FEP-Agent: Active-Inference-Guided LLM Agent Self-Evolution with Semantic Memory

Minghao Chen[1,2], **Xinyi Hu**[1], **Zhou Yu**[1,*], **Yufei Yin**[1]
[1] Zhejiang Key Laboratory of Space Information Sensing and Transmission,
School of Computer Science, Hangzhou Dianzi University, China
[2]Hangzhou Zikirobo Embodied Technology Co., Ltd.
minghaochen01@gmail.com

Abstract

Enabling language model agents to autonomously adapt to new environments remains a fundamental challenge. While recent memory-augmented approaches have achieved promising results, they are largely heuristic in design and treat task execution and knowledge construction as decoupled processes. Inspired by how biological systems rapidly adapt through principled acting and learning, we propose **FEP-Agent**, a framework that guides LLM agent self-evolution with the *Free Energy Principle (FEP)*. Our system instantiates the two components of Active Inference into: a Worker agent that approximately minimizes *Expected Free Energy (EFE)* during online interaction, balancing goal-directed exploitation with curiosity-driven exploration to probe uncertain environmental dynamics; and a Builder agent whose post-execution updates are guided by *Variational Free Energy (VFE)*, updating a knowledge base while controlling complexity through counterfactual validation. To support scalable knowledge in open-ended environments, we introduce a structured semantic memory that employs associative linking and overlapping communities for efficient retrieval. The excellent results across multiple LLM-agent benchmarks show that FEP-Agent supports effective self-evolution.

1 Introduction

Large Language Model (LLM)-based agents have demonstrated remarkable potential across diverse domains, from web navigation [21, 37] and software engineering [16, 30] to embodied control [1, 40]. However, when deployed in unfamiliar environments or domains, they typically require human experts to design domain-knowledge-rich workflows and provide careful prompting [47, 45]. Alternatively, they may require extensive fine-tuning on large task-specific datasets [49, 29], a process that is both data-hungry and computationally prohibitive. This raises a crucial question: *Can agents rapidly adapt through experience, as biological systems do?*

Recent works have explored explicit agent memory systems to achieve self-evolving agents without parameter updates. One line of research [55, 41, 44] stores successful trajectories as reusable skills, serving as in-context examples to guide subsequent tasks. Other studies [51, 11, 3] further extract structured rules or reasoning patterns [26] from experience, providing more abstract guidance for future decisions. Despite these advances, existing approaches suffer from two critical limitations. *First*, they are largely **heuristic in design**, lacking a unifying theoretical framework to guide how knowledge should be acquired, updated, and utilized. *Second*, and more fundamentally, these methods treat task execution (trajectory rollout) and knowledge building as **decoupled processes**. Consequently, the agent remains a passive accumulator of experience rather than an actively exploring learner that resolves ambiguities through acting.

*Corresponding Author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Biological organisms offer a compelling counterexample. Based on the well-known Free Energy Principle (FEP) [8] theory in cognitive neuroscience, animals employ a principled inference process grounded in Active Inference [9] when encountering unfamiliar environments. Under this framework, agents maintain an internal generative model of the world and continuously refine it by minimizing prediction error (*perception & learning*) while actively seeking information to reduce uncertainty (*exploration*). This dual process enables rapid adaptation with minimal experience, suggesting a promising paradigm for artificial agents.

In this work, we draw inspiration from Active Inference to guide LLM agent design. We propose **FEP-Agent**, an Active-Inference-guided self-evolving agent system where both online decision-making and memory construction follow operational surrogates of free-energy objectives. The system operates through two complementary components, each optimizing a distinct aspect of free energy. A Worker agent approximates *Expected Free Energy* (EFE) minimization during online interaction. It predicts future outcomes for candidate policies and selects actions that balance pragmatic value (exploiting known knowledge to achieve goals) with epistemic value (exploring uncertain mechanisms to reduce model ambiguity). After execution, a Builder agent approximates *Variational Free Energy* (VFE) minimization by updating the knowledge base to better explain observed trajectories, while using counterfactual validation to control model complexity.

At its core, FEP-Agent maintains a structured semantic memory $\mathcal{K}$ that serves as the learnable parameters of its generative model, enabling adaptation without modifying LLM weights. Drawing from cognitive neuroscience, we organize $\mathcal{K}$ with these structural priors: Unlike traditional triplet-based knowledge graphs [15, 50], our structure allows nodes to contain rich natural language descriptions with Wikipedia-style references (e.g., `[[Entity Name]]`) [20], capturing complex, non-binary relationships. These references form a semantic graph backbone and, after further overlapping community detection, enable efficient navigation of the growing $\mathcal{K}$.

Our main contributions are as follows:

- Inspired by Active Inference, we propose **FEP-Agent**, a framework that unifies exploration, exploitation, and learning under a common objective decomposition.
- We derive operational surrogates of the dual Active Inference objectives (EFE/VFE) and instantiate them into the Worker and the Builder agents. To enable a scalable yet flexible memory for open-ended environments, we further design a structured semantic memory for agents.
- Experiments on web automation, mobile use, and embodied planning benchmarks demonstrate that FEP-Agent achieves state-of-the-art performance, validating the effectiveness of our approach to LLM agent self-evolution.

2 Related Works

2.1 Self-Evolving LLM Agent

Standard LLM-based agents leverage the emergent reasoning capabilities of large language models [25, 24, 5], combined with handcrafted prompting or workflows [47, 34, 21, 40], to conduct multi-turn interactions with environments for complex tasks. However, such paradigms demand domain expertise or fine-tuning on human-annotated trajectories [49], which motivates *self-evolving agents* that autonomously adapt to novel environments. There are two primary research directions:

Parameter-based Adaptation. This paradigm finetunes LLM through preference or reinforcement learning on self-generated trajectories. AgentQ [29] and UI-TARS [31] employ MCTS combined with Direct Preference Optimization (DPO) for GUI automation. More recent works adopt end-to-end RL: DeepResearcher [53] and Search-R1 [19] train agents through RL to acquire multi-round search capabilities, while UI-TARS-2 [42] extends this to GUI domains. However, these approaches demand substantial resources and data collection, struggling to achieve rapid adaptation.

Memory-based Adaptation. An alternative paradigm maintains explicit textual memory without modifying model parameters. A prevalent strategy [38, 44] stores successful task trajectories in a *skill library* for in-context retrieval. GITM [55] and Voyager [41] pioneered this approach in Minecraft. Furthermore, ExpeL [51] and AutoManual [3] extract higher-level *rules* or *insights* from both successful and failed experiences, providing more transferable guidance. ReasoningBank [26] similarly distills high-level reasoning patterns. However, these methods treat trajectory collection as

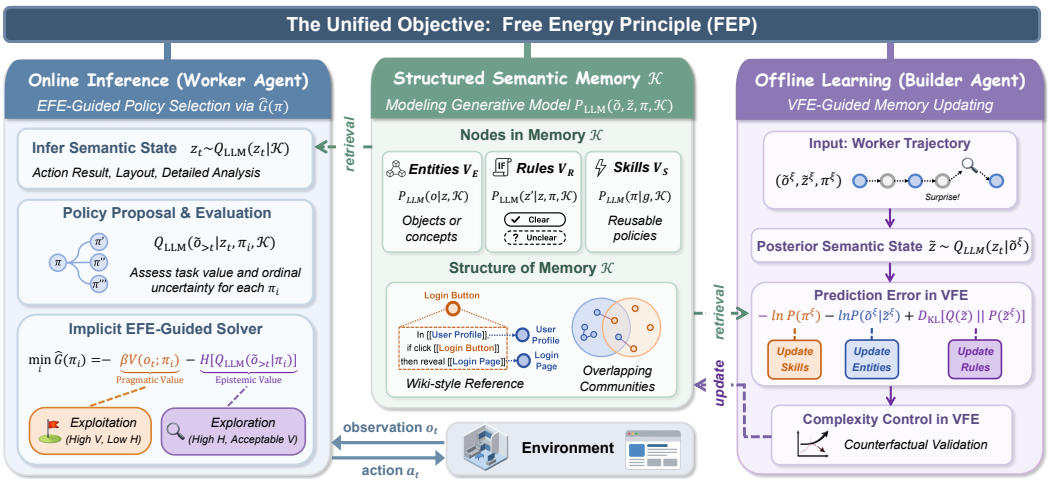


Figure 1: **Overview:** FEP-Agent instantiates the classic Active Inference loop into two distinct timescales handled by two LLM agents: (1) *Online*: the Worker approximately minimizes EFE for action selection, balancing goal-directed exploitation with curiosity-driven exploration; (2) *Offline*: the Builder uses a VFE-guided objective for knowledge updates. Both agents share a Semantic Memory $\mathcal{K}$ that serves as the learnable component of the generative model.

a *passive* process. Our work instead is guided by the Active Inference theory and allows the agent to probe uncertain mechanisms during rollout, bridging execution and learning.

2.2 Active Inference Agent

Active Inference [9], derived from the Free Energy Principle [8], provides a unified Bayesian framework explaining how biological agents simultaneously perceive, learn, and act by minimizing free energy. Several works have translated this theoretical framework into practical algorithms for simple environments: The foundational work by Friston et al. [9] established discrete-state implementations, while subsequent research extended to continuous domains through deep neural networks [7]. Other studies [39, 33] connect Active Inference to reinforcement learning, naturally balancing exploration and exploitation. Our work instantiates Active Inference within LLM agents, in which the language model with semantic memory serves as the generative model, enabling effective self-evolution for complex, realistic environments.

3 Methods

In this section, we first present the theoretical formalism that motivates FEP-Agent. We then introduce FEP-Agent, a framework that instantiates these principles through a dual-agent architecture with a semantic memory, as shown in Fig. 1.

3.1 Preliminaries: Active Inference

Following the formalism of Active Inference [9], we formalize the environment as a Partially Observable Markov Decision Process (POMDP). The generative process (the true environmental dynamics) is defined by the tuple $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, r \rangle$, where $\mathcal{S}$ is the true hidden states of the environment, $\mathcal{O}$ is the observation space, and $\mathcal{A}$ is the action space. $\mathcal{T}$ denotes transition dynamics $P(s_{t+1}|s_t, a_t)$ and observation likelihoods $P(o_t|s_t)$, and r is the reward function under the user goal g.

Since the agent does not have access to the true states $\mathcal{S}$, it maintains an internal **Generative Model** P to explain the observations. Notably, while original literature [9] often reuses the symbol s for internal states, we distinguish the agent's internal representation as $z \in \mathcal{Z}$, acknowledging that the semantic abstraction $\mathcal{Z}$ by the agent may differ structurally from the true state space $\mathcal{S}$. Then the generative model is a joint probability distribution $P(\tilde{o}, \tilde{z}, \pi, \eta)$ over observation sequence $\tilde{o} = (o_1, ..., o_t)$, internal states $\tilde{z}$, policies π (where $\pi(t) = a_t$), and model parameters η.

Under the Free Energy Principle (FEP) [8], the agent is a self-evidencing system minimizing *surprise* (negative log-evidence): $-\ln P(\tilde{o})$. Since direct minimization is intractable, the agent minimizes an upper bound known as **Variational Free Energy (VFE)**, denoted as F:

$$F = \underbrace{-\mathbb{E}_Q[\ln P(\tilde{o}|\tilde{z},\pi,\eta)]}_{\text{Prediction Error}} + \underbrace{D_{\mathrm{KL}}\left[Q(\tilde{z},\pi,\eta|\tilde{o}) \,\|\, P(\tilde{z},\pi,\eta)\right]}_{\text{Complexity}}, \tag{1}$$

where $Q(\tilde{z},\pi,\eta|\tilde{o})$ is an approximate posterior of the intractable true posterior $P(\tilde{z},\pi,\eta|\tilde{o})$. Given past observations, minimizing F serves perception and learning of the agent: optimize internal beliefs $\tilde{z}$ to better explain experiences and model structure η to update the world model.

While VFE minimizes surprise for *observed* data, planning requires evaluating *future*, unobserved outcomes under a policy π. For this, the agent minimizes the **Expected Free Energy (EFE)**, denoted as $G(\pi)$. EFE is the expected VFE over predicted future trajectories, conditioned on a policy:

$$G(\pi,\tau) \approx -\underbrace{\mathbb{E}_{Q(o_\tau|\pi)}[\ln P(o_\tau|g)]}_{\text{Pragmatic Value (Instrumental)}} - \underbrace{\mathbb{E}_{Q(o_\tau|\pi)}\left[D_{\mathrm{KL}}\left[Q(z_\tau|o_\tau,\pi) \,\|\, Q(z_\tau|\pi)\right]\right]}_{\text{Epistemic Value (Information Gain)}}, \tag{2}$$

where η and z_t are omitted from conditions for simplicity, and minimize $G(\pi) = \sum_{\tau>t} G(\pi,\tau)$ to create a policy that balances goal-directed and information-seeking behavior.

3.2 Structured Semantic Memory

Semantic Approximations. Guided by this formalism, we first interpret the latent state z_t not as a numeric vector, but as a *Semantic State*: the detailed textual understanding of the LLM agent at step t, and $\mathcal{Z}$ is the semantic space expressible by the agent. We then decouple $\eta = \theta_{\text{LLM}} \cup \mathcal{K}$, where θ_{LLM} denotes the frozen weights of the backbone LLM, providing general reasoning and world knowledge; $\mathcal{K}$ is an evolving semantic memory that captures the specific knowledge of the environment. Consequently, the frozen LLM acts as the computational engine for the generative model. We denote the LLM-instantiated distributions as P_{LLM} and Q_{LLM}. Crucially, since explicitly calculating or sampling probability densities in the high-dimensional semantic space $\mathcal{Z}$ is intractable, we do not perform exact numerical operations. Instead, we leverage the in-context reasoning capabilities of the LLM to implicitly estimate conditional probabilities: e.g., sampling from $Q_{LLM}(z|o,\mathcal{K})$ becomes generating plausible state descriptions given o and $\mathcal{K}$ in context.

To systematically model the environment, we note that the generative model can be factorized as:

$$P_{\text{LLM}}(\tilde{o},\tilde{z},\pi,\mathcal{K}) = P_{\text{LLM}}(\mathcal{K})P_{\text{LLM}}(\pi|g,\mathcal{K})\prod_{t=1}^{T}\Big[P_{\text{LLM}}(o_t|z_t,\mathcal{K})P_{\text{LLM}}(z_t|z_{t-1},a_{t-1},\mathcal{K})\Big] \tag{3}$$

We decompose $\mathcal{K}$ into three distinct categories, each corresponding to a component of the generative model, which aligns with empirical designs in recent LLM agent works [3, 43].:

- **Entities ($\mathcal{V}_E$):** Extract valid objects or concepts in the environment, grounding raw observations into semantic states. These expand the state space $\mathcal{Z}$, and refine the likelihood mapping $P_{LLM}(o|z,\mathcal{K})$, which can boost posterior inference $Q_{LLM}(z|o,\mathcal{K})$.
- **Rules ($\mathcal{V}_R$):** Capture the internal transition dynamics $P_{LLM}(z'|z,a,\mathcal{K})$. Rules are further classified into *Clear* (high confidence) or *Unclear* (low confidence), reflecting the entropy of the transition.
- **Skills ($\mathcal{V}_S$):** Record successful trajectories for user goals, updating the policy prior $P_{LLM}(\pi|g,\mathcal{K})$.

Active Inference emphasizes that the structure of a generative model (not just its parameters) embodies crucial prior beliefs about the world [10]. In hierarchical generative models [4], higher levels encode abstract, slowly-changing regularities, while lower levels capture specific, rapidly-changing details. Motivated by this principle, we organize $\mathcal{K}$ through two structural mechanisms:

Associative Linking via Wiki-Style References. Neuroscience research [28, 2] suggests that human semantic knowledge is organized through *associative networks*, where concepts are linked by flexible, content-addressable associations rather than fixed schema. Inspired by this, we let each node of $\mathcal{V}$ ($= \mathcal{V}_E \cup \mathcal{V}_R \cup \mathcal{V}_S$) contain a natural-language description, in which it can reference other nodes using Wikipedia-style citations (syntax `[[Node Name]]`) [20]. It forms a flexible citation graph that captures non-binary dependencies without forcing all knowledge into rigid triplets [15, 14].

Hierarchical Structure via Overlapping Communities. To guide efficient retrieval and to induce high-level abstractions (as structural priors), we periodically cluster $\mathcal{K}$. While existing methods [15,

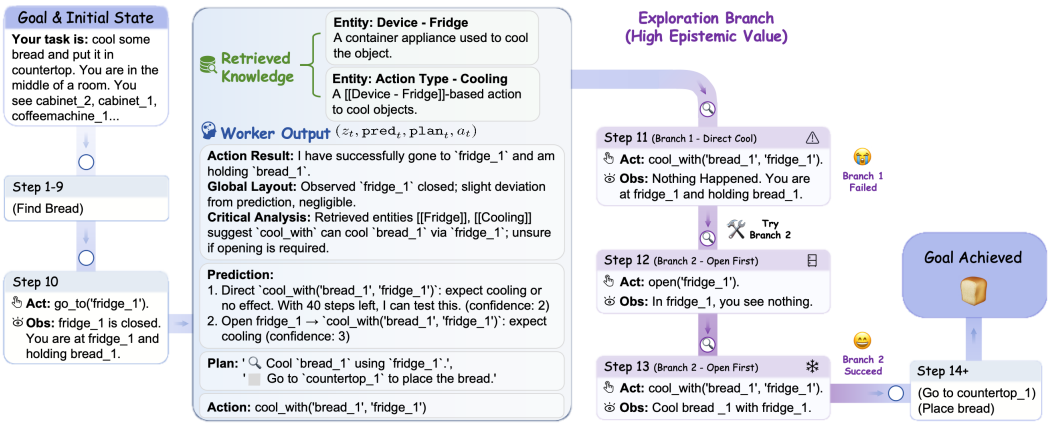


Figure 2: An illustration of the Worker trajectory in ALFWorld. At each step, the Worker outputs $(z_t, \texttt{pred}_t, \texttt{plan}_t, a_t)$. Under the EFE-guided policy objective, the Worker might trigger an exploration branch for uncertain mechanisms.

50] often enforce mutually exclusive clusters, neuroscience evidence that semantic concepts are represented through *overlapping cortical networks* [17, 23]: a single concept simultaneously activates multiple brain regions encoding different features. Therefore, we allow communities to be multi-faceted (e.g., "User Comments" may belong to both "User Profile" and "Forum Controls").

We inject current communities into the context of all agents as a coarse-grained "cognitive map" of $\mathcal{K}$, allowing agents to localize relevant concepts before retrieving specific nodes. More details of node attributes, reference graph, community detection, and retrieval are provided in Appendix D.

3.3 Worker Agent: EFE-Guided Policy Selection

Lemma 3.1. *Under the Boltzmann distribution assumption $P(o_\tau \mid g) \propto \exp(\beta r(o_\tau))$, the summed per-step EFE in Eq. 2 is upper-bounded by the following rollout-level surrogate:*

$$G(\pi) \leq \widehat{G}(\pi) := \underbrace{-\beta\, V(o_t;\pi)}_{\textit{Pragmatic Value}} \underbrace{- H[Q_{\text{LLM}}(\tilde{o}_{>t} \mid z_t, \pi, \mathcal{K})]}_{\textit{Epistemic Value}}, \tag{4}$$

where $\tilde{o}_{>t}$ denotes the predicted future rollout, $H(\cdot)$ is Shannon entropy and $V(o_t;\pi) = \mathbb{E}_{Q_{\text{LLM}}(\tilde{o}_{>t}|\pi)}[\sum_{\tau=t+1}^{T} r(o_\tau)]$ is the expected task value. (Derivation is provided in Appendix C.)

The Worker acts as the online inference engine, interacting with the environment in a ReAct-style loop [47]. In our setting, the reward signal is episodic and binary (i.e., success or failure within a step limit). Therefore, the pragmatic value equals the probability of success from the current state: $V(o_t;\pi) = P\left(\text{success} = 1 \mid z_t, g, \pi\right)$. At each step t, the Worker outputs $(z_t, \texttt{pred}_t, \texttt{plan}_t, a_t)$ to find the optimal policy $\arg\min_\pi \widehat{G}(\pi)$, as illustrated in Fig. 2.

State Inference. Given the observation o_t (e.g., HTML, screenshots), the Worker first infers the semantic state $z_t \sim Q_{LLM}(z_t|\tilde{o}_{\leq t}, \mathcal{K})$. This posterior inference is a grounded interpretation structured into three components: (1) *Action Result:* Whether the previous action achieved its intended effect. (2) *Global Layout:* A concise summary of the current location and layout. (3) *Critical Analysis:* Detailed description of task-relevant elements (including their types and states). Crucially, the Worker explicitly cites knowledge that helps its analysis (e.g., "click `[[Login Button]]` at the bottom to login"). It also checks whether o_t matches the prior prediction. "Surprise" (high prediction error) signals the need to adjust prior plans and for the subsequent builder to update rules.

Policy Proposal. Since the combinatorial explosion of the action sequences, directly solving $\arg\min_\pi \widehat{G}(\pi)$ over all policies is intractable in realistic environments. Instead, the Worker samples plausible semantic candidate policies based on current knowledge: $\{\pi_i\}_{i=1}^m \sim Q_{\text{LLM}}(\pi \mid z_t, g, \mathcal{K})$. To evaluate these candidates, we initially explored an *Explicit Solver*: the Worker enumerates possible rollouts $\{\tilde{o}_{>t}^j\}_{j=1}^{n_i}$, assigns probabilities $q_{ij} = Q_{\text{LLM}}(\tilde{o}_{>t}^j \mid \pi_i)$ and estimates the task success probability $\hat{V}_i$ to compute $\widehat{G}(\pi_i) = -\beta\hat{V}_i + \sum_j q_{ij} \ln q_{ij}$ and select policy by $\arg\min_i \widehat{G}(\pi_i)$.

Although this solver is faithful to the operational EFE objective, we find it unreliable in practice: LLMs struggle to calibrate probabilities accurately, leading to cascading estimation errors.

Implicit EFE-Guided Solver. For each candidate policy π_i, it generates $\texttt{pred}_t$ to describe the expected rollouts alongside a natural language confidence level $[1, 5]$. This provides an ordinal proxy for the predictive entropy $H[Q_{\mathrm{LLM}}(\tilde{o}_{>t} \mid \pi_i)]$ without requiring exact numerical computation. To generate the next $\texttt{plan}_t$ and a_t, the Worker is prompted to qualitatively approximate $\arg\min_i \widehat{G}(\pi_i)$ by balancing V and H across candidate policies:

Exploitation (High V, Low H): If the retrieved $\mathcal{K}$ identifies clear rules or skills linking the current state to the goal, the outcome is highly certain (low $\widehat{H}_i$) and the expected success probability is high (high $\widehat{V}_i$). The Worker chooses the policy to exploit known dynamics (marked with $\square$ in $\texttt{plan}_t$).

Exploration (High H, Acceptable V): If a candidate policy relies on unclear rules, unsupported assumptions, or low-confidence predictions, the Worker treats it as having high predictive entropy $\widehat{H}_i$. Such uncertainty can have epistemic value: executing the policy may reveal the true mechanism of the environment and reduce future ambiguity. However, the Worker only selects this exploratory policy when its value remains acceptable, considering the remaining step budget, whether the action is reversible, and whether failure to verify the mechanism would endanger task completion. These exploratory branches are marked with 🔍 in $\texttt{plan}_t$, which is analogous to branched rollouts in model-based reinforcement learning [18].

3.4 Builder Agent: VFE-Guided Memory Update

Lemma 3.2. *For a completed trajectory $\xi = (\tilde{o}^\xi, \tilde{z}^\xi, \pi^\xi, g^\xi)$, substituting the factorized generative model (Eq. 3) into VFE (Eq. 1) yields the following minibatch objective:*

$$F_B(\xi, \mathcal{K}) \approx \lambda|\mathcal{K}|_{\mathrm{tokens}} - \ln P_{\mathrm{LLM}}(\pi^\xi \mid g^\xi, \mathcal{K}) - \sum_{t=0}^{T} \ln P_{\mathrm{LLM}}(o_t^\xi \mid z_t^\xi, \mathcal{K}) + \sum_{t=1}^{T} D_{\mathrm{KL}}\Big(Q_{\mathrm{LLM}}(z_t \mid \tilde{o}^\xi, \pi^\xi, \mathcal{K}) \,\Big\|\, P_{\mathrm{LLM}}(z_t \mid z_{t-1}^\xi, a_{t-1}^\xi, \mathcal{K})\Big), \tag{5}$$

where the knowledge prior is modeled with a exponential form $-\ln P(\mathcal{K}) \approx \lambda|\mathcal{K}|_{\textit{tokens}}$, which gives a minimum-description-length style complexity penalty on the memory size.

The Builder uses the mini-batch VFE F_B as a guiding objective for an epoch trajectory ξ. This process updates $\mathcal{K}$ to better explain the observed reality. Since each state z_t inferred by the Worker is conditioned only on past observations $\tilde{o}_{\leq t}$, we first let the Builder generate posterior semantic state with hindsight: identify and clarify the misunderstanding in each z_t given the full trajectory $\tilde{o}^\xi$, which instantiates $Q_{\mathrm{LLM}}(z_t \mid \tilde{o}^\xi, \pi^\xi, \mathcal{K})$ in Eq. 5.

Reducing Prediction Error. The update design for $\mathcal{K}$ follows the VFE decomposition (Eq. 5): (1) The Builder updates entities to improve $\ln P_{\mathrm{LLM}}(o_t^\xi \mid z_t^\xi, \mathcal{K})$, i.e., allow the semantic state to better explain the observation; (2) The Builder scans the trajectory for moments of high *Surprise* where the prior prediction of the Worker diverged from the posterior hindsight. Then it rectifies this by adding new rules or correcting erroneous rules. Additionally, the Builder promotes validated *Unclear Rules* to *Clear Rules*, sharpening the transition belief. (3) The Builder updates skills to improve $P_{\mathrm{LLM}}(\pi^\xi \mid g^\xi, \mathcal{K})$, i.e., concentrate the policy prior for future tasks.

Complexity Control. The prior penalty $\lambda|\mathcal{K}|_{\mathrm{tokens}}$ in Eq. 5 prevents unconstrained memory growth. To operationalize this, before committing any update $\Delta\mathcal{K}$, we seek updates that satisfy

$$\Delta F_B(\xi, \Delta\mathcal{K}) = F_B(\xi, \mathcal{K} \cup \Delta\mathcal{K}) - F_B(\xi, \mathcal{K}) < 0. \tag{6}$$

Specifically, the Builder is prompted to perform **Counterfactual Validation** filtering: before any proposed updates $\Delta\mathcal{K}$, it evaluates: *"If $\Delta\mathcal{K}$ had been known, would the surprise (prediction error) at corresponding steps have been significantly reduced?"*. This filtering retains updates judged to reduce prediction error enough to justify added complexity.

Table 1: Experiment results on WebArena benchmark. Success rates are reported on 5 subsets for two different backbone LLMs.

Methods	Reddit	Shopping	Admin	Gitlab	Multi	Overall
Gemini-2.5-flash						
ReAct [47]	55.7	39.0	44.5	33.9	10.3	40.5
Synapse [52]	59.4	40.6	45.1	35.6	10.3	42.1
AWM [44]	62.3	44.4	46.7	37.2	3.4	44.1
AutoManual [3]	65.8	47.1	48.9	38.9	13.8	46.9
ReasoningBank [26]	67.0	49.7	51.1	40.6	13.8	48.8
FEP-Agent	**71.7**	**52.9**	**54.4**	**46.7**	**17.2**	**53.1**
Gemini-2.5-pro						
ReAct [47]	71.7	45.5	51.1	35.0	6.9	46.7
Synapse [52]	68.9	46.5	52.2	38.3	6.9	47.7
AWM [44]	68.9	48.1	49.3	40.0	3.4	47.6
AutoManual [3]	77.4	52.4	54.9	50.0	13.8	54.7
ReasoningBank [26]	80.2	51.9	56.6	44.4	13.8	53.9
FEP-Agent	**83.2**	**58.7**	**58.8**	**66.1**	**20.7**	**62.9**

4 Experiments

4.1 Experiment Setup

Benchmarks. We evaluate FEP-Agent on three representative LLM agent benchmarks spanning web navigation, mobile device control, and embodied planning: (1) **WebArena** [54]: A realistic web automation benchmark featuring self-hosted websites that simulate real-world applications. Following ReasoningBank [26], we conduct experiments on 684 task instances across 5 domains: Shopping (187), Admin (182), GitLab (180), Reddit (106), and Multi-site (29). The agent receives the accessibility tree of the current webpage as the observation. (2) **AndroidWorld** [32]: A realistic mobile-use benchmark featuring 116 tasks across 20 real-world Android applications (productivity, communication, utilities, etc.). AndroidWorld provides the screenshot and the accessibility tree for observations. The action space mimics human interactions, such as tapping, long-pressing, and swiping. (3) **ALFWorld** [35]: A text-based embodied planning benchmark that bridges symbolic reasoning and visual grounding. We evaluate on 135 test tasks spanning six distinct task types.

Baselines. We compare FEP-Agent against representative methods across different paradigms: (1) *Non-evolving*: ReAct [47], which uses interleaved reasoning and acting without any self-evolving mechanism. (2) *Exemplar-based*: Synapse [52], AdaPlanner [38], and AWM [44]. (3) *Rule/Insight-based*: ExpeL [51], AutoManual [3], ReasoningBank [26], and G-Memory [50]. (4) *Domain-specific Agents*: M3A [32], Agent S3 [12], MobileUse [22], Gemini 2.5 Computer Use [6], Mobile-Agent-v3 [48], and UI-TARS-2 [42].

Implementation Details. We build the agents on Gemini-2.5 [5]. For ALFWorld, we additionally evaluate with GPT-4o-mini as the backbone to enable fair comparison with prior methods. We adopt the average success rate (SR) as the primary evaluation metric. More implementation details and prompts of our agents are provided in Appendix E.

4.2 Main Results

Results on WebArena. As shown in Tab. 1, FEP-Agent significantly outperforms the existing methods, evidenced by overall success rates of 53.1% with Gemini-2.5-flash and 62.9% with Gemini-2.5-pro. Prior self-evolving LLM agents, including exemplar-based (Synapse, AWM) and insight-based (AutoManual, ReasoningBank) methods, have surpassed the ReAct baseline by leveraging textual memory. However, these approaches suffer from passive rollout strategies and suboptimal knowledge management. In contrast, FEP-Agent delivers 8.2% absolute improvement over AutoManual (54.7%), especially in the GitLab domain. We attribute this gap to our EFE-guided exploration: when the Worker encounters ambiguous UI elements or unclear navigation paths, the entropy term actively drives hypothesis testing rather than relying solely on cached experiences.

Table 2: Success rate (%) of LLM agent methods on 135 test tasks of ALFWorld.

Methods	Put	Clean	Heat	Cool	Examine	Put two	ALL
GPT-4o-mini							
ReAct [47]	79.2	25.8	39.1	40.9	55.6	35.3	45.2
ExpeL [51]	66.7	64.5	39.1	63.6	61.1	35.3	56.3
AdaPlanner [38]	87.5	48.4	69.6	72.7	66.7	41.2	64.4
AutoManual [3]	95.8	80.6	87.0	77.3	**100.0**	76.5	85.9
G-Memory [50]	95.8	90.3	86.9	**86.4**	83.3	**88.2**	88.8
FEP-Agent	**100.0**	**93.5**	**91.3**	85.7	83.3	**88.2**	**91.1**
Gemini-2.5-flash							
ReAct [47]	95.8	87.1	87.0	90.9	88.9	70.6	87.4
ExpeL [51]	95.8	90.3	91.3	95.5	94.4	70.6	90.4
AdaPlanner [38]	91.7	90.3	87.0	90.9	83.3	88.2	88.9
AutoManual [3]	**100.0**	96.8	**100.0**	95.5	**100.0**	88.2	97.0
G-Memory [50]	**100.0**	96.7	95.6	**100.0**	**100.0**	94.1	97.7
FEP-Agent	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**

Table 3: Success rate (%) of different methods on AndroidWorld.

Methods	SR (%)
Domain-specific models	
MobileUse [22]	62.9
Gemini 2.5 Computer Use [6]	69.7
Mobile-Agent-v3 [48]	73.3
UI-TARS-2 [42]	73.3
Gemini-2.5-pro	
M3A [32]	59.7
ReAct [47]	65.9
Agent S3 [12]	68.1
FEP-Agent	**78.4**

Table 4: Ablation study on WebArena (Gemini-2.5-flash). We systematically remove components to validate their contribution.

Methods	SR (%)	Δ
FEP-Agent (Full)	**53.1**	–
ReAct (no self-evolving)	40.5	-12.6
FEP-Agent$_{\text{explicit}}$	48.2	-4.9
Active Inference Components		
w/o Epistemic Value in EFE	49.8	-3.3
w/o Complex Control in VFE	51.5	-1.6
Memory Structure		
Associative Links → Triplet KG	51.0	-2.1
Overlapping Comm. → Leiden	52.7	-0.4

Results on AndroidWorld. As shown in Table 3, FEP-Agent achieves 78.4% success rate, outperforming even domain-specific models fine-tuned for mobile automation. Compared to Gemini 2.5 Computer use, which is built upon Gemini-2.5-pro for GUI interaction, our method achieves +8.7% improvement using only a general-purpose LLM with our semantic memory. These improvements demonstrate that self-evolving capabilities are crucial for addressing the combinatorial complexity of real-world mobile environments. This is because each real-world scenario contains unique knowledge that cannot be fully encompassed by large-scale pre-training.

Results on ALFWorld. Tab. 2 presents results on the embodied planning benchmark. With GPT-4o-mini, FEP-Agent achieves 91.1% success rate, outperforming G-Memory (88.8%) and AutoManual (85.9%). Remarkably, with Gemini-2.5-flash, FEP-Agent achieves 100% success rate across all six task types, especially for the challenging *Put Two* category. While G-Memory designed a multi-agent system with self-evolution mechanisms for ALFWorld, FEP-Agent demonstrates that a streamlined worker agent coupled with efficient memory management is sufficient to yield better results.

We provide the token cost during the building and testing of the FEP-Agent in Appendix E.

4.3 Ablation Studies

To validate the contribution of each component in FEP-Agent, we conduct systematic ablation experiments with Gemini-2.5-flash on WebArena in Tab. 4.

Active Inference Components. The EFE and VFE mechanisms contribute significantly to overall performance. As shown in Tab. 4, eliminating the prompt instructing the Worker to seek cognitive value (Sec. 3.3) results in a 3.3% decrease in performance. It is because the Worker agent loses the ability to actively probe uncertain mechanisms, and it becomes a passive accumulator that only exploits cached knowledge, analogous to prior self-evolving methods. Similarly, disabling complexity control in the Builder agent (Sec. 3.4) leads to a larger knowledge base cluttered with

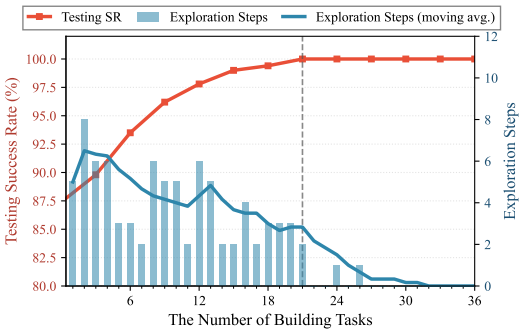


Figure 3: During the building process on ALFWorld, the average success rate on test tasks and the number of exploration steps per task over time.

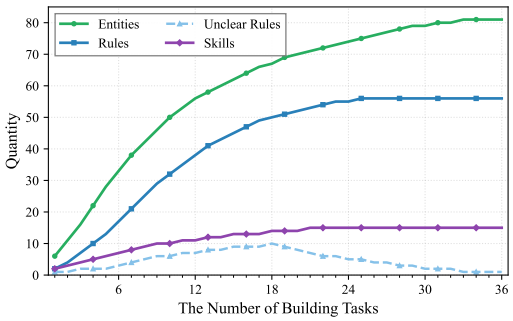


Figure 4: During the building process on ALFWorld, the growth dynamics of the knowledge base components, including Entities, Skills, Rules, and Unclear Rules.

spurious rules; without counterfactual validation filtering, retrieval noise impairs decision-making (-1.6%). These results support the effectiveness of the Active-Inference-guided design for self-evolution. Furthermore, we demonstrate that the performance of FEP-Agent$_{\text{explicit}}$, the Explicit Solver mentioned in Sec. 3.3, is inferior to that of the currently employed Implicit EFE-Guided Solver. This is because LLMs without specific fine-tuning struggle with accurate numerical estimation, whereas the Implicit EFE-Guided Solver circumvents this issue.

Memory Structure. As shown in Tab. 4, replacing Wiki-style references with traditional triplet-based KG as in GraphRAG [15] reduces performance by 2.1%. This is because triplets struggle to represent complex, multi-entity relationships that are naturally expressed via our associative links (e.g., "clicking `[[Button A]]` after filling `[[Form B]]` triggers `[[Popup C]]`"). We also replace our overlapping community detection with the Leiden algorithm in GraphRAG [15], which enforces mutually exclusive partitions and directly generates community summaries. This causes a modest 0.4% drop, as some multi-faceted concepts (e.g., a "Submit button" belonging to both "Form Filling" and "Navigation" contexts) lose their cross-community associations.

4.4 Analysis

Here, we analyze the learning dynamics of FEP-Agent on ALFWorld with Gemini-2.5-flash to validate that our framework produces the expected emergent behaviors.

Exploration-Exploitation Transition. Fig. 3 visualizes how the agent naturally shifts from information-seeking to goal-directed behavior. During the early stages of knowledge base construction, the Worker frequently encounters uncertain dynamics, resulting in exploratory probes driven by epistemic value. As uncertainty is resolved, exploration behaviors decline sharply and vanish after approximately 21st building tasks (grey dashed line). This behavioral shift aligns with improvements in generalization capability: the testing success rate rises from 87.4% to 100% by the 21st task. This emergent phenomenon supports our EFE-guided action selection: the agent autonomously balances curiosity and exploitation based on internal uncertainty.

Knowledge Base Dynamics. Fig. 4 illustrates the growth of $\mathcal{K}$ components. As building tasks progress, the number of entities grows rapidly, gradually reaching saturation, representing a vocabulary encompassing all important objects or concepts in the environment. Skills accumulate gradually through successful task completion and approach saturation after all types of tasks have been successfully completed. The Rule dynamics exhibit a "rise-then-refine" pattern: the total number of rules gradually reaches saturation, while the number of "unclear rules" peaks and then steadily declines as exploratory probes validate hypotheses and the builder updates them to clear rules (or discards invalid rules). This demonstrates the role of the 'unclear' mechanism in exploration and complexity control.

5 Conclusion

In this paper, we introduced FEP-Agent, a framework inspired by Active Inference to guide LLM agent self-evolution, unifying execution and learning under free-energy-motivated objectives. FEP-Agent dynamically balances epistemic exploration with goal-directed exploitation through an EFE surrogate and refines its semantic memory through a VFE-guided update criterion, overcoming the limitations of heuristic memory accumulation. Equipped with the scalable Semantic Memory, our framework exhibits outstanding performance across complex, realistic environments.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grants (No. 62502135, 62422204), the Zhejiang Provincial Natural Science Foundation of China under Grant (No. LQN25F030014, LRG26F020001), the Key Research and Development Program of Zhejiang Province (No. 2025C01026), and the Scientific Research Innovation Capability Support Project for Young Faculty. The authors declare that they have no competing interests.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, and et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

[2] Jeffrey R. Binder, Rutvik H. Desai, William W. Graves, and Lisa L. Conant. Where is the semantic system? a critical review and meta-analysis of 120 functional neuroimaging studies. *Cerebral cortex*, 19 12:2767–96, 2009.

[3] Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning. *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024.

[4] Andy Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *The Behavioral and brain sciences*, 36 3:181–204, 2013.

[5] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, and et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *ArXiv*, abs/2507.06261, 2025.

[6] Google DeepMind. Introducing the gemini 2.5 computer use model, Oct 07 2025.

[7] Zafeirios Fountas, Noor Sajid, Pedro A.M. Mediano, and Karl Friston. Deep active inference agents using monte-carlo methods. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[8] Karl J. Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11:127–138, 2010.

[9] Karl J. Friston, Thomas H. B. FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, and Giovanni Pezzulo. Active inference: A process theory. *Neural Computation*, 29:1–49, 2017.

[10] Karl J. Friston, Thomas Parr, and Bert de Vries. The graphical brain: Belief propagation and active inference. *Network Neuroscience*, 1:381 – 414, 2017.

[11] Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[12] Gonzalo Gonzalez-Pumariega, Vincent Tu, Chih-Lun Lee, Jiachen Yang, Ang Li, and Xin Eric Wang. The unreasonable effectiveness of scaling agents for computer use. *ArXiv*, abs/2510.02250, 2025.

[13] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. A survey on llm-as-a-judge. *ArXiv*, abs/2411.15594, 2024.

[14] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. *ArXiv*, abs/2410.05779, 2024.

[15] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh M. Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qianru He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. Retrieval-augmented generation with graphs (graphrag). *ArXiv*, abs/2501.00309, 2024.

[16] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[17] Alexander G. Huth, Wendy A. de Heer, Thomas L. Griffiths, Frédéric E. Theunissen, and Jack L. Gallant. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532:453 – 458, 2016.

[18] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *ArXiv*, abs/1906.08253, 2019.

[19] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025.

[20] Andrej Karpathy. Llm wiki: A pattern for building personal knowledge bases using llms. `https://gist.github.com/karpathy/442a6bf555914893e9891c11519de94f`, 2026.

[21] Geunwoo Kim, Pierre Baldi, and Stephen Marcus McAleer. Language models can solve computer tasks. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[22] Ning Li, Xiangmou Qu, Jiamu Zhou, Jun Wang, Muning Wen, Kounianhua Du, Xingyu Lou, Qiuying Peng, Jun Wang, and Weinan Zhang. Mobileuse: A hierarchical reflection-driven GUI agent for autonomous mobile operation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[23] Alex Martin. The representation of object concepts in the brain. *Annual Review of Psychology*, 58:25–45, 2007.

[24] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[25] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. In *Neural Information Processing Systems (NeurIPS)*, 2022.

[26] Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. *ArXiv*, abs/2509.25140, 2025.

[27] Thomas Parr, Giovanni Pezzulo, and Karl J. Friston. *Active Inference: The Free Energy Principle in Mind, Brain, and Behavior*. The MIT Press, 03 2022.

[28] Karalyn E Patterson, Peter J. Nestor, and Timothy T. Rogers. Where do you know what you know? the representation of semantic knowledge in the human brain. *Nature Reviews Neuroscience*, 8:976–987, 2007.

[29] Pranav Putta, Edmund Mills, Naman Garg, Sumeet Ramesh Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *ArXiv*, abs/2408.07199, 2024.

[30] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, Maosong Sun, and Wei Liu. Communicative agents for software development. *ArXiv*, abs/2307.07924, 2023.

[31] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, and et al. Ui-tars: Pioneering automated gui interaction with native agents. *ArXiv*, abs/2501.12326, 2025.

[32] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[33] Noor Sajid, Philip J. Ball, and Karl J. Friston. Active inference: Demystified and compared. *Neural Computation*, 33:674–712, 2021.

[34] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[35] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[36] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag. *ArXiv*, abs/2501.09136, 2025.

[37] Paloma Sodhi, S.R.K Branavan, Yoav Artzi, and Ryan McDonald. Step: Stacked LLM policies for web actions. In *First Conference on Language Modeling*, 2024.

[38] Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[39] Alexander Tschantz, Beren Millidge, Anil. K. Seth, and Christopher L. Buckley. Reinforcement learning through active inference. *ArXiv*, abs/2002.12636, 2020.

[40] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Chatgpt empowered long-step robot control in various environments: A case application. *IEEE Access*, 11:95060–95078, 2023.

[41] 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 (TMLR)*, 2024.

[42] Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, Wanjun Zhong, Yining Ye, Yujia Qin, Yuwen Xiong, Yuxin Song, Zhiyong Wu, Bo Li, Chen Dun, Chong Liu, Fuxing Leng, Han rui Wang, Hao Yu, Haobin Chen, and et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *ArXiv*, abs/2509.02544, 2025.

[43] Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. Mobile-agent-e: Self-evolving mobile assistant for complex tasks. In *Workshop on Scaling Environments for Agents*, 2025.

[44] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning (ICML)*, 2025.

[45] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Qin Liu, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, and et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68, 2025.

[46] Jierui Xie, Bolesław K. Szymański, and Xiaoming Liu. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 344–349, 2011.

[47] 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 (ICLR)*, 2023.

[48] Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, Jitong Liao, Qi Zheng, Fei Huang, Jingren Zhou, and Ming Yan. Mobile-agent-v3: Fundamental agents for gui automation. *ArXiv*, abs/2508.15144, 2025.

[49] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *ACL (Findings)*, 2024.

[50] Gui-Min Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *ArXiv*, abs/2506.07398, 2025.

[51] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Y. Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

[52] Longtao Zheng, R. Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations (ICLR)*, 2023.

[53] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *ArXiv*, abs/2504.03160, 2025.

[54] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[55] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Y. Qiao, Zhaoxiang Zhang, and Jifeng Dai. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *ArXiv*, abs/2305.17144, 2023.

A Limitations

Despite the significant contributions of our framework, several limitations warrant further discussion.

First, our method focuses solely on updating explicit textual memory while keeping the LLM's parameters frozen. This means only updating $\mathcal{K}$ in the generative model's parameters $\eta = \theta_{\text{LLM}} \cup \mathcal{K}$. Although this strategy enables rapid adaptation to novel environments, it imposes a ceiling on long-term mastery. This distinction parallels the human cognitive process: individuals initially rely on summarizing declarative experiences when facing a new environment, but gradually develop implicit memory over time. As domain-specific experiences accumulate, further elevating the performance upper bound will require fine-tuning the LLM via methods such as preference learning or reinforcement learning.

Secondly, our framework consumes more tokens than a basic ReAct Agent. This increase arises from the exploration phase, which may necessitate more steps to investigate unknown mechanisms, as well as the additional token costs associated with the builder. Consequently, a trade-off between efficiency and performance should be considered during real-world applications.

Thirdly, while allowing agents to explore out of curiosity can enable them to autonomously try unknown mechanisms, it can pose certain risks in some security-sensitive environments. Therefore, for such environments, security mechanisms are needed during exploration, e.g., human-in-the-loop.

B Broader Impacts

The potential positive impacts of our work include reducing the amount of task-specific engineering, annotation, and fine-tuning needed to build useful agents for user tasks. At the same time, stronger autonomous adaptation may introduce risks. Agents that interact with realistic environments could perform unintended or harmful actions, especially when exploration is enabled in security-sensitive or irreversible settings. Misuse is possible if such agents are deployed to automate spam, manipulation, unauthorized data access, or other harmful workflows. Our experiments are conducted in benchmark environments rather than real deployments. For real-world use, we recommend sandboxed execution, scoped permissions, human approval for high-risk actions, and privacy filtering.

C Derivations of Operational Free-Energy Objectives

C.1 Preliminary: Derivations of VFE and EFE

Derivations of VFE. Start from the evidence:

$$
\begin{aligned}
-\ln P(\tilde{o}) &= -\ln \sum_{\tilde{z},\pi,\mathcal{K}} P(\tilde{o},\tilde{z},\pi,\mathcal{K}) && (7)\\
&= -\ln \sum_{\tilde{z},\pi,\mathcal{K}} Q(\tilde{z},\pi,\mathcal{K}\mid\tilde{o})\frac{P(\tilde{o},\tilde{z},\pi,\mathcal{K})}{Q(\tilde{z},\pi,\mathcal{K}\mid\tilde{o})} && (8)\\
&= -\ln \mathbb{E}_Q\left[\frac{P(\tilde{o},\tilde{z},\pi,\mathcal{K})}{Q(\tilde{z},\pi,\mathcal{K}\mid\tilde{o})}\right], && (9)
\end{aligned}
$$

where $\mathcal{K}$ represents the parameter. By Jensen's inequality $-\ln \mathbb{E}[X] \le \mathbb{E}[-\ln X]$, we obtain:

$$
-\ln P(\tilde{o}) \le \mathbb{E}_Q[\ln Q(\tilde{z},\pi,\mathcal{K}\mid\tilde{o}) - \ln P(\tilde{o},\tilde{z},\pi,\mathcal{K})] := F[\tilde{o},\tilde{z},\pi,\mathcal{K}] \tag{10}
$$

Defining the right-hand side as the Variational Free Energy (VFE), this is the standard derivation that VFE is the upper bound of surprise/negative log evidence. It can also be written as:

$$
\begin{aligned}
F[\tilde{o},\tilde{z},\pi,\mathcal{K}] &= -\mathbb{E}_Q[\ln P(\tilde{o}|\tilde{z},\pi,\mathcal{K})] + \mathbb{E}_Q[\ln Q(\tilde{z},\pi,\mathcal{K}\mid\tilde{o}) - \ln P(\tilde{z},\pi,\mathcal{K})] && (11)\\
&= \underbrace{-\mathbb{E}_Q[\ln P(\tilde{o}|\tilde{z},\pi,\mathcal{K})]}_{\text{Prediction Error}} + \underbrace{D_{\text{KL}}\left[Q(\tilde{z},\pi,\mathcal{K}|\tilde{o}) \parallel P(\tilde{z},\pi,\mathcal{K})\right]}_{\text{Complexity}} && (12)
\end{aligned}
$$

Derivations of EFE. Based on the definition in Eq. 10, the Expected Free Energy (EFE) for the τ step can be defined as (the model parameter $\mathcal{K}$ and the current state z_t are omitted from conditions

for simplicity):

$$G(\pi,\tau) = \mathbb{E}_{Q(o_\tau,z_\tau|\pi)}\left[\ln Q\left(z_\tau \mid \pi\right) - \ln P\left(o_\tau, z_\tau \mid \pi\right)\right] \tag{13}$$

$$= \mathbb{E}_{Q(o_\tau,z_\tau|\pi)}\left[\ln Q\left(z_\tau \mid \pi\right) - \ln Q\left(z_\tau \mid o_\tau, \pi\right) - \ln P\left(o_\tau\right)\right] \tag{14}$$

$$= \underbrace{\mathbb{E}_{Q(o_\tau,z_\tau|\pi)}\left[\ln Q\left(z_\tau \mid \pi\right) - \ln Q\left(z_\tau \mid o_\tau, \pi\right)\right]}_{\text{(-ve) Mutual Information (Epistemic Value)}} - \underbrace{\mathbb{E}_Q \ln P_{\text{pref}}\left(o_\tau \mid g\right)}_{\text{Pragmatic Value}}, \tag{15}$$

where $Q\left(o_\tau, z_\tau \mid \pi\right) = P\left(o_\tau \mid z_\tau\right) Q\left(z_\tau \mid \pi\right)$ is the joint distribution of predictions for the τ step. Here, $Q\left(z_\tau \mid \pi\right)$ is the Worker's belief in the future state based on the current state and action. The first part of Eq 15 is exactly the definition of (negative) mutual information $I_Q\left(z_\tau; o_\tau \mid \pi\right)$:

$$I_Q(z_\tau; o_\tau \mid \pi) = H[Q(o_\tau \mid \pi)] - \mathbb{E}_{Q(z_\tau|\pi)}[H(Q(o_\tau \mid z_\tau, \pi))] \tag{16}$$

$$= \mathbb{E}_{Q(o_\tau|\pi)}[D_{\mathrm{KL}}\left(Q(z_\tau \mid o_\tau, \pi) \| Q(z_\tau \mid \pi)\right)]. \tag{17}$$

C.2 The EFE upper-bound surrogate used by the Worker

Starting from the epistemic term of the EFE:

$$I_Q(z_\tau; o_\tau \mid \pi) = H[Q(o_\tau \mid \pi)] - \mathbb{E}_{Q(z_\tau|\pi)}[H(Q(o_\tau \mid z_\tau, \pi))]. \tag{18}$$

In our semantic setting, once the relevant latent semantic state z_τ and mechanism are specified, the predicted observation is close to deterministic. Hence $H(Q(o_\tau \mid z_\tau, \pi)) \approx 0$, and the information-gain term is approximately the predictive entropy $H[Q(o_\tau \mid \pi)]$. Intuitively, high entropy here corresponds to reducible uncertainty: after the agent acts and observes the outcome, the posterior state becomes substantially more certain.

For the pragmatic term, we adopt the standard Boltzmann prior preference assumption [27, 39]

$$P(o_\tau \mid g) = \frac{\exp(\beta r(o_\tau))}{Z_g}, \quad \text{so } -\mathbb{E}_{Q(o_\tau|\pi)}[\ln P(o_\tau \mid g)] = -\beta \mathbb{E}_{Q(o_\tau|\pi)}[r(o_\tau)] + \ln Z_g. \tag{19}$$

The normalizer $\ln Z_g$ is independent of the policy and can be dropped for policy selection. Summing from $\tau = t+1$ to T gives

$$G(\pi) \approx -\sum_{\tau=t+1}^{T} H[Q(o_\tau \mid \pi)] - \beta \sum_{\tau=t+1}^{T} \mathbb{E}_{Q(o_\tau|\pi)}[r(o_\tau)]. \tag{20}$$

If we define the joint predicted rollout distribution as $Q(\tilde{o}_{>t} \mid \pi)$, subadditivity of joint entropy gives $H[Q(\tilde{o}_{>t} \mid \pi)] \leq \sum_{\tau=t+1}^{T} H[Q(o_\tau \mid \pi)]$. Combining this inequality with the per-step form above yields the rollout-level upper-bound surrogate

$$G(\pi) \leq \widehat{G}(\pi) := -H[Q(\tilde{o}_{>t} \mid \pi)] - \beta V(o_t; \pi), \tag{21}$$

which yields Eq. (4) (**Lemma 3.1.**). Therefore, among a finite set of LLM-proposed candidate policies $\{\pi_i\}$, the Worker approximates

$$\pi^* = \arg\min_i \widehat{G}(\pi_i) = \arg\min_i \left(-H[Q(\tilde{o}_{>t} \mid \pi_i)] - \beta V(o_t; \pi_i)\right). \tag{22}$$

We do not require the LLM to output calibrated probabilities and explicitly compute the entropy term. In high-dimensional semantic spaces, such numerical estimates tend to propagate errors. Instead, the Worker jointly reasons about predicted rollouts, confidence, unclear rules, and goal progress, which provides a tractable proxy for the EFE trade-off between epistemic value and pragmatic value.

C.3 The VFE-guided objective used by the Builder

Starting from the VFE definition in Eq. (10), we treat the frozen LLM weights as fixed and optimize only the semantic memory $\mathcal{K}$. Moreover, we use the factorized LLM generative model:

$$P_{\mathrm{LLM}}(\tilde{o}, \tilde{z}, \pi, \mathcal{K} \mid g) = P_{\mathrm{LLM}}(\mathcal{K}) P_{\mathrm{LLM}}(\pi \mid g, \mathcal{K}) \prod_{t=1}^{T} [P_{\mathrm{LLM}}(o_t \mid z_t, \mathcal{K}) P_{\mathrm{LLM}}(z_t \mid z_{t-1}, a_{t-1}, \mathcal{K})]. \tag{23}$$

Substituting this factorization into VFE gives, up to constants independent of $\mathcal{K}$,

$$F = \mathbb{E}_Q \Bigg[\ln Q(\tilde{z}, \pi, \mathcal{K} \mid \tilde{o}) - \ln P_{\mathrm{LLM}}(\mathcal{K}) - \ln P_{\mathrm{LLM}}(\pi \mid g, \mathcal{K}) - \sum_{t=0}^{T} \ln P_{\mathrm{LLM}}(o_t \mid z_t, \mathcal{K}) \tag{24}$$

$$- \sum_{t=1}^{T} \ln P_{\mathrm{LLM}}(z_t \mid z_{t-1}, a_{t-1}, \mathcal{K}) \Bigg]. \tag{25}$$

For an observed trajectory $\xi = (\tilde{o}^\xi, \tilde{z}^\xi, \pi^\xi, g^\xi)$, the executed policy is fixed, and the Builder uses hindsight to approximate the posterior $Q_{\mathrm{LLM}}(z_t \mid \tilde{o}^\xi, \pi^\xi, \mathcal{K})$. Under the common factorization over time, the terms involving $\ln Q$ and transition priors form a sum of KL divergences:

$$F_B(\xi, \mathcal{K}) = - \ln P_{\mathrm{LLM}}(\mathcal{K}) - \ln P_{\mathrm{LLM}}(\pi^\xi \mid g^\xi, \mathcal{K}) - \sum_{t=0}^{T} \ln P_{\mathrm{LLM}}(o_t^\xi \mid z_t^\xi, \mathcal{K}) \tag{26}$$

$$+ \sum_{t=1}^{T} D_{\mathrm{KL}} \Big(Q_{\mathrm{LLM}}(z_t \mid \tilde{o}^\xi, \pi^\xi, \mathcal{K}) \Big\| P_{\mathrm{LLM}}(z_t \mid z_{t-1}^\xi, a_{t-1}^\xi, \mathcal{K}) \Big). \tag{27}$$

To make the complexity term explicit, we place a Laplace/exponential prior over the size of the textual memory:

$$P_{\mathrm{LLM}}(\mathcal{K}) \propto \exp(-\lambda |\mathcal{K}|_{\mathrm{tokens}}), \quad \text{so } - \ln P_{\mathrm{LLM}}(\mathcal{K}) = \lambda |\mathcal{K}|_{\mathrm{tokens}} + \mathrm{const}. \tag{28}$$

This yields Eq. (5) (**Lemma 3.2.**). The resulting objective provides a direct interpretation of Builder updates: entities improve observation likelihood; rules reduce the mismatch between posterior hindsight states and transition priors; skills increase the prior probability of successful policies; and the memory prior penalizes unnecessary complexity. To operationalize this complexity penalty, a proposed update $\Delta\mathcal{K}$ is evaluated against the criterion:

$$\Delta F_B(\xi, \Delta\mathcal{K}) = F_B(\xi, \mathcal{K} \cup \Delta\mathcal{K}) - F_B(\xi, \mathcal{K}) < 0, \tag{29}$$

This motivates counterfactual validation: an update is retained only when its predicted error reduction is judged to outweigh the added complexity of storing the update.

D Method Details

D.1 Structure of Semantic Memory

Our explicit memory bank comprises three distinct types of nodes: Entity, Rule, and Skill. Each node type is characterized by specific attributes:

- Entity: contains 'name' and 'description';
- Rule: contains 'name', 'description', 'status' (labeled as either 'clear' or 'unclear'), 'base_entity' (the entity most relevant to the rule), and an 'evidences' field to track the rule's validation and evolution process;
- Skill: contains 'name', 'description', 'skill body', 'evidences', and a 'dependency' field indicating reliance on nested skills to complete sub-tasks.

In addition to these individual nodes, each Community node encompasses the attributes listed above, along with an explicit list of member node names. As mentioned in the main text, the description field of any node may contain Wiki-style citations. On the backend, we convert all such references and dependency relationships into pairwise edges within the node graph, which is subsequently utilized for graph expansion during community detection and retrieval tasks.

D.2 Overlapping Community Detection

To maintain a scalable and well-organized knowledge base $\mathcal{K}$, we perform periodic community detection every E epochs (we set $E = 5$ in our implementation). This interval is motivated by two considerations: (1) allowing sufficient knowledge to accumulate between clustering operations, and (2) avoiding excessive computational overhead from frequent restructuring.

Table 5: Experiment results on WebArena benchmark with Gemini-2.5-pro. The results are averaged across 684 tasks of 5 subsets.

Methods	**Pipeline**	**Tokens(k)**	**Success Rate(%)**
ReAct	Test only	155.8	46.7
ReasoningBank	Test-Time Adaptation	-	53.9
AutoManual	Test-Time Adaptation	190.5	51.4
AutoManual	Building-Testing	*Building:* 196.3 *Testing:* 174.4	54.7
FEP-Agent	Test-Time Adaptation	203.3	60.1
FEP-Agent	Building-Testing	*Building:* 210.6 *Testing:* 169.8	62.9

Graph Construction. We first construct an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from the Wiki-style knowledge base. For the Entity/Rule/Skill community, the nodes correspond to the entity nodes in $\mathcal{V}_E$, $\mathcal{V}_R$, or $\mathcal{V}_S$, and the edges come from two sources: (1) *Reference-based edges*: When the description of a node refers to other nodes through `[[...]]` references, we create `related_to` edges between all the mentioned nodes. (2) *Semantic similarity edges*: We compute pairwise cosine similarities between node embeddings, which can be substituted with Approximate Nearest Neighbor (ANN) search for scalability. Edges are added between nodes whose similarity exceeds a threshold θ_{sim} (default 0.6), preventing isolated nodes from being excluded due to sparse citations.

Weighted SLPA Algorithm. Unlike traditional community detection methods (e.g., GraphRAG [15]) that produce mutually exclusive partitions, we employ the Speaker-Listener Label Propagation Algorithm (SLPA) [46] to detect overlapping communities:

$$\{C_i\} = \text{SLPA}(\mathcal{V}, \mathcal{E}_{sim} \cup \mathcal{E}_{cite})$$

This choice reflects our observation that many environmental concepts are inherently multi-faceted; for example, an entity "Submit Button" may belong to both "Form Filling" and "Navigation" themes simultaneously. The algorithm operates through an iterative speaker-listener process over T iterations (we use $T = 30$): In each iteration, every node acts as a *listener* that receives labels from its neighbors (acting as *speakers*). Each speaker probabilistically selects a label from its memory based on historical frequency, weighted by the connecting edge weight. The listener then adopts the most popular label received. After T iterations, each node retains all labels that appear with frequency exceeding a threshold r (we use $r = 0.15$), allowing nodes to belong to multiple communities.

Incremental Community Management. To ensure temporal stability across epochs, we maintain *seed labels* from previous community detection runs. When initializing SLPA, nodes that were previously assigned to a community start with their historical seed label, promoting consistent community evolution rather than arbitrary restructuring. Rather than discarding all existing communities and rebuilding from scratch, we adopt an incremental update strategy. For each newly detected community C_{new}, we compute its overlap with existing communities using the *containment coefficient*:

$$\text{overlap}(C_{new}, C_{old}) = \frac{|C_{new} \cap C_{old}|}{\min(|C_{new}|, |C_{old}|)}$$

This metric is more suitable than Jaccard similarity for detecting containment relationships, as it properly handles cases where one community is a subset of another. Based on the overlap scores, we ask a Community Builder agent to update existing or upsert new community nodes through operators of the knowledge base.

D.3 Retrieval Mechanism

To navigate the expanding knowledge base $\mathcal{K}$ while balancing token consumption and information granularity, we implement a hybrid retrieval mechanism. As mentioned in the main text, we inject the current communities into the context of all agents (Worker, Builder, and the subsequent Retriever) as a coarse-grained "cognitive map" of $\mathcal{K}$. Based on this community-level index and wiki-style reference, agents can understand the global scope of available knowledge and quickly localize candidate concepts. Agents can further issue targeted queries to retrieve specific nodes, following agentic RAG [36].

Hybrid Retrieval for Worker. Although the Worker agent now possesses the capability to directly query $\mathcal{K}$, we find that the reasoning-heavy Worker agent often fails to formulate accurate queries and incurs prohibitive token costs. Therefore, we employ lightweight auxiliary agents to first gather broadly relevant potential information, while the Worker initiates more precise queries for deeper concepts. Specifically, before each Worker step, a lightweight *Retriever Agent* analyzes the current observation and history to initiate embedding-based semantic search or exact name matching. Retrieved nodes are further expanded to include graph-connected neighbors. Based on relevance to the query, the top 5 nodes of each type (Entity/Rule/Skills) are retained. Subsequently, another lightweight *Filter Agent* performs filtering and reranking, forwarding only the most relevant nodes to the Worker.

D.4 Decoupled Builders

In our framework, the Builder is responsible for analyzing trajectories and updating the knowledge base $\mathcal{K}$. To effectively handle the distinct aspects of this update task, we decompose the Builder into three specialized LLM agents. (1) **Concluder Agent**: First, this agent reviews the complete trajectory to identify specific steps where state misinterpretations occurred, articulating both the nature of the bias and the correct understanding. (2) **Insight Builder**: Second, based on the trajectory and the Concluder's analysis, this agent retrieves relevant existing nodes from the knowledge base. It extracts valuable Entities and Rules, which are primarily derived from 'Surprise Moments.' Before updating K, it classifies these Rules as either 'clear' or 'unclear' and strictly verifies their necessity through Counterfactual Validation. (3) **Skill Builder**: Finally, if the current trajectory is successful, this agent extracts reusable skills. It supports hierarchical nesting, where a skill for a long-horizon task may encompass multiple sub-task skills, and subsequently updates the knowledge base.

E Experiment Details

E.1 Implementation Details

In our framework, we adopt a standard building-testing pipeline: a subset of tasks is sampled from the environment for worker exploration and for the builder to optimize $\mathcal{K}$. Subsequently, the optimized $\mathcal{K}$ is frozen, and the worker equipped with this $\mathcal{K}$ is evaluated on a separate set of tasks. Specific implementations for each environment are as follows: (1) **WebArena**: We randomly select 228 tasks to serve as building tasks for both our method and AutoManual. (2) **AndroidWorld**: We generate one task from each of the 116 task templates (using random seed 1) to form a set of 116 building tasks. We then adhere to the standard AndroidWorld test set by using tasks generated with random seed 0 from each template for testing. (3) **ALFWorld**: Following AutoManual, we randomly sample 6 tasks for each of the 6 task types from its training set, resulting in 36 building tasks. Evaluation is conducted on the 135 tasks within its standard "eval_out_of_distribution" test set.

We use Gemini-2.5-pro for the Builder. We use Gemini-2.5-pro, Gemini-2.5-flash, or GPT-4o-mini for the worker, depending on the specific experiment. For the remaining Retriever and Filter, we use Gemini-2.5-flash. For semantic similarity queries on $\mathcal{K}$, we employ OpenAI's text-embedding-3-large to embed both query and node description. We calculate the vector similarity between them and apply a threshold of 0.4 for automatic filtering.

Discussion of Pipelines. We find that AWM (online) and ReasoningBank employ a Test-Time Adaptation (TTA) pipeline, utilizing an LLM-as-a-judge [13] to provide reward signals and continuously performing the building process across all test tasks. While this approach eliminates the need for ground-truth environmental rewards and allows for continuous improvement, it incurs extra token consumption and is sensitive to the order of test tasks. Furthermore, ReasoningBank adopts a Test-Time Scaling (TTS) pipeline, executing parallel runs with $k = 5$ for each test task. Therefore, our default configuration employs a more robust building-testing pipeline. Nevertheless, our method can be readily adapted to this supervision-free setting by leveraging an LLM-as-a-judge to provide reward signals, as the prior preferences of LLMs trained via RLHF are largely aligned with human preferences. As shown in Tab. 5, our method also demonstrates strong performance under the TTA pipeline setting.

Discussion of Token Usage. Tab. 5 presents the token consumption of different methods. In the building phase, our method consumes slightly more tokens because the worker allocates a portion

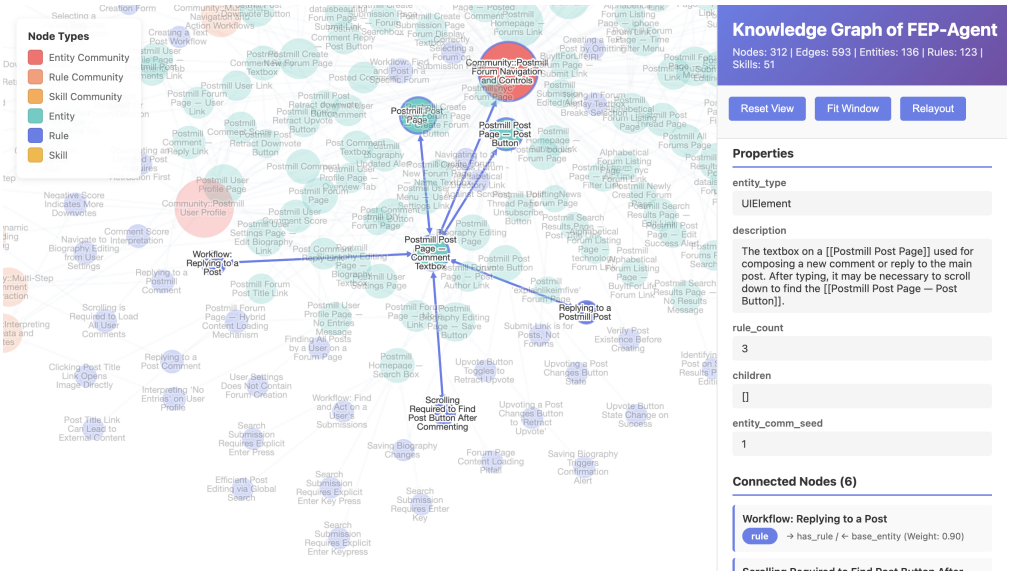


Figure 5: Visualization of any node in the knowledge base $\mathcal{K}$ can link to other nodes by adding a wiki reference to its description. This Knowledge Base is built by FEP-Agent on WebArena.

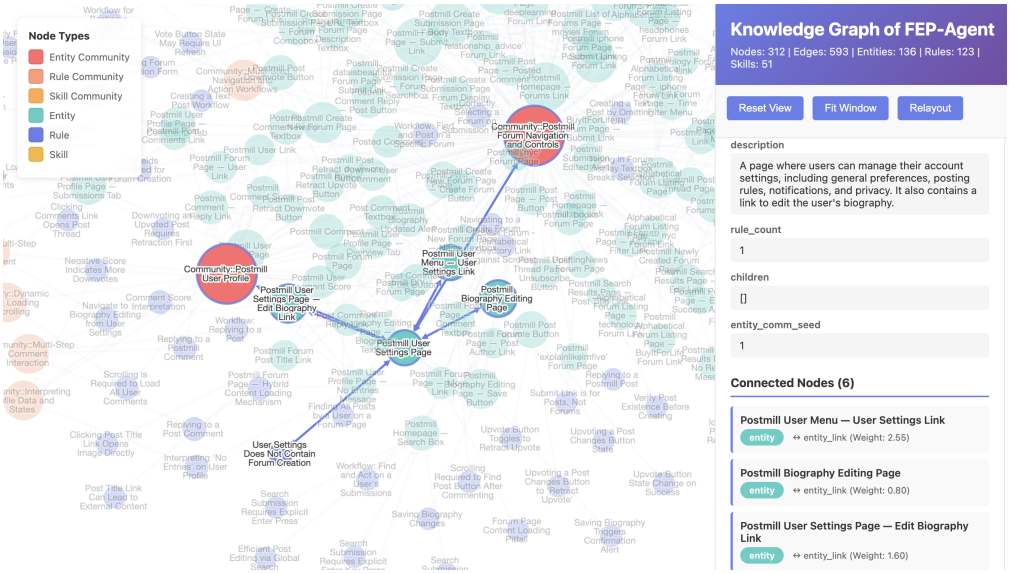


Figure 6: Visualization of overlapping communities can represent multi-faceted nodes. This Knowledge Base is built on WebArena.

of steps to exploration. However, during the testing phase, we no longer prompt the worker to explore but rather to exploit the acquired knowledge, resulting in consumption comparable to existing methods.

E.2 Additional Analyses

In this section, we provide a qualitative analysis of the learned Semantic Memory $\mathcal{K}$ to visualize how FEP-Agent constructs, organizes, and abstracts knowledge.

Wiki-style Connectivity for Complex Relationships. Fig. 5 illustrates a microscopic view of the knowledge graph constructed in the WebArena environment. Unlike traditional knowledge graphs that rely on rigid triplet constraints (Head, Relation, Tail), our Wiki-style Semantic Memory allows for more expressive, non-binary relationships. As shown in the "Properties" panel, the node

description for the Comment Textbox explicitly references the [[Postmill Post Page]] and [[Postmill Post Page - Post Button]] via a Wiki-style citation. This citation mechanism acts as a hyper-edge, capturing the causal dependency that "finding the button is necessary after typing in the textbox." This structure allows the Worker Agent to retrieve the context of an entity (e.g., its related actions and constraints) simply by reading its description, thereby grounding the semantic state z_t in a rich web of interconnected concepts.

Multi-faceted Representation via Overlapping Communities. Fig. 6 shows that real-world concepts are often multi-faceted, and the effectiveness of our overlapping community detection approach. The central node, "Postmill User Settings Page", is simultaneously associated with two distinct communities: "Community::Postmill User Profile" and "Community::Postmill Forum Navigation and Controls". This reflects the true nature of the page: it is both a destination for profile management and a hub for navigating forum controls. By allowing nodes to belong to multiple communities, this flexibility minimizes the risk of retrieval failure caused by rigid, mutually exclusive clustering.

Macroscopic Topology of Embodied Knowledge. Fig 7 presents a macroscopic view of the knowledge graph evolved after completing building tasks in the ALFWorld environment. The graph exhibits a structured topology where Entities (green) and Rules (blue) are organically organized around high-level communities (red nodes), such as "Community::Appliance-Based Cooling" or "Community::Cleaning Supplies". The density of connections indicates that the Builder Agent successfully integrated isolated observations into a coherent world model. The clear separation yet interconnection of different household tasks demonstrates that FEP-Agent does not merely memorize trajectories but constructs a structured "cognitive map" of the environment, enabling efficient navigation and planning in the large state space of embodied tasks.

Abstraction and Rule Generalization. A key capability of the community detection is abstracting general rules from specific instances. Fig. 8 highlights a high-level rule node: "Container Appliance Interaction Protocol". Initially, the agent may have learned separate rules for using a microwave and a fridge. However, after the community detection, the LLM agent synthesized a general protocol: "go to appliance $\rightarrow$ open the appliance $\rightarrow$ perform action $\rightarrow$ close the appliance." The visualization shows edges linking this protocol to specific instances like "Microwave Heating Procedure" and "Fridge Cooling", indicating that the agent understands these are subclasses of the same general mechanism. This abstraction allows FEP-Agent to generalize zero-shot to new container-like appliances (e.g., a safe or a cupboard) by applying the learned protocol, significantly boosting its adaptability.

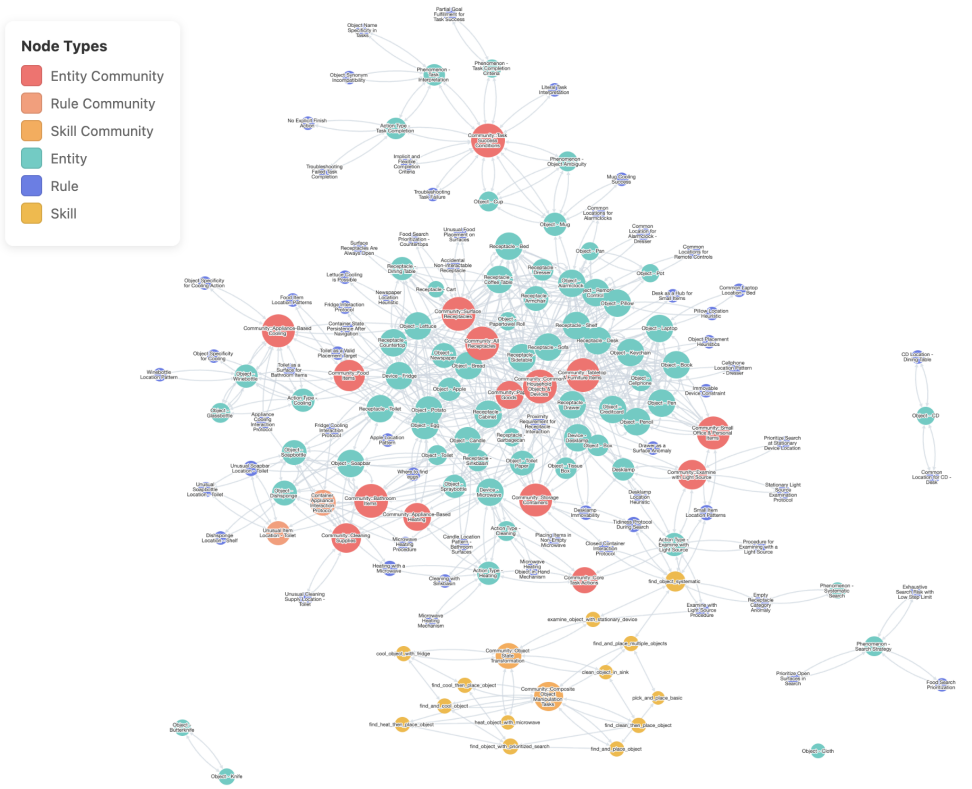


Figure 7: Visualization of the knowledge graph built by FEP-Agent on ALFWorld.

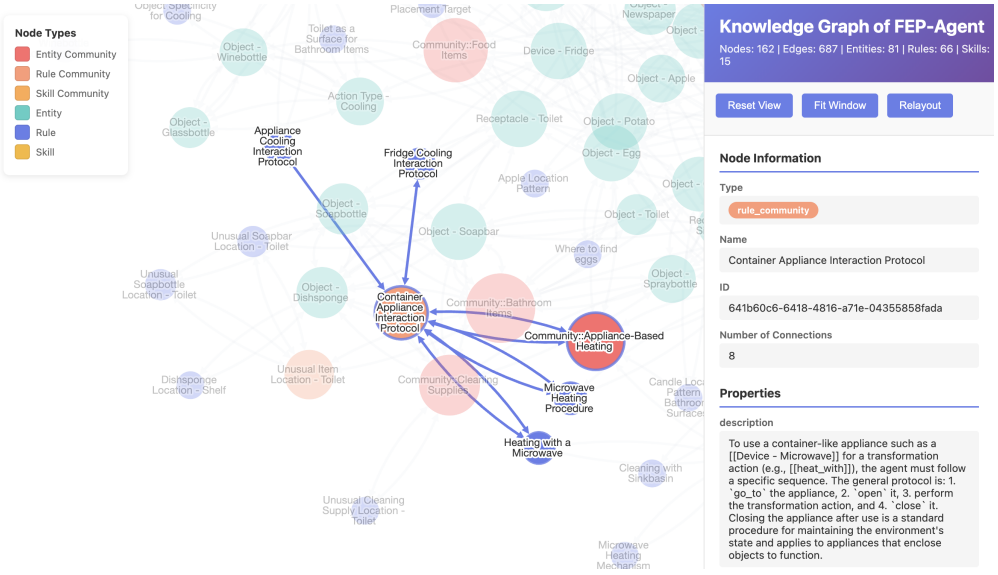


Figure 8: Visualization of abstracting general rules from multiple concrete rules. This Knowledge Base is built on ALFWorld.

E.3 Prompts Used in Our Method

System Prompt

```
[Role]
You are an AI agent to complete tasks in an environment. Your main goal is to select the current policy
     that minimizes the expected free energy by balancing goal realization and information gain.

[Admissible Actions]
{Env Actions}

[Knowledge and Retrieval]
In your context, you may have performed some retrieval operations and obtained:
- **Retrieved Entities**: Entities in the knowledge bank.
- **Retrieved Rules**: Environmental mechanisms and successful strategies that might guide your approach.
     Some rules may be marked as "unclear" (speculative/needs verification).
- **Retrieved Skills**: Previous succeeded workflows that might be relevant.

**To retrieve information** If the current retrieved information is insufficient, you have a special
     tool, `RetrievalTool`, to request more: You can choose not to immediately output page actions, but
     instead format your entire "output" as a `RetrievalTool` request to search the knowledge base.

[Output Formats]
Your action output should consist of the following things:
* ### "output.semantic_state" (z_t): Infer the current semantic state from the observation, history, and
     retrieved K. This is the textual posterior state used by future Workers and Builders.
Structure your description naturally in three parts:
  (1) **Action result** -  whether the previous action achieved its intended effect;
  (2) **Current layout** - concisely describe where you are and the overall structure;
  (3) **Critical details and analysis** - provide rich details ONLY for task-relevant elements. Also
     state whether the current observation **matches or violates your previous prediction** and whether
     anything is **surprising** given the knowledge base. Do NOT describe every element - only those
     relevant to deciding the next action. Simultaneously, analyze how the provided
     [[entities/rules/skills]] inform your next step, citing specific items using [[Exact Name]].
* ### "output.prediction" (pred_t): In natural language, describe 1+ candidate policies/actions and their
     expected outcomes, along with success probability and predictive confidence. If you mention multiple
     candidates, separate them clearly.
* ### "output.plan" (plan_t): You should consider whether to change or continue with your previous plan
     based on the new observation and the retrieved knowledge. Prefix each step with □ for task-critical
     actions or 🔍 for exploratory experiments (e.g., testing unclear rules or investigating unknown
     mechanisms).
* ### "output.code" (a_t): This is the env action to be executed. It must be a single line of Python code
     corresponding to one of the "Admissible Actions" list.

[Extra Instructions]
* Instructions For Your "output.prediction": Use 1-3 short sentences. Each sentence should include the
     candidate action, expected rollout, success probability, and confidence level (1-5). If the
     prediction relies on a specific rule, you may cite it with [[Rule Name]].
* Instructions For "output.plan": 1. Planning Strategy: You must generate a plan that balances two
     objectives: - Pragmatic value □: If retrieved clear rules or skills reliably connect the current
     state to the goal, exploit them. - Epistemic value 🔍: If you encounter "Unclear Rules" or high
     uncertainty in your prediction, you can prioritize actions that resolve this ambiguity. 2. Trigger
     🔍 only with a **testable hypothesis/question** (something you can confirm/deny via a small number
     of actions and a clear expected observation). Its purpose is **mechanism verification**: when you
     have a concrete uncertainty or hypothesis about how the environment works, and validating it could
     improve success now or in future tasks. 3. Keep 🔍 **low-risk and ideally **reversible**. Balance
     curiosity with task progress-prioritize task completion as steps are limited; explore only when you
     have spare capacity.
```

Figure 9: System prompt for Worker Agent

System Prompt

```
[Role]
Your task is to update Entities and Rules in the semantic memory K using completed trajectories and the
     Concluder's hindsight analysis. You should minimize the Builder-side Variational Free Energy by
     reducing observation-level and transition-level prediction errors while controlling memory
     complexity.

[Admissible Functions]
- `upsert_entity(name: str, description: str, new_name: Optional[str] = None)`
- `upsert_rule(rule_name: str, rule_description: str, base_entity: str, status: str = "clear", evidences:
     List[str], new_name: Optional[str] = None)`
- `delete_rule(rule_name: str)`
**Knowledge and Retrieval**:In your context, you will also be given a partial preview sample of available
     knowledge indices. If this information is insufficient for your understanding, you have a special
     tool to request more: You can choose not to immediately output code, but instead format your entire
     output as a `RetrievalTool` request to search the knowledge base.

[Clarifications about Entity]
**Entity (noun-like, static subject)**: A concrete object, location/area, action type, state/phenomenon,
     or constraint extracted from an event. An entity's `description` should clearly describe what the
     entity IS and the properties of the entity. You can use Wiki-style reference `[[Entity Name]]` to
     refer to related entities (e.g., "A link appearing on the [[Reddit Forums Directory Page]] that can
     open an alphabetical list of forums.").

[Clarifications about Rule]
**Rule (verb-like, dynamic relation/behavior/interaction/procedural knowledge)**: A localized,
     non-trivial statement capturing relationships between entities, environmental mechanisms, or
     behavioral patterns that are NOT duplicate entity descriptions.
  - **Description**: In rule description, you can use Wiki-style reference to refer to `[[Entity
     Name/Rule Name/skill_name]]`. Use only names that exist in the retrieved context or ones you have
     just added in this code block.
  - **Evidences**: List of evidence strings showing where this rule was discovered
  - **Status**: 1. "clear": For directly observed facts, behaviors you saw happen, confirmed
     relationships. These don't need additional evidence unless significantly changed. 2. "unclear": For
     inferred mechanisms or patterns seen only once, mark these for future verification or
     experimentation.

**Complexity Control**
  - Distinguish Rules from Entity Descriptions: Rules must provide additional insight beyond what the
     entity description already conveys: Reveal entity relationships, special mechanisms, conditional
     behaviors, or non-obvious consequences. Ask yourself: "Does this rule add new insight beyond the
     entity description?" If no, enhance the entity description instead.
  - Counterfactual Validation: Accept a proposed update only if it provides **tool value** (improves task
     success, reduces steps, avoids errors) or **epistemic value** (reduces uncertainty, explains a
     surprise, resolves an unclear mechanism). Ask yourself: "If this knowledge had been known, would the
     surprise at corresponding steps have been significantly reduced?" If a rule does not change
     decisions or explain surprises, do not add it, and might enhance entity descriptions instead.
  - Distinguish Between Rules and Specific Events: You should prefer generalizable or reusable-level
     rules. When generalization is necessary, but you feel uncertain about the generalized rule, it may
     be appropriate to mark it as unclear.

[Guidelines]
**Knowledge Extraction Workflows** 1. Identify Noticeable Entities in the trajectories (Focus on
     Actionable Knowledge: Extract information that helps future task execution). 2. Capture Helpful
     Behavioral Rules: Document relationships, patterns, and mechanisms. Prefer rules that explain
     **surprise** (prediction failed) or **verification** (unclear hypothesis resolved). 3. Retrieve and
     Compare with Existing Knowledge: Before adding new entities/rules, check if the (almost) same ones
     already exist and retrieve their details to see how to update.
```

Figure 10: System prompt for Insight Builder Agent

System Prompt

```
[Role]
Your task is to extract reusable skills from successful task trajectories and store them for future
     reference.

[Admissible Functions for Building]
- ‘upsert_skill(skill_name: str, description: str, skill_body: Optional[str] = None, dependencies:
     Optional[List[str]] = None, evidences: Optional[List[str]] = None, new_name: Optional[str] = None)‘

**Parameters:**
- **skill_name**: The name of the skill (it describes the task_type this skill targets)
- **description**: Brief, natural description covering (1) application scenario/problem and (2) solution
     approach.
- **dependencies**: Optional. A list of other code skill’s skill_names this skill depends on
- **evidences**: List of evidence strings showing where this skill was derived.
- **skill_body**: Pseudocode process guidance (NOT executable Python). Use comments to express logic,
     element IDs, and worker tips. It’s process guidance for future LLM workers. Future workers will read
     this pseudocode, then observe the actual page and decide actions. Should start with ‘#
     Precondition:‘ and ‘# Task Variables:‘ comments to explain the skill’s application scope and
     conditional variables.

**Knowledge and Retrieval**: In your context, you will also be given a partial preview sample of
     available knowledge indices. You may have performed some retrieval operations and obtained related
     existing knowledge. If this information is insufficient for your understanding, you have a special
     tool to request more: You can choose not to immediately output code, but instead format your entire
     output as a ‘RetrievalTool‘ request to search the knowledge base.

[Output Format]
Provide your response in two fields: ‘thought‘ (your reasoning process) and ‘code‘ (the Python script to
     execute).

[Skill Extraction Guidelines]
When processing a successful trajectory, you must: 1. **Identify Key Actions**: Extract the sequence of
     actions that led to success 2. **Generalize**: Focus on patterns that can apply to similar tasks,
     not just this specific instance. 3. **Handle Skill Updates**: If a skill already exists for this
     task_type, check whether you should update it 4. **Subflows**: Identify self-contained subflows that
     are broadly reusable (e.g., "Sort forum posts by New"). For each reusable subflow, create a separate
     skill and declare dependency edges: the overarching flow skill ‘depends_on‘ the subflow skills. Only
     declare dependencies that exist in the retrieved context or are added in your code block 5. **Defer
     Local Flows**: The Insight Builder has added single-entity focused flows (e.g., "Open sort menu and
     select New"). DO NOT add such local skills, which duplicate entities’ rules. 6. **Retrieve and
     Compare with Existing Skills**: Before adding new skills, check if the (almost) same ones already
     exist and retrieve their details to see how to update.
```

Figure 11: System prompt for Skill Builder Agent

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main claims of the paper: formulating LLM agent self-evolution as an Active Inference process, instantiating EFE/VFE minimization through Worker and Builder agents, and introducing a structured semantic memory. These claims are supported by the theoretical formulation, experiments, and ablation studies.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The appendix includes derivations for VFE, EFE, the Worker's EFE approximation, and the Builder's VFE approximation. The assumptions include the Boltzmann prior preference assumption, near-deterministic semantic likelihood, and an exponential prior over memory size.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper describes the benchmarks, task splits, baselines, backbone LLMs, evaluation metric, building-testing pipeline, retrieval settings, community detection settings, and prompts used by the agents. These details provide a reasonable path for reproducing the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Source code is provided in the supplemental material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the evaluated benchmarks, task numbers, data splits, building/testing protocols, model backbones, retrieval and embedding settings, community detection hyperparameters, evaluation metrics, and agent prompts. These details are provided in the main experiments section and the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We directly report success rate using point estimation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We use the OpenAI or Gemini API for all experiments and report the cost.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper includes a Broader Impacts section.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not provide pretrained language models, image generators, or scraped datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have stated and cited the asset used in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are a core component of the proposed method. The paper describes how LLMs are used as Worker, Builder, Retriever, and Filter agents, and specifies the backbone models used in experiments, including Gemini-2.5 variants and GPT-4o-mini, as well as the embedding model used for semantic retrieval.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.