2024; Dhuliawala et al., 2024) to improve task success rate. Integrating these capabilities, recent lines of work begin to build agents for embodied actions (Zheng et al., 2024; Huang et al., 2024a; Zhu et al., 2023) and tool use (Schick et al., 2023; Qin et al., 2023; Qian et al., 2024) grounded in environments including the Web (Nakano et al., 2021; Furuta et al., 2024; Gur et al., 2024), games (Guo et al., 2023; Xu et al., 2023; Hu et al., 2024), and society (Park et al., 2023; Liu et al., 2023c; Li et al., 2023; Ren et al., 2024).

The surge of LM agent systems also accelerates the development of simulation environments, including tasks like computer-based operations (Yao et al., 2022; Deng et al., 2024; Zhou et al., 2024b; Xie et al., 2024; Liu et al., 2024b), scientific research (Wang et al., 2022; Bran et al., 2023; Boiko et al., 2023; Huang et al., 2023a), and interactive experiences in text-based (Côté et al., 2019; Urbanek et al., 2019; O'Gara, 2023; Wu et al., 2024) or virtual sandbox game environments (Lin et al., 2023; BAAI, 2023; Wang et al., 2024a). However, most existing benchmarks are usually goal-oriented, emphasizing models' planning, reasoning, and error-handling abilities, while overlooking their *creativity*: the capacity to think innovatively and adapt their observations to new, unstructured scenarios.

Current agents still significantly lack creativity in novel tool use (Zhang et al., 2023), as their training predominantly focuses on memorizing tool-task associations. This emphasis overshadows their ability to explore tool affordances and adapt to unstructured scenarios (Zhang et al., 2024c). Despite this, creativity is still widely recognized as a crucial component of intelligence. In cognitive science, the well-established Triarchic Theory of Intelligence (Sternberg, 1984) divides intelligence into three components: *practical*, *analytical*, and *creative*. While current reasoning benchmarks primarily assess analytical intelligence through problem-solving, and simulation environments focus on practical intelligence by testing knowledge application in real-world scenarios, creative intelligence remains largely unaddressed.

To bridge this gap, we introduce **EscapeBench**, a benchmark that evaluates LM's creative reasoning using scenarios inspired by *room escape* games. These scenarios challenge conventional thinking through unusual settings and require "thinking outside the box" skills including creative tool use and strategic problem-solving. As shown in Figure 1, a wooden stick, typically used for walking or poking, has to be repurposed to pry open a lid due to its hard texture. This demands agents to perform adaptive reasoning under customized constraints. Overall, our benchmark has several distinctive features:

- **Creative Tool Use**: The tools at hand might be repurposed for creative use in order to solve the puzzle. These innovative ways of tool use are uncommon in the LM agent's existing parametric knowledge, requiring it to reason creatively and adapt its observation into customized scenarios.
- **Uncertain Goal Pathways**: While the final goal of each game is escaping from the room, the pathways to achieving it cannot be explicitly foreseen. An agent cannot devise precise, long-range plans initially and must rely on trial and error to discover viable strategies.
- **Super-Long Reasoning Chain**: Each scenario requires even an omniscient agent to perform over 100 steps, with at least 40 bottleneck actions required to achieve the goal. A human player may take up to an hour to complete one game.

We benchmarked multiple models within the BaseAgent framework, which incorporates working memory and Chain-of-Thought reasoning (Wei et al., 2022). Our results show that even the best models struggle to complete the easiest game setting without hints, often requiring up to ten times the optimal steps and falling far behind human performance. These findings highlight how models tend to be constrained by conventional thinking patterns, struggling to break free and show creativity.

To overcome this limitation, we introduce **EscapeAgent**, enhanced with *Foresight* for creative tool use and *Reflection* for implicit goal identification. Foresight enables the agent to propose and evaluate tool-use hypotheses before acting, while Reflection maintains an unsolved task list to guide future actions. Experiments show EscapeAgent reduces hint reliance by nearly 50%, lowers total action steps, and performs robustly across difficulty levels, achieving more efficient progress and higher action success rates with creative strategies. Our contributions include the following:

- We identify challenges in LLM agent creative intelligence and introduce EscapeBench, a robust environment for evaluating agent creativity.
- We present EscapeAgent, which boosts creative reasoning by identifying implicit goals and generating innovative hypotheses.
- We propose measuring creativity through tool use and crafting, and introduce new metrics that provide a fresh dimension for agent evaluation.

2 Related Work

Creativity in Language Models. Creativity is a cornerstone of human intelligence and a growing focus in AI research (Legg and Hutter, 2007; Lake et al., 2017). LMs have demonstrated notable creative capabilities across domains - they excel at generating narratives and poetry (Brown et al., 2020; Akoury et al., 2020), show effectiveness in tool creation and design (Qian et al., 2023; Cai et al., 2024), and augment human creativity through interactive ideation (Mialon et al., 2023). In scientific discovery, research has also found that LM-generated ideas tend to be more novel but slightly less feasible than those from human experts (Si et al., 2024; Wang et al., 2024b).

However, research on LM creativity still remains nascent, emphasizing novelty, surprise, and practical value through psychological assessments like the Alternative Uses Test (AUT)(Guilford, 1967) and Torrance Tests of Creative Thinking (TTCT)

(Boden, 1998). Creativity in LMs is categorized as combinatorial, exploratory, or transformational, with transformational being the most challenging (Franceschelli and Musolesi, 2023). A TTCT study found GPT-4 performing in the top 1% of human fluency and originality, but adapting such assessments to other LMs faces limitations like sample randomness and high evaluation costs (Guzik et al., 2023). Similarly, a modified Torrance Test (Zhao et al., 2024) identified strengths in elaboration and originality but highlighted gaps influenced by prompts and role-play. Notably, most research evaluates backbone models, whereas our work explores creativity within an LM agent-based setting that requires complex reasoning and planning.

Agent Evaluation in Simulated Environment. Agent evaluation often focuses on text-based or sandbox environments for assessing cognitive and behavioral abilities in goal-oriented tasks (Zhou et al., 2023b; Chen et al., 2022, 2024; Yu et al., 2024; Deng et al., 2024), with emerging work exploring LM/VLM-enabled agents in robotics for real-world challenges (Liang et al., 2023a; Huang et al., 2023b, 2024b; Rana et al., 2023). Text-based environments (Yuan et al., 2018; Côté et al., 2019), such as interactive fiction games (Lin et al., 2024) or conversational agents (Qiao et al., 2023), evaluate natural language understanding, reasoning, and decision-making consistency (Uludağlı and Oğuz, 2023; Qi et al., 2024). Games like Zork (Infocom, 1980) and TextWorld measure narrative comprehension and problem-solving in structured contexts. In contrast, sandbox environments (Lin et al., 2023; Gan et al., 2021; Fan et al., 2022) like Minecraft (Zhu et al., 2023) and Roblox (Rospigliosi, 2022) provide more open-ended settings that test spatial reasoning, planning, and collaboration (Carroll et al., 2019; Agashe et al., 2023). The settings typically rely on task-specific metrics for goal achievement but overlook creative and proactive problem-solving in unfamiliar contexts. To address this, we introduce EscapeBench to evaluate agents' creative reasoning in navigating uncertain goal pathways, offering a novel approach to agent assessment.

3 EscapeBench Construction

Most agent benchmarks focus on explicit, goal-oriented tasks grounded in commonsense knowledge, where agents can chart clear pathways to achieve goals using analytical and practical intelligence, but they often overlook creative intelligence. This raises our core research question: **How to build an environment that benchmarks an agent's creative intelligence?** Given that tool use is central to agent functionality, we propose room escape game scenarios, which naturally require creative tool use to solve complex puzzles, as an ideal environment for this evaluation.

3.1 Engine Design

Our game engine aims to simulate the room escape environment that **i)** receives agent actions and **ii)** makes corresponding environment feedback as agent action's reward. Specifically, our game engine involves three key components:

- **Scenes**: The container of tools and items, connected with each other forming a graph structure that constitutes the whole game scenario.
- **Items**: Objects that are intractable in each scene. Tools, inputs, and other interactions may be applied to trigger its state change or other effects.
- **Tools**: Objects that could be collected in each scene, usually applied to other items to take effect or to other tools to craft new ones.

The interaction of these components defines a game's basic logic. In the van example from Figure 2, scenes are connected into a graph, representing physical connectivity via doors or tunnels. The tool *key chain* is collected in the bag for future use, while the *wire iron* awaits something sharp to cut it to trigger effects. Please refer to Appendix A for more detailed examples and explanations.

3.2 Action Space

The model agent could take five different actions. While the action space is well-defined, the parameter space—regarding the scenes, items, or tools involved in these actions—is high-dimensional, thus allowing for dynamic interactions.

- **Move (Scene)**: Move to an adjacent scene.
- **Click (Item)**: Click to simply interact with an item in the scene.
- **Apply (Tool, Item)**: Apply a tool in the bag to an item in the scene.
- **Input (*str*, Item)**: Input an arbitrary string to an item in the scene.
- **Craft (Tool, Tool)**: Use two tools in the bag to craft a new one.

Figure 2 illustrates the connections between game engine components and agent action space. Among all the actions, "Apply" and "Craft" stand out as the most creativity-driven, as they require the agent

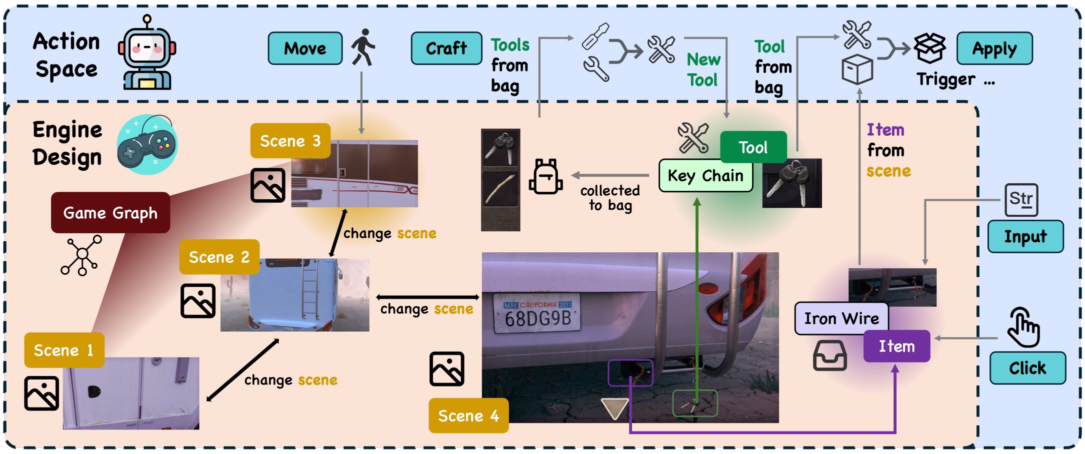

Figure 2: An illustration of Scenes, Tools, and Items in the game and their relations with agent action space. Tools can be collected for "Apply" and "Craft", while items require "Input", "Click" or "Apply" of tools to trigger effects.

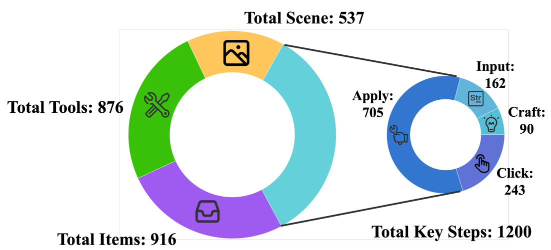

Figure 3: Statistics of total Scenes, Tools, and Items across all game settings. "Key Steps" refer to the essential bottleneck actions required to complete the game.

to think innovatively about how to use or craft tools in an unseen way during its training. We delve into specific examples in Section 3.4.

3.3 Annotation and Statistics

Building on existing online room escape games and puzzle-solving logic[1], we present EscapeBench, featuring 36 game settings across three difficulty levels (see Appendix B for details). All scenes, items, and tools are manually annotated to ensure high quality. These scenarios emphasize creative tool use and crafting strategies, challenging agents throughout the game, making EscapeBench a robust environment for testing creativity. A detailed statistic is presented in Figure 3.

3.4 Preliminary Study

We sample scenarios from EscapeBench and test GPT-4o (Hurst et al., 2024) and LLama-3.1-70B (Dubey et al., 2024)'s creative reasoning performance through case studies in Table 1. Our results reveal that: **i)** EscapeBench presents diverse creative reasoning challenges, including unconventional tool use, implicit numerical puzzles, and innovative tool crafting. **ii)** Both closed- and open-source models struggle with creativity, especially in identifying implicit goals and forming creative strategies. These findings highlight the complexity of EscapeBench and the gaps in model creativity.

[1] https://spotlight.ee

4 EscapeAgent Design

To address challenges identified in the preliminary study, we introduce **EscapeAgent**, a framework addressing two core issues from EscapeBench:

- *Uncertain Goal Pathways*: A **Reflection** module dynamically manages a task list, refining goals through trial-and-error to enhance action focus and proactive task discovery (Section 4.2).
- *Creative Tool Use*: A **Foresight** module enabling explicit reasoning about tool applications, allowing the agent to hypothesize and evaluate strategies before execution (Section 4.3).

Integrating both modules with a BaseAgent, EscapeAgent excels in handling *Super-long Reasoning Chains* and significantly boosts the model's creativity, problem-solving, and strategic thinking.

4.1 BaseAgent

The **BaseAgent** forms the foundation of EscapeAgent (Figure 4) by taking actions based on scenario context and updating its working memory with feedback. This memory stores prior actions and their outcomes. BaseAgent determines its next steps through Chain-of-Thought (Wei et al., 2022), thus serving as a strong baseline and standard eval-

Tool / Item Involved	GPT-4o Inference	Llama-3.1-70B Inference	Actual Creative Use in Game
Tool: soap	Make a surface slippery, clean something to reveal a clue, create bubble to test air current.	Create slippery surface to slide heavy object, write hidden message that can be reveal when wet.	Lubricate an old rusty zipper on a handbag to open it smoothly.
Tool: iron weight, chain, big brush	Attach chain to the iron weight as a swing or pulling tool. Attach the big brush to the other end of chain. Use the combined tool to **manipulate object otherwise inaccessible**.	Use big brush to sweep surrounding area. If a hidden object is found, use iron weight to break or move it. If it is a lock, use chain to trigger it to **reveal hidden door or passage**.	Attach iron weight to one end of the chain, and the big brush to the other end. Using the momentum given by the weight, let it **free fall** from a pipe above fireplace **to clean clogged objects**.
Tool: a license plate written May California 2011, a magazine CA 06 **Item**: 8-digit lock	License plate should suggest '052011', the magazine indicate number 06, combining these should **get the code '05201106'**	California is the 6 issue of the magazine, so 'May California 2011' should translate to **'05 06 2011'**	CA is the abbreviation for California, so substitute California with 06. Since lock is 8-digit, May should be 2-digit, 05, thus getting the **code '05062011'**
Tool: pestle, chalk, tape **Item**: dirty mirror	Use chalk to draw on mirror to review hidden message. Use pestle to clean off dirt on mirror for reflection. Use tape to hold loose pieces on mirror.	Use the pestle to scrape off the dirt on mirror. Use chalk to draw a pattern on clean mirror. Hold the mirror to reflect light to **reveal hidden message**.	Use the pestle to grind the chalk into powder. Use the powder to mirror surface and blow the dust. Use tape to get the revealed fingerprint for **safe authorization**.

Table 1: Creative reasoning cases in EscapeBench and model's responses versus actual creative uses of tools.

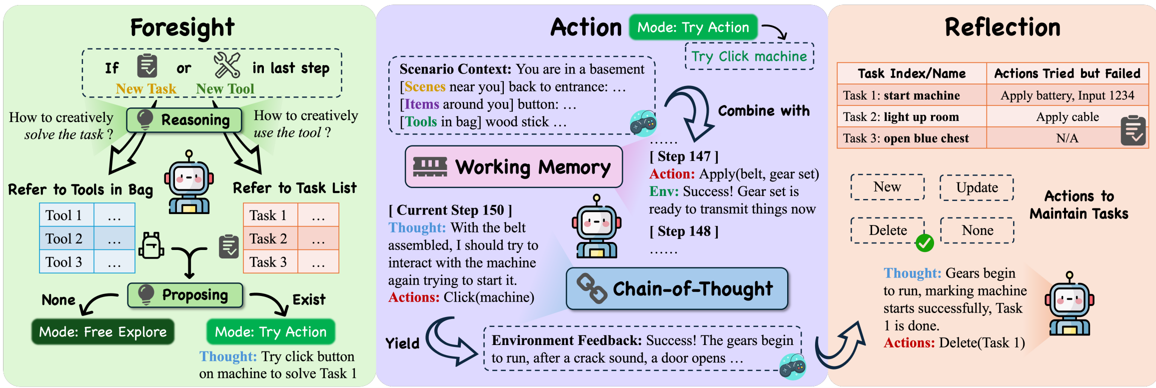

Figure 4: Illustration of the EscapeAgent design. Building on the BaseAgent (*Action*), we integrate the *Foresight* and *Reflection* modules to enhance the agent's capabilities in creative reasoning and implicit goal identification.

uation method for EscapeBench. For implementation details, see Appendix C.1.

4.2 Reflection Module

The **Reflection Module** manages a structured task list updated through three actions:

- *New*: Add a newly identified, unsolved task.
- *Update*: Record attempted but failed actions.
- *Delete*: Remove a task when its goal is achieved.

Each entry includes the task name, target item, and failed actions, preventing repeated mistakes and improving efficiency. Triggered after non-*move* actions, it uses feedback to update the task list. For example, in Figure 4 (right), Task 1 is deleted once the machine starts, encouraging focused problem-solving over random exploration. See Appendix C.2 for details.

4.3 Foresight Module

The **Foresight Module** enhances creative reasoning by explicitly evaluating tool use and problem-solving strategies. It activates in two cases:

- *New Task Identified*: The agent hypothesizes potential actions to achieve it using available tools.
- *New Tool Collected*: The agent assesses its use for solving existing tasks or crafting new tools.

If a valid hypothesis is proposed, the agent enters "Try Action" state to test it; otherwise, it stays in "Free Explore" state, operating like the BaseAgent. For example in Figure 4 (left), the agent identifies clicking the button as an action worth trying, thus guiding it into the "Try Action" state. It enables the agent to adapt flexibly under customized scenarios, make bold hypotheses, and execute targeted trials efficiently. See Appendix C.2 for details.

5 Experiments

We divide experiments into: **i)** Benchmarking model creativity within the BaseAgent, and **ii)** Evaluating EscapeAgent's effectiveness.

5.1 Settings

Environment. Experiments are conducted on 36 game settings. An agent is considered to be **mak-**

Model Name	↓Hints Used	↓Total Steps	↑Early Exit Progress (%)	↓Tool Hints Used (percentage)	↓Key Steps Hints Used (percentage)
GPT-4o	10.30	723.61	24.75	2.17 (8.55%)	8.14 (24.27%)
GPT-4o-mini	15.19	1002.39	16.06	2.00 (8.97%)	13.19 (38.84%)
Claude-3.5-Sonnet	8.97	690.31	28.95	1.34 (5.13%)	7.64 (22.44%)
Gemini-1.5-pro	11.06	824.31	24.18	2.50 (9.89%)	8.56 (24.83%)
Llama-3.1-70B	14.53	982.42	19.00	3.11 (12.22%)	11.42 (33.29%)
Qwen2.5-72B	16.50	1102.50	12.46	5.33 (20.97%)	11.17 (32.02%)
DeepSeek-LLM-67B	25.50	1558.47	6.63	10.50 (42.95%)	15.00 (43.73%)
Yi-1.5-34B	24.00	1573.33	11.96	8.11 (33.83%)	15.92 (46.18%)
Phi-3-medium-128k	32.19	1871.19	7.34	12.11 (49.45%)	20.11 (59.1%)
Llama-3.1-8B	25.86	1543.30	10.10	6.81 (28.56%)	19.11 (56.00%)
Ministral-8B	25.31	1556.97	8.97	7.17 (29.90%)	18.19 (53.87%)
Qwen2.5-7B	32.20	1950.42	6.52	13.81 (55.96%)	18.47 (54.43%)
Average Human	4.33	257.83	59.65	0.17 (0.69%)	4.17 (12.28%)

Table 2: Benchmarking results of **BaseAgent** with different core models on EscapeBench. An oracle action chain's total step is only 107.83 on average. Both closed- and open-source models rely heavily on hints to complete the escape compared to human performance, with smaller-scale models exhibiting a particularly high dependency.

ing progress if it either **achieves a key step** (defined in Figure 3) or **collects a new tool**. Agents will receive help if they fail to make progress for 50 consecutive actions (see Appendix D.1), thus ensuring full completion of the game. The working memory length is set to 10.

Models. We evaluate both closed- and open-source models: GPT-4o, GPT-4o-mini (Hurst et al., 2024), Claude-3.5 (Anthropic, 2024), Gemini-1.5 (Team et al., 2024), Llama-3.1 (Dubey et al., 2024), Qwen-2.5 (Team, 2024), DeepSeek-LLM (Liu et al., 2024a), Phi-3.5 (Abdin et al., 2024), Yi (Young et al., 2024), Ministral (MistralAI, 2024). Models with fewer than 7B parameters are excluded due to near-random behavior. For consistency, we set sampling temperature to $T = 0$ and $n = 1$.

Metrics. We use two main metrics including:

- **Hints Used**: Total hints used in a game.
- **Total Steps**: Total actions taken in a game.

Auxiliary metrics for analysis include:

- **Early Exit Progress**: Proportion of key steps and tools collected before needing a hint (game progress before needing a hint for the first time).
- **Tool Hints Used (percentage)**: Hints used for tool collection (normalized by total tools).
- **Key Step Hints Used (percentage)**: Hints used for key steps (normalized by total key steps).

Results are micro-averaged across the 36 settings.

5.2 Benchmarking Results

We benchmark current models using the BaseAgent framework, with results in Table 2 showing that

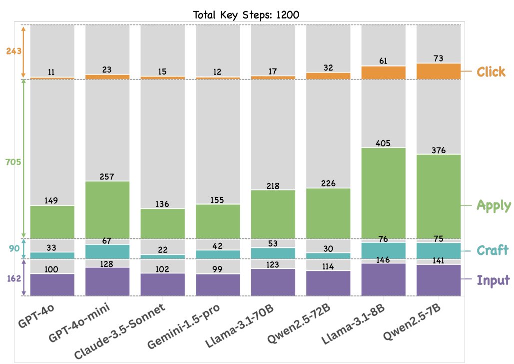

Figure 5: Distribution of Key Steps Hints Used, categorized by different actions. Colored bars represent the percentage of hints used for each action type relative to the total key steps for that type (See right of Figure 3).

large-scale closed-source models consistently outperform smaller models. Key insights include:

- Most hints are used on key steps, which demand creative reasoning, while models may often collect tools through random exploration.
- Models require significantly more action steps and hints than the average human, and up to $\tilde{2}$0x more steps than the most efficient action chain.

We further present an ablation study of total hints and steps used in Appendix D.3.

Input and craft are the most challenging actions. As shown in Figure 5, while "Apply" actions require the most hints in absolute terms, "Input" and "Craft" actions have the highest relative hint usage compared to the total number of key steps. This likely reflects the large parametric space of "Input" actions, where random guesses are impractical, and

Error Type (Explanation)	**Case / Model Action**	**Wrong Reason**	**Common Seen**
Invalid Action The action or parameter is invalid (wrong action, parameter type/number mismatch)	**Context:** There's decoration around door <interactable item> van door **Qwen-2.5-7B:** click(decoration)	Decoration is not interactable, only van door is.	7B/8B smaller scale models
Useless Repetition Repeat same action despite failing last time	**Memory:** last action: click(magazine) Feedback: Nothing happens ... **Llama-3.1-70B:** click(magazine)	Environment gives negative feedback but model still tries consecutively.	All models
Superficial Attempts Not bold enough to try creative action, easily gives up and do other things instead	**Memory:** last action: click(safe) **Context:** There's a mental safe with slot <applicable tool> mental card, license plate **Gemini-1.5-pro:** move(Back to the bed overview)	Model has right mental card to fill in the slot, but give up and turn away after simple click try	Larger scale models
Ignore Environment Overlook item's description and environment feedback, leading to unreasonable trials	**Context:** The license plate has 4 screws fixing tightly **GPT-4o:** apply(bunch of keys, license plate)	License plate needs a screwdriver from description. There's no key hole on it.	All models

Table 3: Error Analysis of **BaseAgent**'s inefficiency or failures. All cases are selected in the same game setting.

Model Name	$^{\downarrow}$**Hints Used**	$^{\downarrow}$**Total Steps**	$^{\uparrow}$**Early Exit Progress (%)**	$^{\downarrow}$**Tool Hints Used (percentage)**	$^{\downarrow}$**Key Steps Hints Used (percentage)**
GPT-4o	$5.03_{\downarrow 5.27}$	$452.75_{\downarrow 270.86}$	$47.03_{\uparrow 22.28}$	$0.33_{\downarrow 1.84}$ (1.19%)	$4.70_{\downarrow 3.44}$ (13.74%)
GPT-4o-mini	$10.58_{\downarrow 4.61}$	$752.25_{\downarrow 250.14}$	$28.17_{\uparrow 12.11}$	$1.14_{\downarrow 0.86}$ (4.16%)	$9.44_{\downarrow 3.75}$ (28.53%)
Llama-3.1-70B-Instruct	$7.92_{\downarrow 6.61}$	$645.19_{\downarrow 337.23}$	$31.44_{\uparrow 12.44}$	$1.42_{\downarrow 1.69}$ (5.44%)	$6.56_{\downarrow 4.86}$ (19.15%)
Qwen2.5-72B-Instruct	$9.72_{\downarrow 6.78}$	$746.61_{\downarrow 355.89}$	$28.62_{\uparrow 16.16}$	$2.72_{\downarrow 2.61}$ (10.61%)	$7.00_{\downarrow 4.17}$ (20.71%)
DeepSeek-LLM-67b-Chat	$20.14_{\downarrow 5.36}$	$1285.03_{\downarrow 273.44}$	$15.30_{\uparrow 8.67}$	$10.81_{\uparrow 0.31}$ (42.72%)	$9.34_{\downarrow 5.66}$ (27.20%)
Yi-1.5-34B-Chat	$22.59_{\downarrow 1.41}$	$1468.03_{\downarrow 105.30}$	$12.04_{\uparrow 0.08}$	$10.42_{\uparrow 2.31}$ (41.61%)	$12.19_{\downarrow 3.73}$ (35.71%)
Phi-3-medium-128k-instruct	$25.75_{\downarrow 6.44}$	$1513.69_{\downarrow 357.50}$	$9.99_{\uparrow 2.65}$	$11.20_{\downarrow 0.91}$ (44.99%)	$14.55_{\downarrow 5.56}$ (43.35%)
Llama-3.1-8B-Instruct	$19.81_{\downarrow 6.05}$	$1271.53_{\downarrow 271.77}$	$19.22_{\uparrow 9.12}$	$6.64_{\downarrow 0.17}$ (25.23%)	$13.22_{\downarrow 5.89}$ (39.34%)
Ministral-8B-Instruct	$19.47_{\downarrow 5.84}$	$1233.72_{\downarrow 323.25}$	$19.34_{\uparrow 10.37}$	$6.61_{\downarrow 0.56}$ (25.56%)	$12.86_{\downarrow 5.33}$ (38.93%)
Qwen2.5-7B-Instruct	$27.53_{\downarrow 4.67}$	$1639.58_{\downarrow 310.84}$	$8.56_{\uparrow 2.04}$	$13.00_{\downarrow 0.81}$ (52.66%)	$14.58_{\downarrow 3.89}$ (43.99%)

Table 4: Benchmarking Results of **EscapeAgent** with different core models. Nearly all the performance raises compared to Table 2, showcasing the effectiveness of EscapAgent in promoting the agent's creativity.

the creativity-demanding nature of "Craft" actions.

Error analysis. We observe from Table 3 that the BaseAgent often gets stuck and relies on hints due to its struggles with environment-following and creativity. Smaller models tend to perform invalid actions in complex scenarios, while larger models excel at tool collection but fail to attempt creative actions, resorting to superficial strategies. Our EscapeAgent addresses this by promoting more purposeful and creative actions.

5.3 EscapeAgent Results

The introduction of the Reflection and Foresight in EscapeAgent, as shown in Table 4, significantly reduces hint uses and total steps, with larger models benefiting the most. Key insights include:

- Larger models still outperform smaller ones, suggesting while new modules aid creative reasoning, the core model's capabilities remain crucial.
- Early exit progress improves across models, despite the exponentially increasing difficulty of consecutively making progress without hints, demonstrating EscapeAgent's effectiveness.

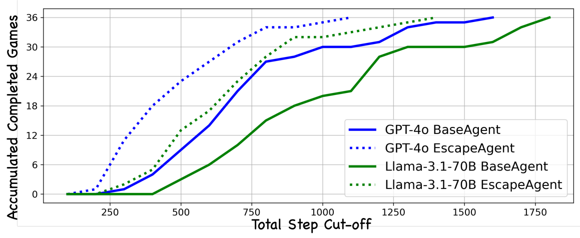

Figure 6: The accumulated number of completed games (in total 36) relative to total steps a game setting takes. EscapeAgent, shown in dotted lines, completes games in fewer steps, demonstrating greater efficiency.

EscapeAgent progresses more efficiently. As shown in Figure 6, EscapeAgent demonstrates steeper progress slopes, reflecting greater efficiency. Figure 8 further shows that EscapeAgent requires fewer actions to reach the next key step, indicating stronger creative reasoning ability. The spike at 50 steps corresponds to hints provided after prolonged inactivity, so the lower pink bar here further highlights EscapeAgent's reduced hint dependency. Notably, across all models, progress after 15 steps without hints is rare, underscoring a lack of sponta-

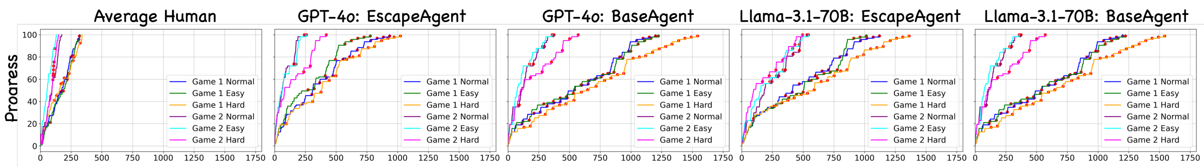

Figure 7: Case study on Human, BaseAgent, and EscapeAgent's progress map corresponding to six game settings. Although EscapeAgent uses 40% fewer hints and makes significant progress independently, it still falls far short of average human performance, often requiring twice as many total steps to complete a game.

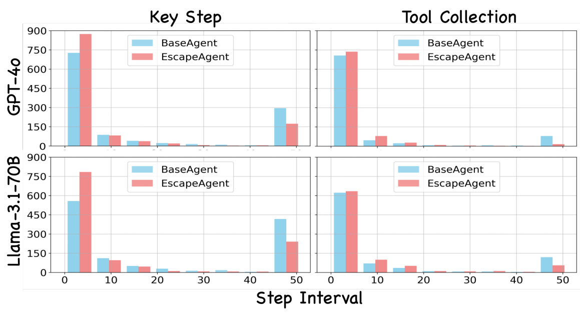

Figure 8: Distribution of *step intervals* for progress made through tool collection and key step achievement. EscapeAgent uses fewer steps to achieve the next progress and relies less on hints.

neous insights typically seen in humans.

Case study. Figure 7 illustrates agent progress relative to action steps across six game settings, with red dots marking hint provided. We observe that: **i)** EscapeAgent requires fewer hints and achieves steeper progress; **ii)** it can make consecutive progress in shorter intervals; **iii)** harder scenarios remain challenging, especially for BaseAgent, which heavily relies on hints; **iv)** Average human performs far better. Humans rarely make mistakes shown in Table 3, while agents still struggle with short memory due to context length and creative tool use strategies. These emphasize the need for further improvements and highlight EscapeBench's challenge to even the most advanced models.

6 Discussions and Future Directions

LM's creativity for benchmarking. Our experiments reveal that even the most advanced language models within EscapeAgent require more hints and twice as many steps as the average human, exposing limitations in creative reasoning and tool use. The benchmark highlights that while analytical and practical intelligence is well-assessed, creative intelligence remains a critical gap. Addressing this gap may require enhancing LMs to link knowledge and objects through affordances—their properties and functions—to foster creativity.

Theoretical Foundations for AI Creativity. Human creativity, characterized by generating novel ideas and adapting to complexity, arises from the interplay of stochastic neuronal noise and structured, learned information (Dainys, 2024; Malach, 2024). In contrast, AI creativity relies on trained data patterns and algorithms. Boden identifies three mechanisms driving AI creativity: combining familiar ideas, exploring conceptual spaces, and enabling transformative innovations (Boden, 1998). Integrating insights from psychology and neuroscience may further enhance AI's creativity.

Human-AI Collaboration. Human-AI collaboration in EscapeBench may promote a new problem-solving paradigm by merging human intuition with AI's systematic reasoning. Humans bring unique insights and ideas that AI might not generate, while AI excels in tasks like information aggregation and logical organization. This synergy fosters innovative strategies, improves efficiency, and creates opportunities for deeper learning, offering a dynamic and enriched problem-solving experience that bridges human creativity with AI's structured problem-solving capabilities.

7 Conclusion

In this work, we introduce EscapeBench, the first benchmark for advancing LM's creativity. Our results show that while LMs still lag in creative reasoning, the EscapeAgent framework improves innovative problem-solving and implicit goal identification. Despite these advancements, enhancing the models' intrinsic creativity remains a challenge. Future work could explore integrating multi-modal perception and new reinforcement learning algorithms to foster greater creativity. Our work serves as an important first step, offering a robust environment for experimentation. Looking ahead, we believe that creative intelligence, beyond just analytical and practical capabilities, will play a key role in shaping the frontier of AI.

Limitations

Our work utilizes a text-based environment to evaluate common language models, focusing on creative reasoning within this framework. However, an Escape Room scenario inherently includes visual and auditory clues, which we have not incorporated into this benchmark. Expanding to include multi-modal inputs could be a valuable next step for future work. Additionally, while the data used in our benchmark is annotated through intensive human effort to ensure high quality, this approach limits scalability. We have explored the use of GPT-4 for automatic annotation through free exploration but found that the model sometimes overlooks important items and clues, and struggles to design environment feedback crucial for adjusting the game's difficulty. We anticipate that more powerful vision-language models may enable better automatic annotation in the future, though current model capabilities are still a limiting factor.

Ethical Statement

In this research, we consider the following ethical issues related to our benchmark and agent design:

• **Fairness:** We ensure that EscapeBench is designed to provide equal evaluation opportunities for all agents, regardless of their underlying model architectures or training methodologies. The tasks and scenarios are crafted to assess creativity and problem-solving abilities without bias, promoting fairness in the benchmarking process. Additionally, we aim to avoid overfitting to specific agent strategies, ensuring a more generalizable and inclusive evaluation framework for future AI advancements. While our environment is robust, we caution against potential misuse and strongly encourage its fair and responsible use.

• **Transparency:** Our work incorporates Chain-of-Thought reasoning in the BaseAgent framework to improve the transparency and interpretability of the agent's decision-making process. This approach makes it easier to attribute the reasoning behind each agent's action. Additionally, we will fully release the benchmarking code, EscapeAgent design, and data to promote transparency in our evaluation process, ensuring that the broader research community can benefit from and build upon our work.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. arXiv preprint arXiv:2404.14219.

Saaket Agashe, Yue Fan, and Xin Eric Wang. 2023. Evaluating multi-agent coordination abilities in large language models. arXiv preprint arXiv:2310.03903.

Nader Akoury, Shufan Wang, Josh Whiting, Stephen Hood, Nanyun Peng, and Mohit Iyyer. 2020. Storium: A dataset and evaluation platform for machine-in-the-loop story generation. arXiv preprint arXiv:2010.01717.

Anthropic. 2024. Introducing claude 3.5 sonnet. https://www.anthropic.com/news/claude-3-5-sonnet.

PKU BAAI. 2023. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. arXiv preprint arXiv:2303.16563.

Margaret A. Boden. 1998. Creativity and artificial intelligence. Artificial Intelligence, 103(1-2):347–356.

Daniil A Boiko, Robert MacKnight, and Gabe Gomes. 2023. Emergent autonomous scientific research capabilities of large language models. arXiv preprint arXiv:2304.05332.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools. arXiv preprint arXiv:2304.05376.

Tom Brown et al. 2020. Language models are few-shot learners. Advances in neural information processing systems.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2024. Large language models as tool makers. In The Twelfth International Conference on Learning Representations.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. 2019. On the utility of learning about humans for human-ai coordination. Advances in neural information processing systems, 32.

Xiusi Chen, Jyun-Yu Jiang, Kun Jin, Yichao Zhou, Mingyan Liu, P Jeffrey Brantingham, and Wei Wang. 2022. Reliable: Offline reinforcement learning for tactical strategies in professional basketball games. In Proceedings of the 31st ACM International Conference on Information & Knowledge Management, pages 3023–3032.

Xiusi Chen, Wei-Yao Wang, Ziniu Hu, David Reynoso, Kun Jin, Mingyan Liu, P Jeffrey Brantingham, and Wei Wang. 2024. Playbest: Professional basketball player behavior synthesis via planning with diffusion. In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, pages 4406–4413.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2019. Textworld: A learning environment for text-based games. In Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers 7, pages 41–75. Springer.

Augustinas Dainys. 2024. Human creativity versus machine creativity: Will humans be surpassed by ai?

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. Advances in Neural Information Processing Systems, 36.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason E Weston. 2024. Chain-of-verification reduces hallucination in large language models. In ICLR 2024 Workshop on Reliable and Responsible Foundation Models.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. Minedojo: Building open-ended embodied agents with internet-scale knowledge. Advances in Neural Information Processing Systems, 35:18343–18362.

Giorgio Franceschelli and Mirco Musolesi. 2023. On the creativity of large language models. arXiv preprint arXiv:2304.00008.

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Shane Gu, and Izzeddin Gur. 2024. Multimodal web navigation with instruction-finetuned foundation models. In The Twelfth International Conference on Learning Representations.

Chuang Gan, Siyuan Zhou, Jeremy Schwartz, Seth Alter, Abhishek Bhandwaldar, Dan Gutfreund, Daniel LK Yamins, James J DiCarlo, Josh McDermott, Antonio Torralba, et al. 2021. The threedworld transport challenge: A visually guided task-and-motion planning benchmark for physically realistic embodied ai. arXiv preprint arXiv:2103.14025.

Joy P Guilford. 1967. Creativity: Yesterday, today and tomorrow. The Journal of Creative Behavior, 1(1):3–14.

Jiaxian Guo, Bo Yang, Paul Yoo, Bill Yuchen Lin, Yusuke Iwasawa, and Yutaka Matsuo. 2023. Suspicion-agent: Playing imperfect information games with theory of mind aware gpt-4. arXiv preprint arXiv:2309.17277.

Izzeddin Gur, Hiroki Furuta, Austin V Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2024. A real-world webagent with planning, long context understanding, and program synthesis. In The Twelfth International Conference on Learning Representations.

Erik E. Guzik, Christian Byrge, and Christian Gilde. 2023. The originality of machines: Ai takes the torrance test. Journal of Creativity, 33(3):100065.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 8154–8173.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. Chatdb: Augmenting llms with databases as their symbolic memory. arXiv preprint arXiv:2306.03901.

Sihao Hu, Tiansheng Huang, Fatih Ilhan, Selim Tekin, Gaowen Liu, Ramana Kompella, and Ling Liu. 2024. A survey on large language model-based game agents. arXiv preprint arXiv:2404.02039.

Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. 2024a. An embodied generalist agent in 3d world. In Forty-first International Conference on Machine Learning.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2023a. Benchmarking large language models as ai research agents. In NeurIPS 2023 Foundation Models for Decision Making Workshop.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. 2024b. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. arXiv preprint arXiv:2409.01652.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. 2023b. Voxposer: Composable 3d value maps for robotic manipulation with language models. arXiv preprint arXiv:2307.05973.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. arXiv preprint arXiv:2410.21276.

Infocom. 1980. Zork I. http://ifdb.tads.org/viewgame?id=0dbnusxunq7fw5ro.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. Advances in neural information processing systems, 35:22199–22213.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. Behavioral and brain sciences, 40:e253.

Shane Legg and Marcus Hutter. 2007. Universal intelligence: A definition of machine intelligence. Minds and machines, 17:391–444.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. Advances in Neural Information Processing Systems, 36:51991–52008.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023a. Code as policies: Language model programs for embodied control. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pages 9493–9500. IEEE.

Xinnian Liang, Bing Wang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2023b. Unleashing infinite-length input capacity for large-scale language models with self-controlled memory system. arXiv e-prints, pages arXiv–2304.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. 2024. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. Advances in Neural Information Processing Systems, 36.

Jiaju Lin, Haoran Zhao, Aochi Zhang, Yiting Wu, Huqiuyue Ping, and Qin Chen. 2023. Agentsims: An open-source sandbox for large language model evaluation. arXiv preprint arXiv:2308.04026.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. 2024a. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. arXiv preprint arXiv:2405.04434.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023a. Llm+ p: Empowering large language models with optimal planning proficiency. arXiv preprint arXiv:2304.11477.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. 2023b. Think-in-memory: Recalling and post-thinking enable llms with long-term memory. arXiv preprint arXiv:2311.08719.

Ruibo Liu, Ruixin Yang, Chenyan Jia, Ge Zhang, Denny Zhou, Andrew M Dai, Diyi Yang, and Soroush Vosoughi. 2023c. Training socially aligned language models in simulated human society. arXiv preprint arXiv:2305.16960.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2024b. Agentbench: Evaluating llms as agents. In The Twelfth International Conference on Learning Representations.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. Advances in Neural Information Processing Systems, 36.

Rafael Malach. 2024. The neuronal basis of human creativity. Frontiers in Human Neuroscience, 18:1367922.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. 2023. Augmented language models: a survey. arXiv preprint arXiv:2302.07842.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2024. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. In The Twelfth International Conference on Learning Representations.

MistralAI. 2024. Introducing ministral-8b-instruct. https://huggingface.co/mistralai/Ministral-8B-Instruct-2410.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. arXiv preprint arXiv:2112.09332.

Aidan O'Gara. 2023. Hoodwinked: Deception and cooperation in a text-based game for language models. arXiv preprint arXiv:2308.01404.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In Proceedings of the 36th annual acm symposium on user interface software and technology, pages 1–22.

Siyuan Qi, Shuo Chen, Yexin Li, Xiangyu Kong, Junqi Wang, Bangcheng Yang, Pring Wong, Yifan Zhong, Xiaoyuan Zhang, Zhaowei Zhang, et al. 2024. Civrealm: A learning and reasoning odyssey in civilization for decision-making agents. arXiv preprint arXiv:2401.10568.

Cheng Qian, Chi Han, Yi Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. In Findings of the Association for Computational Linguistics: EMNLP 2023, pages 6922–6939.

Cheng Qian, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2024. Toolink: Linking toolkit creation and using through chain-of-solving on open-source model. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 831–854.

Dan Qiao, Chenfei Wu, Yaobo Liang, Juntao Li, and Nan Duan. 2023. Gameeval: Evaluating llms on conversational games. arXiv preprint arXiv:2308.10032.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. 2023. Tool learning with foundation models. arXiv preprint arXiv.2304.08354, 10.

Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian D Reid, and Niko Suenderhauf. 2023. Sayplan: Grounding large language models using 3d scene graphs for scalable task planning. CoRR.

Siyue Ren, Zhiyao Cui, Ruiqi Song, Zhen Wang, and Shuyue Hu. 2024. Emergence of social norms in large language model-based agent societies. arXiv preprint arXiv:2403.08251.

Pericles ‘asher’ Rospigliosi. 2022. Metaverse or simulacra? roblox, minecraft, meta and the turn to virtual reality for education, socialisation and work.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In Thirty-seventh Conference on Neural Information Processing Systems.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2024. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. arXiv preprint arXiv:2409.04109.

Robert J Sternberg. 1984. Toward a triarchic theory of human intelligence. Behavioral and Brain Sciences, 7(2):269–287.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Muhtar Çağkan Uludağlı and Kaya Oğuz. 2023. Non-player character decision-making in computer games. Artificial Intelligence Review, 56(12):14159–14191.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 673–683.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. Voyager: An open-ended embodied agent with large language models. Transactions on Machine Learning Research.

Qingyun Wang, Doug Downey, Heng Ji, and Tom Hope. 2024b. Scimon: Scientific inspiration machines optimized for novelty. In Proc. The 62nd Annual Meeting of the Association for Computational Linguistics (ACL2024).

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 11279–11298.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023a. Augmenting language models with long-term memory. In Thirty-seventh Conference on Neural Information Processing Systems.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, Yitao Liang, and Team CraftJarvis. 2023b. Describe, explain, plan and select: interactive planning with large language models enables open-world multi-task agents. In Proceedings of the 37th International Conference on Neural Information Processing Systems, pages 34153–34189.

Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. 2024c. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. In Second Agent Learning in Open-Endedness Workshop.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35:24824–24837.

Yue Wu, Xuan Tang, Tom Mitchell, and Yuanzhi Li. 2024. Smartplay: A benchmark for llms as intelligent agents. In The Twelfth International Conference on Learning Representations.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. arXiv preprint arXiv:2404.07972.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2023. Exploring large language models for communication games: An empirical study on werewolf. arXiv preprint arXiv:2309.04658.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. Advances in Neural Information Processing Systems, 35:20744–20757.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems, 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In The Eleventh International Conference on Learning Representations.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. arXiv preprint arXiv:2403.04652.

Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yupeng Cao, Zhi Chen, Jordan W. Suchow, Rong Liu, Zhenyu Cui, Zhaozhuo Xu, Denghui Zhang, Koduvayur Subbalakshmi, Guojun Xiong, Yueru He, Jimin Huang, Dong Li, and Qianqian Xie. 2024. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making.

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. 2018. Counting to explore and generalize in text-based games. arXiv preprint arXiv:1806.11525.

Chi Zhang, Penglin Cai, Yuhui Fu, Haoqi Yuan, and Zongqing Lu. 2023. Creative agents: Empowering agents with imagination for creative tasks. arXiv preprint arXiv:2312.02519.

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024a. Self-contrast: Better reflection through inconsistent solving perspectives. arXiv preprint arXiv:2401.02009.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. 2024b. Agent-pro: Learning to evolve via policy-level reflection and optimization. In ICLR 2024 Workshop on Large Language Model (LLM) Agents.

Yuji Zhang, Sha Li, Jiateng Liu, Pengfei Yu, Yi R Fung, Jing Li, Manling Li, and Heng Ji. 2024c. Knowledge overshadowing causes amalgamated hallucination in large language models. arXiv preprint arXiv:2407.08039.

Yunpu Zhao, Rui Zhang, Wenyi Li, Di Huang, Jiaming Guo, Shaohui Peng, Yifan Hao, Yuanbo Wen, Xing Hu, Zidong Du, Qi Guo, Ling Li, and Yunji Chen. 2024. Assessing and understanding creativity in large language models.

Duo Zheng, Shijia Huang, Lin Zhao, Yiwu Zhong, and Liwei Wang. 2024. Towards learning a generalist model for embodied navigation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13624–13634.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 19724–19731.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024a. Language agent tree search unifies reasoning, acting, and planning in language models. In Forty-first International Conference on Machine Learning.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2023a. Least-to-most prompting enables complex reasoning in large language models. In The Eleventh International Conference on Learning Representations.

Shuyan Zhou, Frank Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2023b. Webarena: A realistic web environment for building autonomous agents. In NeurIPS 2023 Foundation Models for Decision Making Workshop.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2024b. Webarena: A realistic web environment for building autonomous agents. In The Twelfth International Conference on Learning Representations.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. 2023. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. arXiv preprint arXiv:2305.17144.

Appendix

A Engine Design Details

The game engine involves scenes, tools, and items as three main components. We will introduce in detail each of them in the following.

Scene. A scene typically includes a description, its connections to other scenes, and the tools and items it contains. A typical scene example in the game configuration looks like the following:

Scene Example

```
- name: hallway
  desc: You are in a hallway with a blocked path
  straight ahead, a locked cabinet on the left, and
  a corridor to the right.
  scene_relations:
    To the blocked path close-up: blocked path close
    -up
    To the cabinet close-up: cabinet close-up
  items:
    ...
  tools:
    ...
```

In this example, the name of this scene is "hallway". It leads to nearby scenes including "blocked path close-up" and "cabinet close-up", where the model could reach through action "move(To the blocked path close-up)" and "move(To the cabinet close-up)".

Tool. Each tool has various states and a visibility status. In each state, a tool is either awaiting the application of another tool or ready to be applied to another tool or item. A typical tool example in the game configuration looks like the following:

Tool Example

```
- name: key
  visible: False
  states:
  - desc: A rusty silver key
    wait_for:
    - lubricant
  - desc: A silver key shining bright light, ready
  to use now
    apply_to:
    - safe
```

In this example, a rusty key (*tool, state 1*) hidden in the chest (*item*) won't be visible to the user until the chest is opened, and after applying lubricant (*tool*), it may change to a non-rusty functional key (*tool, state 2*) that could be applied to open a safe.

Item. Item is an upgraded version of the tool, as each state may await multiple inputs or tools in order to trigger certain changes, including item and tool's visibility, state, etc. A typical item example in the game configuration looks like the following:

Item Example

```
- name: digital lock
  states:
  - desc: A digital lock linked to a card reader,
  power on now.
    transitions:
    - wait_for:
      - apply, card
      trigger:
      - change_state, 1
      reward: Authorization succeeds, you have to
      input a 4-digit password.
    - desc: A digital lock already authorized. The
    password panel awaits a 4-digit input.
      transitions:
      - wait_for:
        - input, 1672
        trigger:
        - change_state, item, cabinet door, 2
        - change_state, 2
        reward: The password is correct. A door
        opens somewhere ...
    - desc: A digital lock. You have already input
    the correct password.
```

For instance, a digital lock (*item*) may await the application of card (*tool*) for authorization and correct password input to trigger the closed cabinet door (*item*, state 1) to open (*item*, state 2).

B Data Annotation Details

We recruited eight human annotators, all with prior Room Escape game experience (offline and online) and at least a bachelor's degree. To ensure a smooth annotation process, all annotators were U.S.-based students with computer science backgrounds. Each annotator received detailed guidelines to ensure objective annotations and was tasked with extracting game logic and object descriptions (scenes, items, and tools) based on the official guide. The foundational data logic will be released with the software, and all annotators consented to the data collection.

Difficulty. We define difficulty levels based on the clarity of descriptions and the feedback provided for unexpected actions. Among the 36 game settings, each scenario consists of three settings with varying difficulty levels but consistent game-solving logic. A detailed example is provided in Table 5.

C Agent Design Details

For both BaseAgent and EscapeAgent, we apply the same system prompt for its action-taking to ensure fairness:

System Instruction

```
You are in a Room Escape game. You should explore
the scene and find out what to do next.
There are three types of interactives: items, which
are the interactable things in the scene; tools,
which are applicable tools in your bag; scenes,
whcih are interactable scenes near your current
position.
```

Difficulty Levels	**Easy** Desc. and Env. may hint on how Item / Tool might be used	**Normal** Desc. and Env. only describe Item / Tool's traits	**Difficult** Description only describe Item / Tool's traits, no Env. feedback
Example: Gear set (Item), waiting for application of the **Belt** (Tool)			
Gear set Desc.	A set of gears with one independent, missing something to connect them to transmit things.	There's a set of gears but with one independent from all others.	
Gear set Env.	You may try to assemble something in this gear set to make it intact.		N/A
Belt Desc.	A leather belt near 1 meter long, black and has strong tenacity, could serve as good transmission.	A leather belt that is near 1 meter long, black and has strong tenacity	

Table 5: Rules and examples for different difficulty levels. *Desc.* refers to Item or Tool descriptions, while *Env.* represents the game engine's feedback when an unexpected action targets an Item or Tool.

```
You can perform one of the following actions:
- click(<interactable item>): Click an <interactable
 item> to examine it or interact with it. For
example, you can examine a door handle that is
marked as interactable.
- apply(<applicable tool>, <interactable item>):
Apply an <applicable tool> in your bag to an <
interactable item>. For example, you can apply a key
 in your bag to an interactable locked door to open
it.
- input(string, <interactable item>): Input a string
 (only digits and letters) to an <interactable item
>. For example, you can input a string password to
an interactable combination lock.
- move(<interactable scene>): Move to a nearby <
interactable item> to further explore. For example,
you can move to the living room to explore more
interactable items there.
- craft(<applicable tool>, <applicable tool>): Use
one <applicable tool> in bag to another <applicable
tool> in bag to craft something new. For example,
you can use a battery in bag to a controller in bag
to craft a new charged controller.
For instance, some valid actions may be: click(
microwave), apply(key, silver chest), craft(
controller, battery), input(c79a1, combination lock)
, move(Go to operation room).
```

The system prompts explicit instructs on the agent's action space with examples. In the following, we present in this section more details on EscapeAgent design, including BaseAgent, Reflection, and Foresight modules.

C.1 BaseAgent Details

At each step, the BaseAgent receives information from the game engine. This information typically includes an environment description, a list of interactable objects in the scene, and the tools available in the agent's bag. A typical environment description appears as follows:

```
Scene Description:
You are in the scene 'underneath part of the van'.
There is a stepladder on the right side. There is a
license plate on the left side.
Here are the items you can see in this scene:
- On the left side, there is license plate space:
The license plate is currently fixed to the van,
with four screws on each corner
- On the right side, there is stepladder: The
stepladder is unfolded, now you can reach the top of
 the van,

Possible Actions:
Here are all the items in the scene that you can
perform 'click', 'apply' or 'input':
<interactable item> license plate space
Here are nearby scenes that you can perform 'move'
to further explore:
<interactable scene> Back to the back of the van: It
 leads to back of the van

Tools in Bag:
Here are the tools in your bag. You can perform '
craft' to use two tools in your bag to craft a new
one, or perfom 'apply' to apply one tool in your bag
 to an object in the scene:
<applicable tool> bunch of keys: a bunch of keys
with a keychain and some rust on it
<applicable tool> rag: a rag soaked with engine oil"
```

The scene typically includes a general description, while each item within the scene is accompanied by a detailed description. Possible actions specify which items or aspects of the scene the agent can interact with, and tools in the bag indicate which tools are available for use.

This environment description will also be coupled with working memory of previous steps. Each step's memory contains the following fields:

```
History: [Step ...]
Your position: <How you get to that position from
the beginning scene> e.g. Living Room -> Outside
Corridor
Your action: <The action taken> e.g. move(Explore
the blocked path)
Response from the environment: <Feedback from game
engine> e.g. Action executed successfully. Change to
 another scene: blocked path close-up.
```

Using this information, the BaseAgent is instructed to explicitly apply a Chain-of-Thought reasoning process to determine its next action. This action is then parsed and sent back to the game engine. The game engine updates its state based on the agent's input and provides feedback to the agent. By default, this feedback indicates whether the action was successful or not. In Easy and Normal game settings, additional customized environment feedback is provided. However, in the Hard game setting, no extra information is given.

C.2 Reflection Module Details

The Reflection module is integrated as a downstream component after BaseAgent within the EscapeAgent design. This module is responsible for maintaining a task list that is updated solely based on the agent's current actions and the feedback received. Each task in the list generally includes the following fields:

```
[Task Index <index>] Name: <brief task name>, Target
 Item: <item name>
- Task description: <description of the task>
Example Task:
[Task Index 1] Name: open the chest, Target Item:
chest
- Task description: To open the chest wth a matched
key, I have tried simple click, apply safe key but
all failed.
```

The task index facilitates the identification of specific tasks during task list management operations. The target item specifies the item in the scene that the task is focused on, enhancing the agent's sense of purpose when exploring and performing trials within the scene. The task description outlines a potential strategy for solving the task, including actions the agent has previously attempted but failed. The following system prompt is used to guide the model:

System Instruction

```
You are a helpful agent to reflect on your action
and environment response, and then maintain a task
list with solving suggestions.
The role of this task list is that there are some
tasks you currently cannot solve with the tools at
hand, but you think you may need to solve them later
, so write them down with some suggestions and hints
 for your future reference.

After analyzing your current action and the response
 from the environment, you should give an action to
maintain the task list: update ,new, delete or none.
- update(updated_feedback): The parameter should an
updated feedback about what you newly tried but
failed. The updated feedback should retain the
original feedback, and add one new hindsight you got
 from current action.
- new(task_name, feedback): The first parameter
should be a brief name of the new task you propose,
the second parameter should be what you have to do (
extract hint from environment response) to solve
this task.
- delete(index): If you choose delete, then the
first parameter should be the index of the task in
the task list that you thought you have completed or
 is not useful anymore.
- none(): If you choose none, do not give any
parameter, it indicates you believe you don't need
to perform any action on the task list in the
current step
For instance, valid task list maintaining action may
 be: update(The door has a keyhole and needs a key.
I try apply a hammer but fails.), new(open the safe,
 I need a 4-digit password input to open it with a
hint of sigma sign beside the safe.), delete(1),
none().
```

Note that the operations "update" and "delete" can be implemented in a rule-based manner. An update is triggered when the model attempts an action on a specific item and fails, requiring revised feedback. A delete occurs when the model successfully performs an action that advances progress, necessitating the removal of the corresponding task, if it exists, from the task list.

C.3 Foresight Module Details

The Foresight module serves as an upstream component preceding the BaseAgent in the EscapeAgent design. This module is activated when a new tool is collected during the last action step or a new task is added during the previous reflection step. When a new tool is collected, the agent is provided with the current task list and instructed to propose potential applications for the tool within the context of these tasks and their specific scenarios. Additionally, the agent is given a list of all existing tools in its bag and encouraged to creatively evaluate whether the new tool could be combined with others to craft something useful. The following system prompt is employed to guide the model:

System Instruction

```
You are in a Room Escape game. You have to use your
creativity to figure out the use of the tool you
have just collected.
There are generally two ways about how to use the
tool:
1. Combine this tool with another one in your bag to
 craft a new tool. In this case, use acton 'craft(<
collected tool>, <applicable tool>)', e.g. craft(
controller, battery) indicates use a battery in your
 bag you already have to the controller you just
collected to craft a charged controller.
2. Apply this tool to a target item in a task to try
 solve this task. In this case, use action 'apply(<
collected tool>, Target Item in a task)', e.g. apply
(key, locked cabinet) indicates apply the key you
just collected to a locked cabinet to open it.

Here are some general hints that you may follow:
1. Please especially pay attention to the
description of the task and the tool, try to find
the connection between them to justify your action.
2. In your '- Thought: ...' part in response, you
shuold explicitly think about whether there's item
in bag for crafting, or task in the list for
applying this tool. You should read and infer
carefully from the tool descriptions and the task
description, and evaluate one by one.
3. In your '- Actions: ...' part in response, you
should give zero to multiple action calls. For each
action, you should follow the format 'craft(<
collected tool>, <applicable tool>)' or 'apply(<
collected tool>, Target Item in a task)'. If it's a
craft action, you should justify why crafting here
makes sense. If it's an apply action, you should
first give the task index corresponding to the
target item, then justify why this tool may solve
the task.
```

If a new task is created, the agent is provided with a list of all the tools currently in its bag. It is then instructed to reason creatively about which tools could be applied to address the newly created task. The following system prompt is used to guide the model:

System Instruction

```
You are in a Room Escape game. You have to use your
creativity to figure out if you could use any tools
you have now to solve a new task you have just
discovered.
There are generally three ways to solve a task:
1. Click the target item to simply interact with it
to solve the task. In this case, use action 'click(
Target Item in current task)', e.g. click(microwave)
 indicates click the microwave to examine it and try
 solve the task.
2. Use the tool in your bag to apply to the target
item in the task. In this case, use action 'apply(<
applicable tool>, Target Item in current task)', e.g
. apply(key, locked cabinet) indicates apply the key
 in your bag to a locked cabinet to open it.
3. Input a string to the target item in the task. In
 this case, use action 'input(<any string>, Target
Item in current task)', e.g. input(2413, combination
 lock) indicates input a string password to the
combination lock to solve the task.

Here are some general hints that you may follow:
1. Please especially pay attention to the
description of the task about what might be needed.
Please always first try simple click to interact if
haven't done so. Examine the tool description and
your memory pad, try to find the connection between
them and what this task needs to justify your action
.
2. In your '- Thought: ...' part in response, you
should explicitly think about whether there's item
to click, tool in bag for applying, or hint from
memory pad and tools for string input. You should
read and infer carefully from the task description,
evaluate one by one.
3. In your '- Actions: ...' part in response, you
should give zero to multiple action calls. For each
action, you should follow the format 'click(Target
Item in current task)', 'apply(<applicable tool>,
Target Item in current task)', or 'input(<any string
>, Target Item in current task)'. You shuold justify
 why this action may solve the task according to the
 task description, tool description, and memory pad
hint.
```

Depending on whether the model proposes a valid action, the agent transitions into either the "Free Explore" or "Try Action" state. If multiple actions are proposed, the agent attempts them sequentially until a successful action is achieved. In cases where the task cannot be solved with the currently available tools, the task remains on the task list, and the newly acquired tool stays in the bag. For consistency, all action trials are included in the total count of action steps.

D More Experiment Details

D.1 Help Setting

We provide help to the agent through explicit instructions, focusing on two aspects: i) identifying the next action location that could help the model make progress, and ii) specifying the action the model should take. These objectives are addressed by providing the instruction below to the model during the action-taking step:

Action Instruction

```
Since you're stuck, the system will provide you with
 a hint. You MUST follow the hint to complete next
key step.
The next target location should be: <target position
>.
Your next target action should be: <target action>.
You should go to the target position and perform the
 target action. If you are already at the target
location, please directly perform the action.
```

This help will remain available until the model successfully performs the target action. All the helps on how to complete the game is provided through human annotation.

In each game, there may be cases where multiple actions can be taken in parallel since they do not interfere with one another. As a result, there is no fixed sequence among them. However, to simplify the help provided, the actions are linearized into a single action chain that ensures the agent can complete the game. Whenever the agent requires help, we always provide the first action in this chain that the model has not completed, even if the model may have already succeeded in performing some later actions in the chain.

D.2 Resource Setting

For closed-source models, we utilize standard APIs for testing. Under the BaseAgent framework, the average number of API calls per game setting is approximately 800, resulting in a total cost of \$50–60 per model test. While the EscapeAgent framework introduces two additional modules, these are not always triggered. Consequently, the total number of API calls increases to roughly 1.2 times that of BaseAgent, raising the cost to approximately \$60–80 per model test.

For open-source models, all benchmarks are conducted using the vLLM framework on 2 A100-80G GPUs. Inference time varies based on model size: smaller-scale models complete all 36 game settings in approximately 12 hours, while larger 70B-scale models require about twice as much time for benchmarking. The average tool-calling frequency, reflected in the Total Steps metric, is reported in Table 2 for BaseAgent and Table 4 for EscapeAgent. These metrics vary significantly depending on the specific open-source models being tested.

D.3 Ablation Study

We perform an ablation study on the total steps and used hints in Figure 12 and Figure 13. Generally, harder game settings require more steps and hints for an agent to solve. Since our difficulty setting depends solely on the granularity and usefulness of descriptions and feedback (see Table 5), our

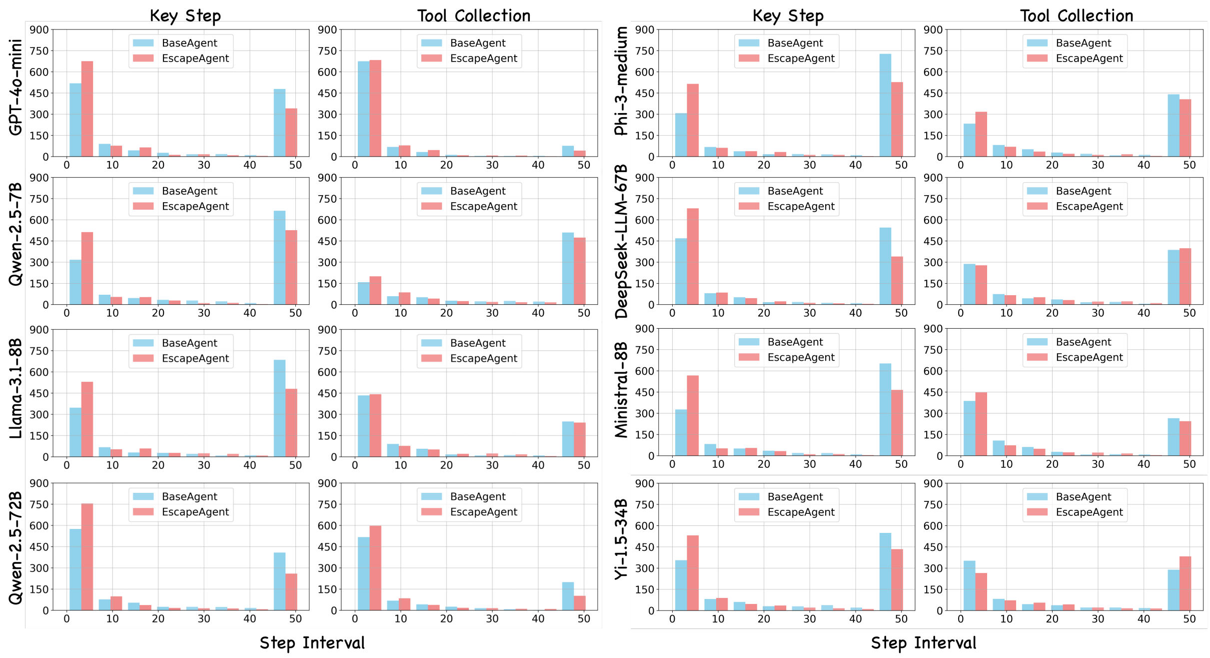

Figure 9: More model's analysis on progress-making interval, extension of Figure 8.

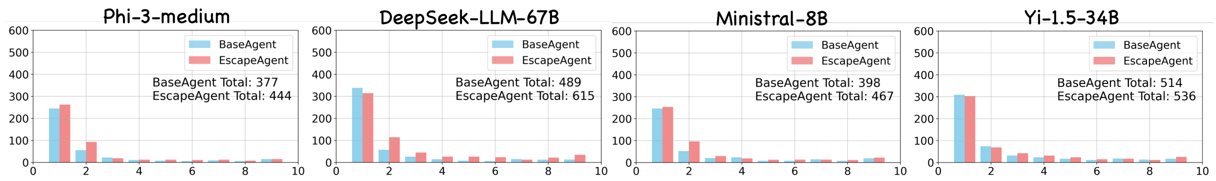

Figure 10: More model's analysis on item trial times, extension of Figure 14.

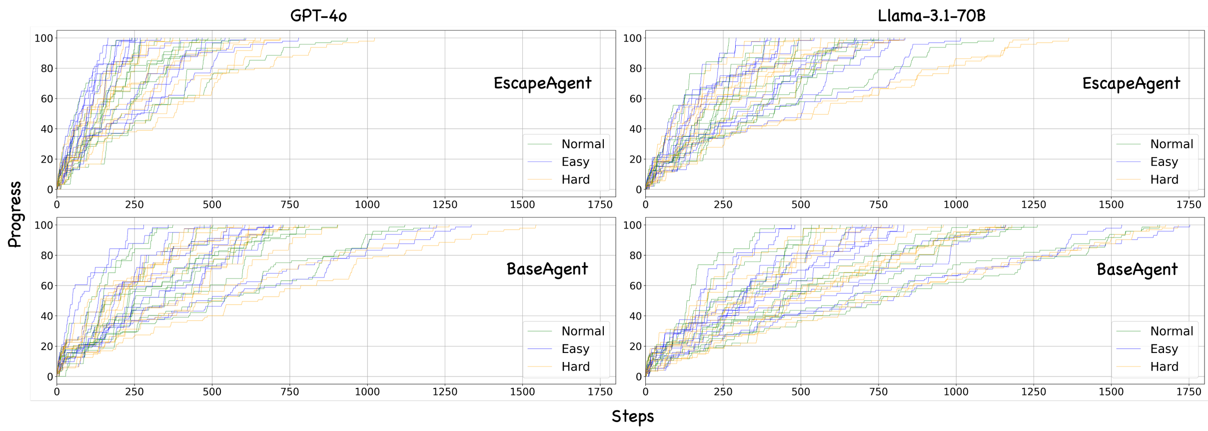

Figure 11: An illustration of progress-making trend through all 36 game settings.

results demonstrate that the way the environment is presented can impact difficulty, even when the core game logic remains unchanged.

D.4 Further Analysis

We further analyze the valid actions agents attempt on different items in scenes before successfully operating them in Figure 14. While the BaseAgent exhibits a higher first-attempt success rate, EscapeAgent achieves greater effectiveness by making multiple attempts, leading to a significantly higher overall success rate within 10 trials. This difference can be attributed to EscapeAgent's strategy of proposing multiple viable actions simulta-

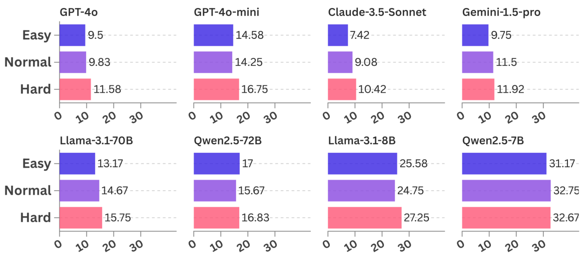

Figure 12: Ablation of difficulty through Hints Used.

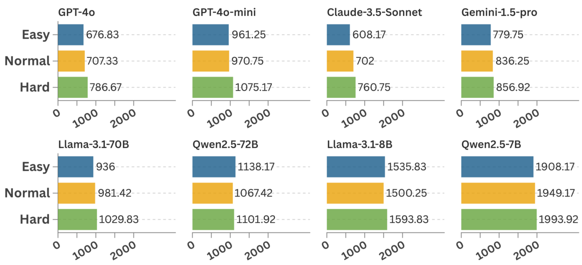

Figure 13: Ablation of difficulty through Total Steps.

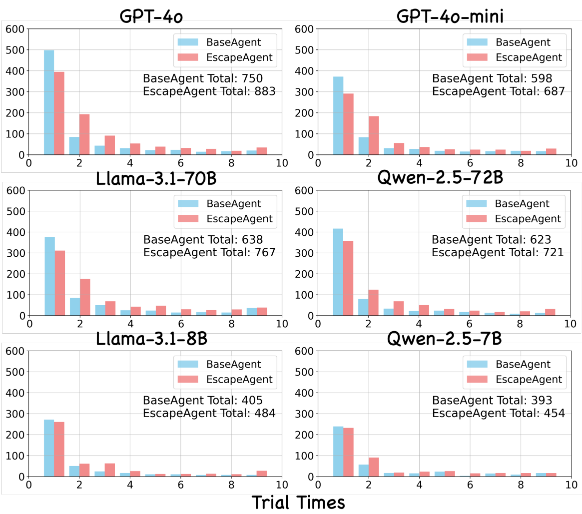

Figure 14: A comparison of action trial times distribution for a specific item before success.

neously. Although this increases the likelihood of eventually succeeding by trying more actions, it does not prioritize the most-likely-to-succeed action first. As a result, the BaseAgent appears more efficient on its initial trial, but its superficial approach, as highlighted in Table 3, limits its overall performance. The EscapeAgent's design effectively addresses this limitation by leveraging a more exploratory approach, which proves advantageous in complex scenarios requiring creativity.

D.5 More Study Results

In Figure 9, we present additional results on the action step intervals with respect to two types of progress-making ways, achieving Key Step or Tool Collection. It can be observed that EscapeAgent consistently requires fewer steps to perform the next bottleneck Key Step. In contrast, for Tool Collection, the difference between the two bars is less pronounced. Despite this, the findings still demonstrate the effectiveness of our design. Tool Collection typically occurs after the successful application or crafting of tools, meaning the reasoning challenge associated with it is significantly lower. Since EscapeAgent focuses primarily on creative reasoning, it is reasonable that it excels in identifying the next Key Action more efficiently.

In Figure 10, we present the results of action trial counts for a specific item across four additional models. We observe that EscapeAgent achieves a higher success rate within 10 trials, even though it does not always succeed on the first attempt. Furthermore, for smaller models like Phi-3 and Ministral, EscapeAgent occasionally outperforms BaseAgent even in terms of one-trial success rates. This highlights how our framework effectively lowers the barriers to creative reasoning, even for less capable language models.

In Figure 15, we showcase eight additional pairs of progress-making maps for eight more models. These case studies illustrate two key points: i) There are significant disparities in creativity among models. For instance, models like GPT-4o-mini and Qwen-2.5-72B require only two-thirds of the total steps than others to achieve success, while smaller models such as Qwen-2.5-7B heavily rely on hints to make progress, even with the EscapeAgent framework. ii) When combining these results with Table 4, we observe that EscapeAgent's performance improvement relative to BaseAgent is more pronounced for larger-scale models like GPT-4o and 70B-scale models. Conversely, while smaller models also benefit from reduced steps and hint usage, they still lag significantly behind in creative intelligence. This underscores the importance of enhancing a model's intrinsic reasoning abilities, as our method primarily mitigates the barriers to creative reasoning but does not fully address inherent limitations.

Lastly, Figure 11 depicts the progress curves across all 36 game settings for GPT-4o and LLama-3.1-70B-Instruct. The trends reveal that

BaseAgent's total step distribution spans a broader range compared to EscapeAgent, resulting in a relatively milder and less steep progression curve. These findings further confirm the effectiveness of EscapeAgent in facilitating more efficient progress.

E More Detailed Discussions

We present a more detailed version of our results discussions and future research directions. **Theoretical Foundations for AI Creativity.** Understanding the cognitive mechanisms behind human creativity is essential for designing AI systems that emulate or surpass human creative processes. Human creativity is a multifaceted phenomenon involving the generation of novel and valuable ideas, problem-solving, and adaptation to complex situations (Dainys, 2024). Neurologically, it arises from the interplay between stochastic neuronal noise and structured, learned information, driven by spontaneous brain activity fluctuations (Malach, 2024). This contrasts with AI's reliance on data patterns and algorithmic processes. Boden identifies three AI creativity mechanisms: combining familiar ideas in novel ways, exploring conceptual spaces, and enabling transformative innovations (Boden, 1998). Our research shows that even advanced models like GPT-4o struggle with implicit goal identification and creative problem-solving, often requiring extensive prompting. The EscapeAgent framework significantly reduces this dependency and improves task-solving efficiency, indicating that these modules effectively overcome barriers to creative reasoning. However, the results also highlight the importance of the core model's capabilities, as larger models like GPT-4o benefit more from the framework than smaller models. This suggests further research is needed to address current models' limitations in generating truly novel ideas. Interdisciplinary approaches integrating psychology, neuroscience, and model architecture can advance agent creativity further.

Multimodal Integration. Expanding the escape room environment to include multimodal data, such as visual and voice cues, offers a promising avenue for enhancing both the agent's performance and the realism of the scenarios. While integrating vision-language models could enable agents to interpret visual clues more naturally, such an extension would also require robust visual understanding and reasoning capabilities to handle the complexity of tasks effectively. Additionally, incorporating multimodal interactions presents opportunities to study how agents synthesize information across modalities, such as correlating visual patterns with instructions or adapting strategies based on dynamic, multimodal feedback. Future work should explore how to seamlessly integrate and harmonize these diverse data types into the benchmark, pushing the boundaries of agents' ability to process, reason about, and act on streams of information.

Step RL for Creative Reasoning. Introducing reinforcement learning into the EscapeBench task is expected to enhance the accuracy and efficiency of the agent's exploration in long-chain tasks. This implies that under the guidance of rewards and penalties, the agent can explore in the correct direction more quickly. Compared with existing end-to-end reinforcement learning schemes, which rely on the final completion of the escape room task as the ultimate reward, introducing step rewards—providing immediate feedback for each step of the model's operation—could potentially accelerate convergence and foster creative advancements in the model. Specifically, the discrete steps in EscapeBench can be organized either as a continuous, logically coherent progression or as strategic, abrupt jumps reflecting non-linear reasoning. Step Rewards can also further draw on the idea of task decomposition to be structured hierarchically. By constructing such a framework, Hierarchical Reinforcement Learning could manage super-long reasoning chains, decompose complex tasks into manageable subtasks, and enable a more fine-grained exploration and evaluation of AI's creative capabilities.

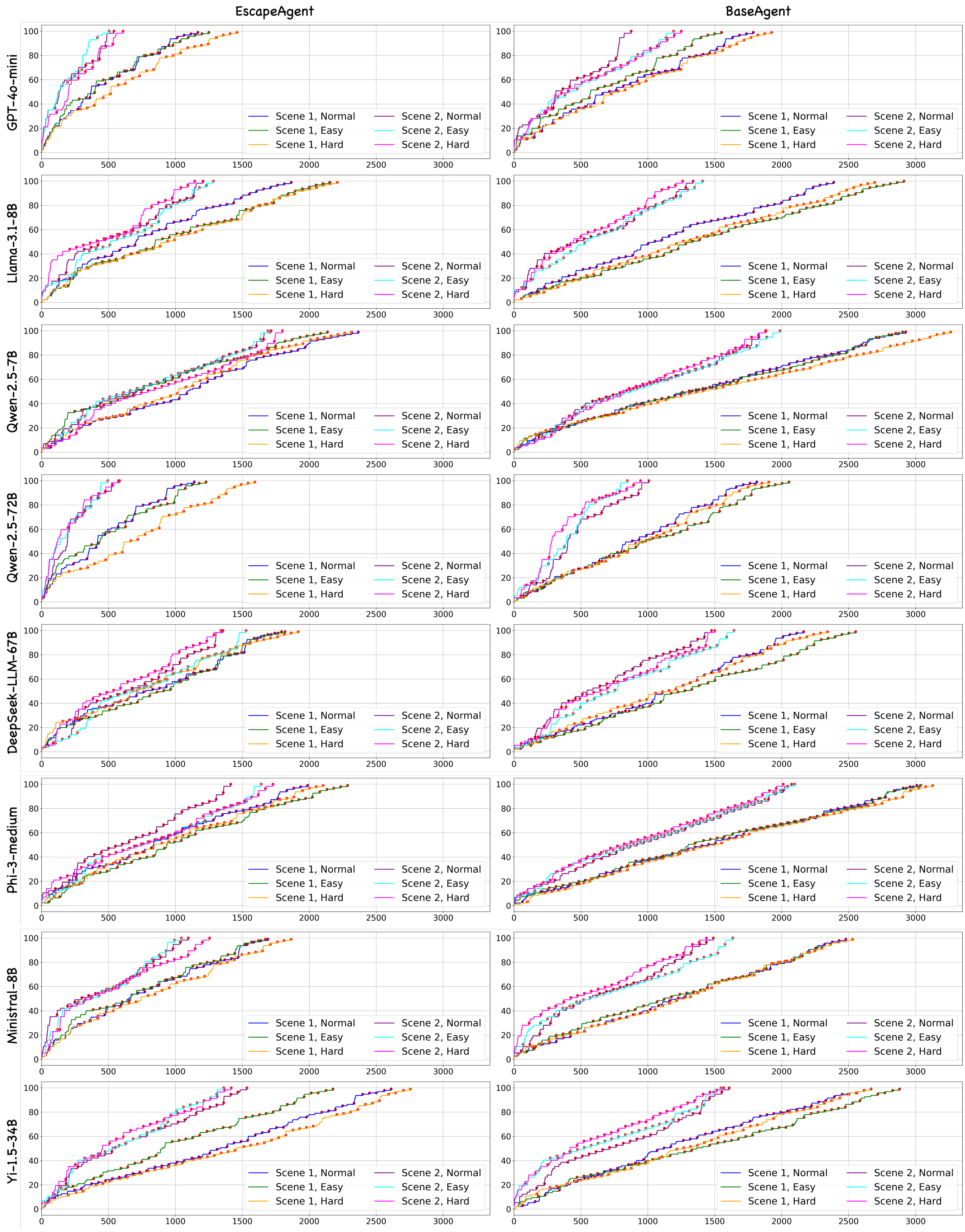

Figure 15: More model's analysis on progress with respect to action steps, extension of Figure 7.