JOINT AGENT MEMORY AND EXPLORATION LEARNING VIA NOVELTY SIGNALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Exploration and memory are two fundamental capabilities for autonomous agents in open-ended environments. However, both are difficult to learn. Effective exploration requires reasoning about the possible consequences of actions and identifying actions that can lead to new states, while a good memory system needs to compress large-scale interaction histories into a form that is easy for an agent to use. Instead of considering these two capabilities separately, we observe that memory and exploration form a mutually dependent loop: sustained exploration requires memory to distinguish exhausted behaviors from unseen ones, while novelty-seeking interactions provide supervision for learning useful memory representations. Based on this insight, we introduce **J**oint **A**gent **M**emory and **E**xploration **L**earning (**JAMEL**), a framework that jointly trains agentic memory and an exploration policy through novelty-driven interaction. JAMEL compresses interaction histories into latent memory tokens and jointly trains memory and policy through novelty-guided rejection sampling fine-tuning on filtered exploration prefixes. We establish evaluation settings for open-environment exploration across GUI, navigation, and household domains. Across these domains, JAMEL consistently outperforms the evaluated baselines, supporting the effectiveness of jointly learning memory and exploration through novelty signals.

1 INTRODUCTION

Exploration is a fundamental capability for intelligent agents to collect information and adapt to new environments. Real-world environments are inherently long-tailed, exposing agents to unseen applications and novel embodied environments whose environment-specific knowledge is not fully captured by pretraining. Such environments often lack human annotations, making environment-specific adaptation difficult to scale. Agents therefore need to acquire environment-specific knowledge through interaction and rapidly adapt their behavior in new environments (Wagenmaker et al., 2025; Xie et al., 2025). Autonomous exploration provides a practical way to support such adaptation. By actively interacting with an environment, an agent can discover available operations, object properties, and interaction outcomes without relying on human annotation for every environment. Exploration progress can often be measured through domain-specific coverage, including code coverage, object discovery, and state changes. However, learning effective exploration is challenging because the agent must reason about the possible consequences of its actions and identify behaviors that can reveal new information (Pathak et al., 2017). As experience accumulates, exploration must continually adapt to what has already been observed, shifting away from previously explored behaviors toward new ones (Wagenmaker et al., 2025).

Memory is another fundamental capability for autonomous agents, enabling them to retain knowledge acquired from past interactions and reuse it in subsequent decisions. However, learning an effective memory system is also challenging. Retaining complete interaction histories provides access to past experience but incurs prohibitively high computational cost over long trajectories. A useful agent memory system must therefore compress past interactions while preserving the information needed for subsequent decisions. Agent memory has received growing attention (Zhang et al., 2025), and latent memory in particular has been studied for its efficiency in compressing interaction history into vectors (Zhang et al., 2026). A key challenge in memory learning is that task-level outcomes provide only sparse supervision for individual memory decisions (Yu et al., 2026; Ma et al., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This makes it difficult to learn both what information should be encoded into compact memory and how the resulting representations should be used to guide subsequent decisions.

We observe that exploration and memory are mutually dependent. Firstly, memory enables the agent to avoid repetition and discover unexplored behaviors. Existing exploration methods already demonstrate the importance of memory for sustained interaction. Some approaches organize past experience as reusable skills that can be retrieved and executed in later situations (Wang et al., 2024a), while others maintain explicit state-transition structures to record visited states, available actions, and navigation paths (Zhao et al., 2025). These designs allow previously acquired knowledge to guide further exploration, but their memory representations and update mechanisms are closely tied to the interaction structure of particular environments. Secondly, exploration exposes which historical information is useful for future decisions. Based on this observation, exploration itself can provide supervision for memory. Reward is given only when the agent reaches behavior not covered by its own history. Maximizing this reward encourages the model to encode and use what has already been tried.

Based on this idea, we introduce **J**oint **A**gent **M**emory and **E**xploration **L**earning (**JAMEL**), a framework for learning autonomous exploration together with compact agent memory. JAMEL compresses interaction history into latent memory tokens and trains an exploration policy through a novelty-driven learning method. Domain-specific coverage signals identify new discoveries and are converted into binary novelty labels for selecting exploration trajectories and constructing supervision. The same learning framework is instantiated in GUI, navigation, and household environments with domain-specific definitions of environmental discovery.

Our contributions are as follows: (1) We propose JAMEL, a joint memory and exploration learning framework that enables agents to autonomously explore new environments and use accumulated experience to continue discovering new behavior. (2) We develop a novelty-driven learning method based on rejection sampling fine-tuning, which converts domain-specific coverage into binary discovery signals for jointly training memory use and exploration policy without human annotation. (3) Experiments across GUI, navigation, and household environments demonstrate improved exploration performance, showing that the same principle of joint memory and exploration learning applies across different domains.

2 Related Work

Agent Memory Efficiently compressing and organizing long-term interaction histories is fundamental to enabling autonomous agents to learn continuously. Long-context architectures extend the accessible context through sparse attention (Beltagy et al., 2020), while retrieval-augmented approaches learn to retrieve external information when needed (Asai et al., 2023).In contrast, prompt tuning methods (Liu et al., 2022; 2024) introduce trainable soft prefixes for parameter-efficient adaptation. To model longer dependencies, recurrent memory transformers (Bulatov et al., 2022) propagate summary states across segments, while recent advances integrate explicit or procedural memory tokens directly into transformer layers. For instance, MemoryLLM and M+ (Wang et al., 2024b; 2025b) maintain layer-wise latent memory pools with controlled forgetting, and TokMem (Wu et al., 2026) tokenizes procedural skills for continual adaptation. Token Memory (Sun et al., 2025a) further stores contextual information to guide generation. These methods learn memory mechanisms for sequence processing, information retention, and reusable procedures. Our approach integrates memory compression directly into the exploration loop, enabling the agent to autonomously consolidate high-value trajectories without external annotation.

Exploration Policy Efficient exploration remains a core challenge in sparse-reward environments where agents must discover valuable states without explicit supervision. Classical approaches propose intrinsic motivation signals like ICM or RND (Pathak et al., 2017; Burda et al., 2019) to encourage novelty through prediction or random network errors. To accelerate search under sparse rewards, archive-based methods like Go-Explore (Ecoffet et al., 2021) maintain frontier states for targeted exploration, while variants such as IGE (Lu et al., 2025) incorporate LLM-based similarity judgments, XTX (Tuyls et al., 2022) combines imitation learning with curiosity, and GLoW (Kim & Hwang, 2026) leverages dual-scale textual world memory for maintaining high-value discoveries and learning from local trial-and-error. Tree-search methods combine Monte Carlo planning

with learned language action-value estimates or memory-guided language-model priors (Jang et al., 2021; Shi et al., 2025). In the GUI domain, GUI-Xplore constructs exploration videos and hierarchical downstream tasks to improve GUI agents' cross-application and cross-task generalization (Sun et al., 2025b), while LLM-Explorer shows that compact knowledge maintained by LLMs can guide efficient app exploration with much lower cost than step-by-step LLM action generation (Zhao et al., 2025). Our work follows this direction but focuses on a different question: how can novelty signals supervise latent agent memory so that exploration-derived knowledge can be encoded into compact memory tokens and used by the policy for future exploration?

3 METHODOLOGY

3.1 EXPLORATION PROBLEM

We model the exploration problem as a finite-horizon partially observable Markov decision process, $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P, \Omega, \rho_0, H)$. At step t, the environment has hidden state $s_t \in \mathcal{S}$, emits an observation $o_t \sim \Omega(\cdot \mid s_t)$, receives an action $a_t \in \mathcal{A}$, and transitions according to $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. The agent observes the current observation and the previous interaction history

$$\mathcal{H}_{<t} = ((o_1, a_1), \ldots, (o_{t-1}, a_{t-1})), \tag{1}$$

and samples actions from $\pi(\cdot \mid o_t, \mathcal{H}_{<t})$. Exploration has no task-specific goal state. Instead, the objective is to expose behavior that has not appeared earlier in the same episode. We formalize this with a novelty score

$$r_t = \mathrm{Novelty}(s_t, \mathcal{H}_{<t}) \in \mathbb{R}_{\geq 0}, \tag{2}$$

where larger values indicate that the current state is less familiar given the visited history. The novelty function is abstract and can be instantiated in different ways. ICM (Pathak et al., 2017), for example, uses world-model prediction error, while JAMEL uses rule-based signals with domain-specific implementations. The exploration objective is

$$J(\pi) = \mathbb{E}_{\tau \sim (\pi, \mathcal{P})} \left[\sum_{t=1}^{H} r_t \right]. \tag{3}$$

Under partial observability, the agent must infer from history which behavior units have already been exhausted. The interaction history and discovery record are maintained within each episode and initialized separately for each new episode.

For domain d, the horizon H is instantiated as an interaction budget B_d, and r_t is instantiated by a novelty rule $r^{\mathrm{nov}}_{d,t}$. The evaluator computes novelty from its record of previously discovered units, using domain-specific identities and reset rules. The domains share a memory and policy-learning framework but use different observations, actions, and novelty units. Method comparisons use the scoring rule of each environment.

Downstream tasks. A downstream task supplies an external goal g and a task reward or completion criterion $R_d(\tau; g)$. We define the downstream objective as

$$J^{(d)}_{\mathrm{task}} = \mathbb{E}_{g \sim p_d(g),\, \tau \sim \pi_d(\cdot \mid g)}[R_d(\tau; g)]. \tag{4}$$

3.2 MODEL ARCHITECTURE OF JAMEL

Let the agent's interaction history up to step t be $H_{<t} = \{(o_1, a_1), \ldots, (o_{t-1}, a_{t-1})\}$. For each historical step, we first serialize the observation-action pair into an input sequence

$$x_i = \mathrm{Format}(o_i, a_i).$$

A vision-language model F_ϕ is then used as the history compressor. We feed x_i into F_ϕ and take the final-layer hidden state of the end-of-sequence token as a compact representation of this step:

$$h_i = F_\phi(x_i)_{\mathrm{EOS}} \in \mathbb{R}^{d_c}.$$

This EOS representation serves as a single latent memory token for the corresponding historical interaction. In this way, each potentially long observation-action pair is compressed into one vector.

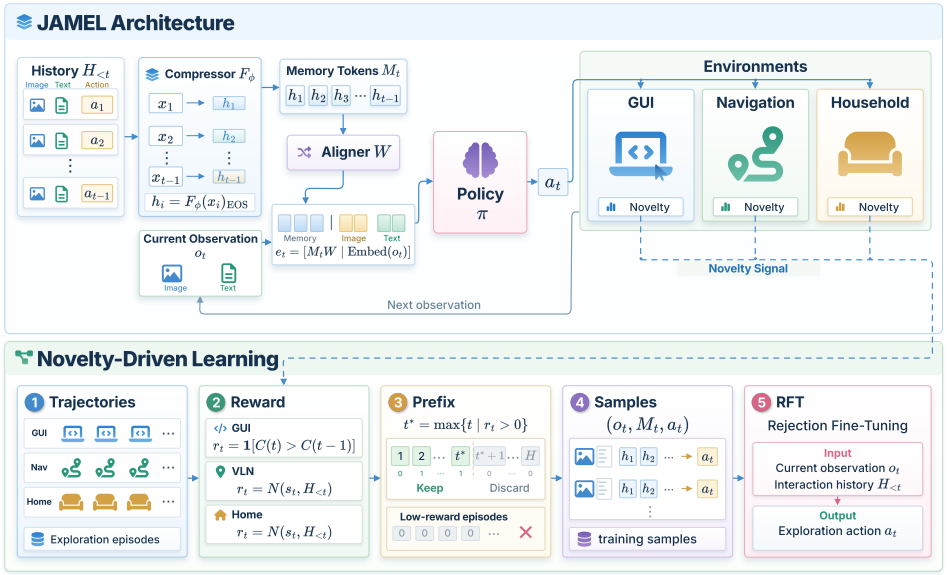


Figure 1: Architecture of JAMEL.

The memory state at time t is the sequence of all the memory tokens:

$$\mathbf{M}_t = [\mathbf{h}_1, \ldots, \mathbf{h}_{t-1}] \in \mathbb{R}^{(t-1)\times d_c}. \tag{5}$$

A learned linear aligner $\mathbf{W} \in \mathbb{R}^{d_c \times d_{\text{LM}}}$ projects the memory tokens into the policy's embedding space. The projected memory is prepended to the input embedding sequence,

$$\mathbf{e}_t = [\mathbf{M}_t\mathbf{W} \mid \text{Embed}(o_t)], \tag{6}$$

and the action is then sampled as $a_t \sim \pi(\cdot \mid \mathbf{e}_t)$.

3.3 Novelty-Based Intrinsic Reward

We seek an exploration signal that measures environmental discovery independently of task-specific success and scales across environments within a domain. Once a domain defines what constitutes a new discovery, the same criterion can be applied to different environments without designing task-specific rewards for each instance. This provides a common supervision interface across environments while allowing the underlying discovery units to remain domain specific.

Directly using the magnitude of coverage increase as reward is less suitable for this purpose. Coverage magnitude depends on how environmental behavior is represented and instrumented. In GUI environments, for example, opening a page may execute a large amount of initialization code, whereas changing a configuration may trigger much less code despite exposing a distinct behavior. The numerical magnitude of a coverage increment therefore does not provide a consistent measure of the value of a discovery.

We use a simple binary abstraction that retains whether an interaction produces a new discovery and discards the magnitude of the corresponding coverage increase. Given the interaction history $\mathcal{H}_{<t}$, the step reward and cumulative novelty over T actions are

$$r_t = \mathbf{1}\big[\text{Novelty}(o_t, a_t, \mathcal{H}_{<t}) > 0\big], N(T) = \sum_{t=1}^{T} r_t. \tag{7}$$

Here Novelty determines whether the current interaction introduces behavior not recorded in the accumulated history. A discovered unit remains familiar within the same episode. Each interaction that expands this record contributes one positive signal regardless of the magnitude of the expansion. For binary rewards, $N(T)$ counts novelty-positive actions, and $N(T)/T$ gives the per-step novelty rate. We instantiate novelty using domain-specific coverage signals, with code coverage for GUI

environments, object discovery for navigation environments, and five-dimensional coverage atoms for household environments. The complete reward definitions and accumulation rules are provided in Appendix A.

Novelty-guided supervision. The binary novelty signal provides a simple criterion for selecting exploration trajectories for supervision. Given a collected episode with trajectory τ and T actions, we first retain trajectories whose cumulative novelty exceeds a threshold η,

$$N(T) \geq \eta. \tag{8}$$

This trajectory-level filtering removes exploration sequences that contain insufficient new discoveries.

For each accepted trajectory, we then locate its last novelty-positive step

$$t^* = \max\{t \mid r_t = 1\}, \tag{9}$$

and retain the prefix $\tau_{\leq t^*}$. Interactions after t^* contain no additional discovery and are removed from supervision. Interactions with $r_t = 0$ before t^* remain in the prefix because they may form intermediate steps leading to a later discovery. Together, threshold filtering and suffix filtering concentrate supervision on trajectories that repeatedly produce new behavior while preserving the interaction sequences that lead to those discoveries.

4 Experiments

We study memory and exploration policy learning in GUI, navigation, and household environments using ScaleWoB (Liu et al., 2026), RoboTHOR (Deitke et al., 2020), and ALFWorld (Shridhar et al., 2021), respectively. The evaluation covers exploration under fixed interaction budgets and goal-directed task execution. Memory variants compare latent memory, textual memory, and a variant without memory across the three domains. Further analyses examine training data scaling, novelty accumulation, input-token consumption, and application-specific interaction patterns.

4.1 Experimental Setup and Metrics

Architecture of JAMEL. A Qwen3-VL-2B-Instruct compressor (Bai et al., 2025a) encodes each historical observation and its associated action into one latent token. A learned alignment layer projects each token from $d_c = 2048$ to $d_{\mathrm{LM}} = 3584$ dimensions for the Qwen2.5-VL-7B-Instruct policy (Bai et al., 2025b).

GUI environments. We select 96 applications from ScaleWoB and randomly partition them into 86 training applications and 10 held-out test applications. Exploration uses $B_{\mathrm{GUI}} = 50$ actions per test application. Agents interact through BrowserGym (de Chezelles et al., 2025) and receive the page accessibility tree and interactive element identifiers. Visual configurations additionally receive a screenshot. The ReAct baselines (Yao et al., 2023) use Gemini 3.1 Flash-Lite (Google DeepMind, 2026) or Qwen3-VL-8B-Instruct (Bai et al., 2025a) and retain the full 50-step interaction history throughout each episode. MAI-UI with MAI-UI-8B (Zhou et al., 2025) maintains a sliding window of recent visual observations while preserving the full textual action and reasoning history. Mobile-Agent-v3.5 with GUI-Owl-1.5-8B (Xu et al., 2026) combines a sliding-window interaction history with explicit notes that summarize information from earlier interactions.

Exploration training uses 24k samples collected from the 86 applications in our training split. Training samples are constructed using the filtering procedure in Section 3.3. We additionally report final total byte coverage after the exploration episode.

Navigation environments. We evaluate exploration on 15 unseen validation scenes in RoboTHOR with four episodes per scene and $B_{\mathrm{emb}} = 300$ actions per episode. We compare JAMEL with three navigation baselines. Random selects actions uniformly from the same navigation action space. ReAct uses Qwen2.5-VL-7B-Instruct (Bai et al., 2025b), while Navigation-PPO (Wang et al., 2025a) is a PPO-trained navigation policy based on Qwen2.5-VL-7B-Instruct. We report novelty and final object coverage described in Appendix A. Coverage is expressed as a percentage. Results are averaged over four episodes within each scene and then over the 15 scenes.

Table 1: Exploration performance across GUI, navigation, and household environments. Novelty reports cumulative binary discovery reward. Coverage reports executed code bytes for ScaleWoB, the percentage of eligible objects observed for RoboTHOR, and the five-dimensional evaluator score for ALFWorld. Bold marks the best result within each domain.

Domain	Method	Model	Novelty	Coverage
GUI	ReAct	Gemini 3.1 Flash-Lite	19.9	5,091,788
	ReAct	Qwen3-VL-8B-Instruct	3.5	3,319,752
	MAI-UI	MAI-UI-8B	8.4	5,035,894
	Mobile-Agent-v3.5	GUI-Owl-1.5-8B	5.9	4,945,518
	JAMEL	JAMEL-9B	**20.7**	**5,409,879**
Navigation	Random	–	3.4	14.0%
	ReAct	Qwen2.5-VL-7B-Instruct	3.2	13.9%
	Navigation-PPO	Navigation-PPO-7B	3.67	16.3%
	JAMEL	JAMEL-9B	**6.1**	**27.7%**
Household	ReAct	Qwen2.5-VL-7B-Instruct	22.5	29.3
	Summary	Qwen2.5-VL-7B-Instruct	8.6	27.3
	Reflexion	Qwen2.5-VL-7B-Instruct	5.3	20.0
	AdaPlanner	Qwen2.5-VL-7B-Instruct	13.9	42.8
	JAMEL	JAMEL-9B	**35.1**	**69.9**

Exploration training in navigation environments uses 60 scenes from the train set of RoboTHOR, including 28.8k training samples.

Household environments. We evaluate exploration in 100 household environments from the ALFWorld validation games with a budget of $B_{\text{house}} = 50$ actions per episode. JAMEL is compared with ReAct (Yao et al., 2023), Summary, Reflexion (Shinn et al., 2023), and AdaPlanner (Sun et al., 2023). Summary maintains an explicit textual memory that is updated at every step. Reflexion uses self-reflection over previous interactions to guide subsequent decisions, while AdaPlanner adaptively updates its task plan based on interaction feedback. We report novelty and average coverage score computed by the five-dimensional ALFWorld evaluator described in Appendix A.

Exploration training uses 13k training samples from 1,200 training games in ALFWorld.

4.2 Exploration Performance

GUI exploration. JAMEL achieves the highest cumulative novelty and byte coverage in the GUI comparison (Table 1). JAMEL outperforms other ReAct and GUI-agent baselines. The GUI-specific baselines cover substantial amounts of code while producing fewer novelty-positive interactions. This contrast illustrates the complementary roles of discovery frequency and final coverage. JAMEL performs well on both measures, supporting the usefulness of novelty-guided learning for exploring unseen applications.

Navigation exploration. JAMEL outperforms the navigation baselines in both cumulative novelty and object coverage (Table 1). More discovery-producing actions are accompanied by a broader set of observed objects. This joint improvement shows that the policy encounters new objects more often and explores a larger share of the scene. The result further shows that the same novelty-guided learning framework remains effective under a different form of environmental interaction.

Household exploration. JAMEL achieves the strongest exploration performance among the evaluated household baselines in both cumulative novelty and coverage (Table 1). The result further shows that the same novelty-guided learning framework remains effective in household environments.

JAMEL's advantage supports learning the memory-conditioned policy from discovery signals. The framework can support exploration under different domain-specific definitions of novelty.

Table 2: Downstream task performance across the three evaluation domains. Bold marks the highest measured result within each domain.

Domain	Benchmark	Method	Model	Success (%)
GUI	**ScaleWoB**	ReAct	Qwen2.5-VL-7B-Instruct	10.8
		JAMEL	JAMEL-9B	**13.5**
Navigation	**RoboTHOR**	ReAct	Qwen2.5-VL-7B-Instruct	8.1
		JAMEL	JAMEL-9B	**30.2**
Household	**ALFWorld**	ReAct	Qwen2.5-VL-7B-Instruct	90.0
		JAMEL	JAMEL-9B	**92.0**

4.3 Downstream Task Performance

The objective in Equation 4 measures goal-directed performance after exploration-oriented training. We evaluate downstream task execution in GUI, navigation and household environments.

GUI tasks. For downstream evaluation, we use the 120 tasks from ScaleWoB benchmark. Among them, 46 tasks require return values, which are not covered by the training data used in our downstream adaptation. We therefore evaluate on the remaining 74 tasks without return-value requirements.

JAMEL and the ReAct baseline are fine-tuned under the same setting using the same 760 cleaned agent trajectories. JAMEL achieves a higher task success rate than ReAct (Table 2).

Navigation tasks. JAMEL achieves a higher navigation success rate than ReAct on 4 RoboTHOR unseen validation scenes (Table 2). Both methods are evaluated directly without downstream fine-tuning, with the same navigation setting and task goals. An episode is successful when the final distance to the target is within 1.5 m.

Household tasks. JAMEL achieves a higher task success rate than ReAct on ALFWorld (Table 2). Both methods are trained on the same 1k trajectories from the training games and evaluated on 100 validation games. Performance is measured by task success rate, where an episode is successful when the specified household task is completed.

4.4 Memory Variants

We compare three memory configurations across the three domains (Table 3). JAMEL uses learned latent memory tokens. The text-summary variant replaces latent memory with an explicit textual summary of previous interactions, while JAMEL w/o memory removes the memory input entirely. All three configurations are trained with the same novelty-driven rejection sampling fine-tuning method and evaluated under the same domain-specific settings.

The variant without memory still discovers new parts of each environment, but fewer interactions produce new discoveries, supporting the usefulness of memory for sustained exploration. Latent memory performs better in ScaleWoB and RoboTHOR, while the text-summary variant performs slightly better in ALFWorld, where observations and actions are purely textual and can be summarized with relatively little information loss. This suggests that textual memory may be particularly effective in text-only environments, while latent memory may offer advantages when exploration depends on multimodal context.

4.5 Analysis

Training data scaling. We examine exploration performance using 10%, 50%, and 100% of the training data in GUI domain. Figure 3 compares mean cumulative novelty in GUI environments. Exploration performance improves as more training data are used. The improvement on held-out applications supports the benefit of increasing the amount of exploration supervision.

Table 3: Memory variants for exploration. Metrics and aggregation follow the domain-specific settings in Section 4.1.

	ScaleWoB		RoboTHOR		ALFWorld	
Method	**Novelty**	**Coverage (Bytes)**	**Novelty**	**Coverage (%)**	**Novelty**	**Coverage**
JAMEL	**20.7**	**5,409,879**	**6.1**	**27.7**	35.1	69.9
JAMEL w/ text summary	20.5	4,354,632	5.9	26.3	**35.6**	**71.3**
JAMEL w/o memory	10.5	4,329,093	5.4	25.5	32.9	66.4

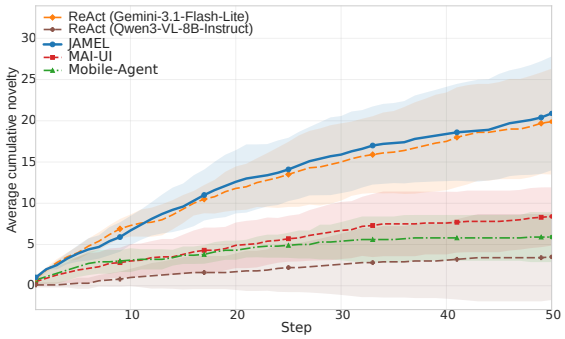


Figure 2: Cumulative novelty averaged across the 10 applications in our GUI test split over 50 steps. Shaded bands denote the standard error across applications.

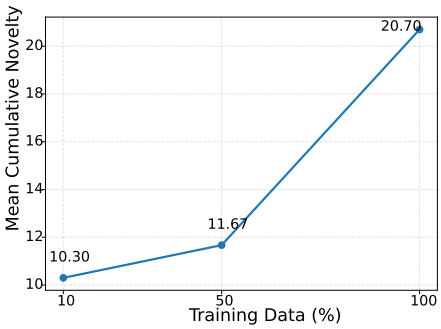


Figure 3: GUI exploration performance with increasing amounts of training data.

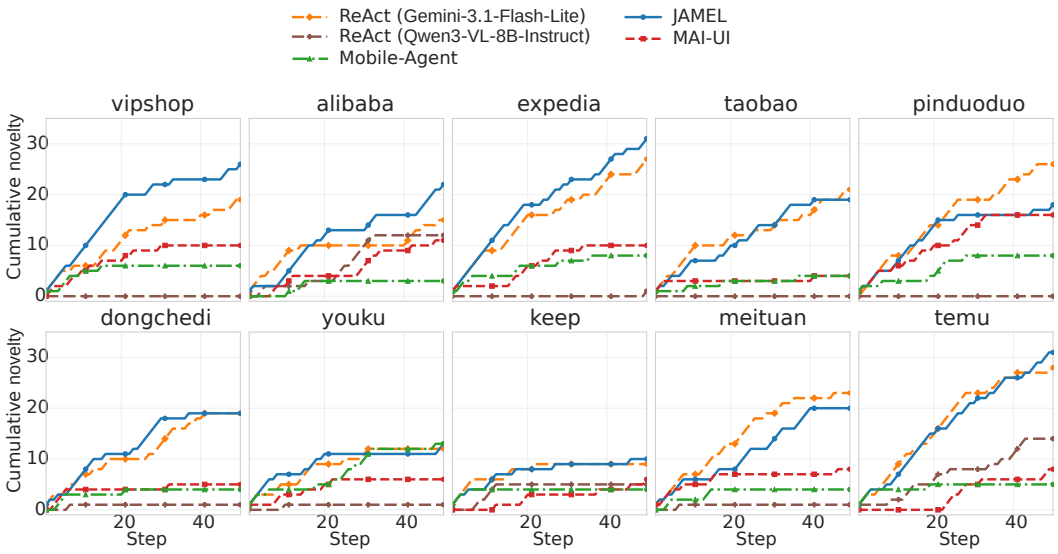


Figure 4: Cumulative novelty on individual unseen applications.

Novelty accumulation. The cumulative novelty curves rise rapidly during the early stages of exploration and diverge later in the episode (Figure 2). MAI-UI, Mobile-Agent-v3.5, and ReAct with Qwen3-VL-8B-Instruct plateau earlier, indicating fewer novelty-positive interactions in the later steps. In contrast, JAMEL and ReAct with Gemini 3.1 Flash-Lite continue to accumulate novelty, showing that they keep discovering new behaviors beyond the initial interactions with an application.

Token efficiency. JAMEL represents interaction history with compact latent tokens and has the lowest total and per-step input-token consumption among the evaluated configurations (Table 5).

Mobile-Agent-v3.5 and MAI-UI use fewer input tokens than the explicit-history ReAct variants, but both remain above JAMEL. ReAct-Vision incurs the highest input-token consumption while retaining explicit history and receiving screenshots. Together with the exploration results, this comparison shows that JAMEL combines strong discovery performance with a compact history representation.

Exploration patterns. In GUI environments, novelty accumulation varies across applications (Figure 4). JAMEL exhibits sustained novelty growth in Vipshop, Expedia, and Temu. Youku and Keep show earlier plateaus across methods, while Alibaba and Taobao exhibit stepwise novelty increases. JAMEL experiences prolonged plateaus in Pinduoduo.

The aggregate result therefore combines sustained discovery in some applications with intermittent progress in others. The per-application curves identify where additional interactions continue to yield discoveries and where JAMEL spends extended intervals without expanding coverage.

5 Discussion

Scaling Laws of Exploration. Future work can examine how exploration performance changes with model capacity, autonomous data scale, environment diversity, and interaction budget. Such studies may help characterize the scaling behavior of novelty-driven memory learning in open environments.

Exploration-Driven Task Execution and Continual Learning. Exploration-derived memory can support an "explore-then-execute" setting, where an agent first explores an unfamiliar environment and then uses the resulting memory for downstream tasks. Extending this process to continual adaptation is a promising direction for future work.

6 Limitations

Our experiments are conducted on a limited set of benchmarks under fixed interaction budgets. The current evaluation therefore does not capture long-term adaptation or knowledge accumulation across repeated interactions with an environment. In addition, our novelty signals rely on domain-specific coverage information, such as code coverage or structured simulator states, which may not be directly available in all environments. Finally, our efficiency analysis measures input-token consumption rather than end-to-end latency or total computational cost, so the broader efficiency of latent memory over much longer trajectories remains to be studied.

7 Conclusion

We presented JAMEL, a framework for jointly learning latent agent memory and exploration policies through novelty-driven interaction. Domain-specific discovery rules provide annotation-free binary signals that guide memory learning, while compact memory tokens encode interaction histories for subsequent decisions. Experiments across GUI, navigation, and household environments show that JAMEL generalizes to unseen environments and achieves higher novelty and coverage than ReAct. Downstream task success also improves across these domains. Memory ablations support the contribution of memory to exploration across all three domains. Analyses further show gains from additional exploration training data and lower input-token consumption with latent memory. These findings support environmental discovery as a possibly scalable source of supervision for learning memory agents that explore unfamiliar environments autonomously.

AI use statement

We used generative AI tools to assist with writing and editing, literature retrieval and discovery, research ideation and experimental planning, drafting parts of the manuscript, and generating synthetic exploration trajectories for training. All AI-assisted content, references, research decisions, and generated data were reviewed and verified by the authors. Experimental results reported in the paper were obtained from actual environment executions. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023. URL https://arxiv.org/abs/2310.11511.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025a. URL https://arxiv.org/abs/2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b. URL https://arxiv.org/abs/2502.13923.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 11079–11091. Curran Associates, Inc., 2022. doi: 10.52202/068431-0805. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/47e288629a6996a17ce50b90a056a0e1-Paper-Conference.pdf.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3. Expert Certification.

Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Google DeepMind. Gemini 3.1 Flash-Lite: Model Card. https://deepmind.google/models/model-cards/gemini-3-1-flash-lite/, March 2026. Published March 2026. Accessed: 2026-05-30.

Istanbul Contributors. Istanbul: JavaScript test coverage made simple. https://istanbul.js.org/, 2026. Software project website. Accessed: 2026-05-30.

Youngsoo Jang, Seokin Seo, Jongmin Lee, and Kee-Eung Kim. Monte-carlo planning and learning with language action value estimates. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=7_G8JySGecm.

Minsoo Kim and Seung-won Hwang. Dual-scale world memory for LLM agents towards hard-exploration problems. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=bH5uHIVtTe.

Guohong Liu, Jialei Ye, Pengzhi Gao, Wei Liu, Jian Luan, Yunxin Liu, and Yuanchun Li. Scale-wob: Guiding gui agents with coding agents via large-scale environmental synthesis, 2026. URL https://arxiv.org/abs/2605.25160.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.8. URL https://aclanthology.org/2022.acl-short.8/.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *AI Open*, 5:208–215, 2024. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2023.08.012. URL https://www.sciencedirect.com/science/article/pii/S2666651023000141.

Cong Lu, Shengran Hu, and Jeff Clune. Intelligent go-explore: Standing on the shoulders of giant foundation models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=apErWGzCAA.

Weitao Ma, Xiaocheng Feng, Lei Huang, Xiachong Feng, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Bing Qin. Fine-mem: Fine-grained feedback alignment for long-horizon memory management. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19666–19684, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.900. URL https://aclanthology.org/2026.acl-long.900/.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2778–2787. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/pathak17a.html.

Zijing Shi, Meng Fang, and Ling Chen. Monte carlo planning with large language model for text-based game agents. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 52924–52944, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/837ff662214b04e7ea8c43f095fe0dd7-Paper-Conference.pdf.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 58202–58245. Curran Associates, Inc., 2023. doi: 10.52202/075280-2537. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/b5c8c1c117618267944b2617add0a766-Paper-Conference.pdf.

Taize Sun, Katsuhide Fujita, Konstantin Markov, and Shengbo Chang. Token memory transformer with infinite context. In *Advanced Intelligent Computing Technology and Applications: 21st International Conference, ICIC 2025, Ningbo, China, July 26–29, 2025, Proceedings, Part XXIV*, pp. 319–330, Berlin, Heidelberg, 2025a. Springer-Verlag. ISBN 978-981-95-0019-2. doi: 10.1007/978-981-95-0020-8_27. URL https://doi.org/10.1007/978-981-95-0020-8_27.

Yuchen Sun, Shanhui Zhao, Tao Yu, Hao Wen, Samith Va, Mengwei Xu, Yuanchun Li, and Chongyang Zhang. Gui-xplore: Empowering generalizable gui agents with one exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19477–19486, June 2025b.

Jens Tuyls, Shunyu Yao, Sham M. Kakade, and Karthik R Narasimhan. Multi-stage episodic control for strategic exploration in text games. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Ek7PSN7Y77z.

Andrew Wagenmaker, Zhiyuan Zhou, and Sergey Levine. Behavioral exploration: Learning to explore via in-context adaptation. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 61885–61912. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wagenmaker25a.html.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Fei-Fei Li, Yejin Choi, and Manling Li. Vagen: Reinforcing world model reasoning for multi-turn vlm agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 172871–172933. Curran Associates, Inc., 2025a. doi: 10.52202/085713-5754. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/fc6688d75adde86b9df910769c1d02e3-Paper-Conference.pdf.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian Mcauley. MEMORYLLM: Towards self-updatable large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50453–50466. PMLR, 21–27 Jul 2024b. URL https://proceedings.mlr.press/v235/wang24s.html.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian Mcauley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending MemoryLLM with scalable long-term memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63308–63323. PMLR, 13–19 Jul 2025b. URL https://proceedings.mlr.press/v267/wang25au.html.

Zijun Wu, Yongchang Hao, and Lili Mou. Tokmem: One-token procedural memory for large language models. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 32460–32480, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/372cb7805eaccb2b7eed641271a30eec-Paper-Conference.pdf.

Bin Xie, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Jie Liu, Min Zhang, and Liqiang Nie. GUI-explorer: Autonomous exploration and mining of transition-aware knowledge for GUI agent. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar

(eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5650–5667, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.282. URL https://aclanthology.org/2025.acl-long.282/.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, Zhiyuan Chen, Jitong Liao, Qi Zheng, Jiahui Zeng, Ze Xu, Shuai Bai, Junyang Lin, Jingren Zhou, and Ming Yan. Mobile-agent-v3.5: Multi-platform fundamental gui agents, 2026. URL https://arxiv.org/abs/2602.16855.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.981. URL https://aclanthology.org/2026.acl-long.981/.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Trans. Inf. Syst.*, 43(6), September 2025. ISSN 1046-8188. doi: 10.1145/3748302. URL https://doi.org/10.1145/3748302.

Zeyu Zhang, Rui Li, Xiaoyan Zhao, Yang Zhang, Wenjie Wang, Xu Chen, and Tat-Seng Chua. Nextmem: Towards latent factual memory for llm-based agents, 2026. URL https://arxiv.org/abs/2603.15634.

Shanhui Zhao, Hao Wen, Wenjie Du, Cheng Liang, Yunxin Liu, Xiaozhou Ye, Ye Ouyang, and Yuanchun Li. Llm-explorer: Towards efficient and affordable llm-based exploration for mobile apps. In *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*, ACM MOBICOM '25, pp. 589–603, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711299. doi: 10.1145/3680207.3723494. URL https://doi.org/10.1145/3680207.3723494.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, and Steven Hoi. Mai-ui technical report: Real-world centric foundation gui agents, 2025. URL https://arxiv.org/abs/2512.22047.

A Novelty Reward in Different Domains

A.1 GUI Exploration reward

GUI exploration measures the mean cumulative binary coverage-increase reward $N^{\mathrm{GUI}}(50)$ using the cumulative novelty in Equation 7. GUI novelty is computed from cumulative code-coverage records. The cumulative coverage score after action t is

$$\mathcal{C}(t) = \mathrm{cov}_{\mathrm{lines}}(t) + \mathrm{cov}_{\mathrm{branches}}(t) + \mathrm{cov}_{\mathrm{statements}}(t) + \mathrm{cov}_{\mathrm{functions}}(t), \tag{10}$$

where each term counts the total number of coverage entities executed at least once across all steps up to and including t. The GUI reward is

$$r_t^{\mathrm{GUI}} = \mathbf{1}\big[\mathcal{C}(t) > \mathcal{C}(t-1)\big]. \tag{11}$$

An action receives a reward of one when coverage increases, regardless of how many coverage entities it reveals. We collect coverage data via the V8 JavaScript engine's coverage reports and compute $\mathcal{C}(t)$ using the Istanbul reporter (Istanbul Contributors, 2026). Steps with invalid actions or execution errors receive $r_t^{\mathrm{GUI}} = 0$. The coverage record is accumulated within each episode.

A.2 Navigation Exploration Reward

In the RoboTHOR exploration setting, coverage measures discovered object instances. The exploration reward indicates whether an action reveals at least one previously unseen eligible instance. No target object or navigation instruction is specified during exploration.

Eligible objects and visibility. For scene s, let $\mathcal{O}_s^{\text{eligible}}$ contain all objects whose simulator metadata marks them as `pickupable` or `openable`. This inventory is fixed at episode reset and includes objects that are not initially visible. Objects are identified by their simulator `objectId`; two instances of the same category count separately. Let $V_t \subseteq \mathcal{O}_s^{\text{eligible}}$ be the objects marked `visible` by the simulator after action t, with V_0 denoting the initial view. Visibility uses the standard 1 m visibility-distance setting. Object eligibility and visibility are obtained from simulator metadata, rather than inferred from generated descriptions.

Episode-local accumulation. Let D_t denote the set of discovered eligible instances. We initialize $D_0 = V_0$ and, after each action, compute

$$\begin{aligned} \Delta_t &= V_t \setminus D_{t-1}, \\ D_t &= D_{t-1} \cup V_t. \end{aligned} \tag{12}$$

Discovery sets are reset between episodes, including repeated rollouts in the same scene. Reobserving a known instance does not create a new discovery.

Binary novelty reward. The per-action reward and cumulative novelty over T actions are

$$\begin{aligned} r_t^{\text{VLN}} &= \mathbf{1}[|\Delta_t| > 0], \\ N^{\text{VLN}}(T) &= \sum_{t=1}^{T} r_t^{\text{VLN}}. \end{aligned} \tag{13}$$

An action revealing several new instances receives one reward. An action revealing none receives zero. Thus cumulative novelty counts discovery-producing actions, not the number of discovered objects. Initial-view objects seed D_0 and contribute to coverage, but receive no novelty reward. Action success is logged separately and does not itself produce exploration reward.

Object coverage. For rollout r in scene s, the reported percentage is

$$C_{s,r}(T) = 100\,\frac{|D_{s,r}(T)|}{|\mathcal{O}_s^{\text{eligible}}|}. \tag{14}$$

The denominator is the complete eligible-object inventory of that scene, not just the objects encountered along the trajectory. Coverage therefore measures object-instance discovery rather than traversed floor area or the number of distinct semantic categories. The evaluator returns full coverage for an empty eligible inventory. As an illustrative example, if a scene has 20 eligible instances and two are visible initially, initial coverage is 10%. An action revealing three additional instances raises coverage to 25% while contributing only one novelty reward.

Evaluation setting and aggregation. We evaluate 15 held-out RoboTHOR validation scenes with four rollouts per scene and a budget of $T = 300$ executed actions per rollout. Each rollout starts from the scene's fixed default pose. The action space contains `moveahead`, `moveback`, `rotateright`, and `rotateleft`, with translations of 0.25 m and rotations of 30°. There is no `stop` action. The policy receives 255×255 RGB observations and the observation text specified in Appendix C.2. Both metrics are averaged over all 60 rollouts, equivalently averaging the four rollouts within each scene and then averaging the 15 scene means:

$$\begin{aligned} \overline{C} &= \frac{1}{15}\sum_{s=1}^{15}\frac{1}{4}\sum_{r=1}^{4} C_{s,r}(300), \\ \overline{N} &= \frac{1}{15}\sum_{s=1}^{15}\frac{1}{4}\sum_{r=1}^{4} N_{s,r}^{\text{VLN}}(300). \end{aligned} \tag{15}$$

The reported scores are means over rollouts, not best-of-four scores.

Table 4: Coverage atoms in the five dimensions of the ALFWorld exploration evaluator. Each dimension has its own set of previously observed atoms.

Dimension	Atom	Meaning
`location`	ℓ	Semantic room or location type with instance identifiers removed.
`entity`	c	Observed object type with instance identifiers removed.
`interaction`	(u, c)	Interaction represented by a normalized verb and an entity type.
`attribute`	$(c, v_{\text{old}}, v_{\text{new}})$	Change between discrete attribute values for an entity type.
`relation`	$(c, \rho_{\text{old}}, \rho_{\text{new}})$	Type-level relation change, such as a change in an object's associated container.

A.3 Household Exploration Reward

The ALFWorld exploration evaluator represents discoveries as coverage atoms across five dimensions. A coverage atom is a canonical record of a location type, an entity type, an interaction, an attribute change, or a relation change. Novelty depends on whether an action introduces an atom that has not previously appeared in the current episode.

Structured atom extraction. Coverage atoms are extracted from structured transition records supplied by the environment adapter through `extract_transition`. Atom identities are determined by these records. Changes in the wording of observation text leave the atom identities unchanged.

Coverage dimensions. Let

$$\mathcal{D} = \{\text{location}, \text{entity}, \text{interaction}, \\ \text{attribute}, \text{relation}\} \tag{16}$$

denote the five dimensions. Table 4 defines the atom representation in each dimension. Here ℓ is a semantic location type, c is an entity type, and u is a normalized interaction verb. The symbols v_{old} and v_{new} denote discrete attribute values before and after a change. The symbols ρ_{old} and ρ_{new} denote the corresponding type-level relations.

Canonicalization and factorization. Location and entity identifiers are reduced to semantic types. For example, `mug_1` is mapped to `mug`. Interaction verbs are normalized to canonical forms, including `take` to `pickup` and `turn_on` to `activate`. Different object instances that yield the same canonical atom share a single coverage entry. Attribute and relation atoms include both the old and new values, so distinct recorded changes are tracked separately.

Coverage is accumulated separately in each dimension. A new combination of previously observed atoms does not create an additional atom. Thus, repeated visits and interactions produce novelty reward only when they introduce a new canonical atom in at least one dimension.

Episode-local accumulation. For each dimension $k \in \mathcal{D}$, let $A_t^{(k)}$ be the atoms extracted after action t, and let $K_{t-1}^{(k)}$ contain the atoms previously seen in that dimension within the current episode. The newly discovered atoms and their total count are

$$\Delta_t^{(k)} = A_t^{(k)} \setminus K_{t-1}^{(k)}, \\ m_t = \sum_{k \in \mathcal{D}} \left| \Delta_t^{(k)} \right|. \tag{17}$$

The binary reward and cumulative novelty over T actions are

$$r_t^{\text{Home}} = \mathbf{1}[m_t > 0],$$
$$N^{\text{Home}}(T) = \sum_{t=1}^{T} r_t^{\text{Home}}. \tag{18}$$

An action that introduces several new atoms contributes one reward. An action whose atoms have all been seen contributes zero. After scoring the action, the evaluator updates each dimension independently as

$$K_t^{(k)} = K_{t-1}^{(k)} \cup A_t^{(k)}. \tag{19}$$

The evaluator retains these sets within an episode and resets them between episodes. Coverage records are not shared across episodes. The cumulative quantity $N^{\text{Home}}(T)$ is stored as `novelty_total` and counts actions that introduce new atoms. Task reward is recorded separately from the exploration reward.

Coverage reporting. Coverage is the episode-level score produced by the five-dimensional evaluator from the accumulated location, entity, interaction, attribute, and relation atoms. Cumulative novelty counts actions that introduce at least one new atom. These metrics describe accumulated coverage and discovery-producing interactions, respectively.

B Token Efficiency

Table 5: Input-token consumption over the 10 applications in our GUI test split with 50 steps per application. Relative consumption uses JAMEL as the reference. Bold marks the lowest values.

Method	Input Tokens	Avg. per Step	Rel. to JAMEL
Mobile-Agent-v3.5	2,931,946	5,863.9	2.76×
MAI-UI	2,980,061	5,960.1	2.81×
ReAct	18,938,833	37,877.7	17.85×
JAMEL	**1,061,103**	**2,122.2**	**1.00×**

C Prompt Design

C.1 GUI Domain

C.1.1 Shared BrowserGym Action Space

```
noop(wait_ms: float = 1000)
send_msg_to_user(text: str)
report_infeasible(reason: str)
scroll(delta_x: float, delta_y: float)
fill(bid: str, value: str)
select_option(bid: str, options: str | list[str])
click(bid: str, button: Literal['left', 'middle', 'right'] = 'left',
modifiers: list[Literal['Alt', 'Control', 'ControlOrMeta', 'Meta', '
Shift']] = [])
dblclick(bid: str, button: Literal['left', 'middle', 'right'] = 'left',
modifiers: list[Literal['Alt', 'Control', 'ControlOrMeta', 'Meta', '
Shift']] = [])
hover(bid: str)
press(bid: str, key_comb: str)
focus(bid: str)
clear(bid: str)
drag_and_drop(from_bid: str, to_bid: str)
upload_file(bid: str, file: str | list[str])
tab_close()
```

```
tab_focus(index: int)
new_tab()
go_back()
go_forward()
goto(url: str)
reset()
```

C.1.2 EXPLORATION PROMPT

```
You are an autonomous browser exploration agent.

This is session step {step_idx}.
Your goal is to explore the target app and maximize novel JavaScript
execution coverage.

Target app: {target_app}
Start URL: {start_url}

Browser action space:
{action_space}

Current open pages URLs:
{open_urls}

Current Observation:
{pruned_axtree}

Current valid interactive element ids:
{element_ids}

The current webpage screenshot is:
<image>

Respond with exactly:
<action>one action</action>

The <action> content must be one single BrowserGym action call. If the
action
uses a bid, use one exact id from the valid interactive element list.
Never
invent bids and never combine two actions in one response.
```

C.1.3 DOWNSTREAM TASK PROMPT

```
You are an autonomous GUI task agent.

Target app: {target_app}
Start URL: {start_url}

Task to complete:
{task_description}

Actions already executed in this task:
{task_history}

Browser action space:
{action_space}

Current open pages URLs:
{open_urls}

Current Observation:
{pruned_axtree}
```

```
Current valid interactive element ids:
{element_ids}

The current webpage screenshot is:
<image>

Respond with exactly:
<action>one action</action>

The <action> content must be one single BrowserGym action call. If the
action
uses a bid, use one exact id from the valid interactive element list.
Never
invent bids and never combine two actions in one response.
```

C.2 Navigation Domain

C.2.1 Exploration Instruction and Action Contract

```
You are a mobile robot exploring an unfamiliar RoboTHOR apartment.
Your goal is to discover as many distinct pickupable or openable
object instances as possible within the action budget.

Choose exactly ONE action per step:
- moveahead: move forward 0.25m
- moveback: move backward 0.25m
- rotateright: turn right 30 degrees
- rotateleft: turn left 30 degrees

Choose your next action using the current image, observation text,
and any available exploration memory.

Reply with exactly one action in this format:
<answer>action</answer>
```

C.2.2 Policy Input and Observation Template

```
{exploration_instruction}

=== Exploration Memory ===
{memory_text}

Step {step_idx} observation:
<image>

Position: ({x}, {y}, {z})
Rotation: {yaw} degrees
Camera horizon: {horizon} degrees
Coverage before action: {coverage_percent}%
Visible object types: {comma_separated_types_or_none}

Choose your next action:
```

C.3 Household Domain

C.3.1 Exploration Prompt

First Step:

```
You are an expert agent operating in the ALFRED Embodied Environment.
Your current observation is: {current_observation}
```

```
Your admissible actions of the current situation are: [{
admissible_actions}].

Return exactly one action wrapped in <action> </action> tags.
Do not include reasoning, commentary, or any other text.
```

With History:

```
You are an expert agent operating in the ALFRED Embodied Environment.
Prior to this step, you have already taken {step_count} step(s).
Below is the most recent observation and the corresponding action you
took:
{action_history}

You are now at step {current_step} and your current observation is:
{current_observation}

Your admissible actions of the current situation are:
[{admissible_actions}].

Return exactly one action wrapped in <action> </action> tags.
Do not include reasoning, commentary, or any other text.
```

Exploration Objective:

```
Your task is to: explore the environment as thoroughly as possible.
Visit as many receptacles as you can, open every openable receptacle to
reveal its contents, and take, examine, cool/heat/clean, or turn on as
many distinct objects as possible to reach novel states. There is no
single goal object; maximize the number of distinct receptacles and
object interactions you touch.
```

Summary Baseline:

```
You are a memory summarizer for an autonomous exploration agent.
Summarize only the new experience represented by the transition below.
Focus on the factual relationship between the action and its observed
outcome.
Do not mention tasks, goals, rewards, hidden state, or prior memory.

Current observation:
{current_observation}

Action taken:
{action}

Resulting observation:
{next_observation}

Return one concise factual summary of one or two sentences wrapped in <
summary> </summary> tags.
Do not include reasoning, commentary, or any other text.
```

C.3.2 Downstream Task Prompt

First Step:

```
You are an expert agent operating in the ALFRED Embodied Environment.
Your current observation is: {current_observation}

Return exactly one action wrapped in <action> </action> tags.
Do not include reasoning, commentary, or any other text.
```

With History:

```
You are an expert agent operating in the ALFRED Embodied Environment.
Prior to this step, you have already taken {step_count} step(s).
Below is the most recent observation and the corresponding action you
took:
{action_history}

You are now at step {current_step} and your current observation is:
{current_observation}

Return exactly one action wrapped in <action> </action> tags.
Do not include reasoning, commentary, or any other text.
```

Goal Suffix:

```
{current_observation}
Your task is to: {goal}
```