SEER: BOOSTING AGENTIC REASONING VIA ENTROPY-REDUCING EVIDENCE SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic reasoning has shown great potential for many real-world applications. Its success largely relies on whether limited context windows can contain the evidence needed for subsequent decisions of agents. However, properly selecting the useful evidence remains highly challenging due to the rapidly growing interaction histories. To tackle this problem, we propose **S**electing **E**vidence by **E**ntropy **R**eduction (**SEER**), a novel and theoretically-grounded context management framework to boost the agentic reasoning. We cast the context management rigorously in an Information Bottleneck (IB) formulation, which can well balance the *compactness* and *effectiveness* of contexts. Then, the IB formulation naturally gives rise to SEER, which measures the value of each evidence based on its marginal reduction in the agent's entropy over subsequent environment feedback. By prioritizing high-value evidence, SEER dynamically constructs compact contexts enriched with information that supports subsequent reasoning. Extensive experiments on challenging benchmarks show that SEER significantly outperforms all baselines across multiple model scales and families in terms of accuracy and task completion, improving over the strongest baseline by up to **9.4 points**.

1 INTRODUCTION

Large language model (LLM)-based agents (OpenAI, 2026; Anthropic, 2026) have made rapid progress in complex real-world applications such as software engineering (Wang et al., 2025b;a), web navigation (Wei et al., 2025; Wang et al., 2026a), and cybersecurity (Wu et al., 2025b; Deng et al., 2026). A key strength of agents is their ability to solve complex tasks through iterative interactions with environments. At each step, they reason, act, and incorporate environment feedback to guide subsequent decisions (Zhang et al., 2026; Chowa et al., 2026). As interactions extend, agents can continually acquire and accumulate new evidence from the environment, enabling them to reason beyond what is available at the outset (Xu et al., 2025; Wu et al., 2025a; Wan et al., 2026).

Context plays a vital role in the agentic reasoning, as it directly determines what the agent receives as input at each step (Yu et al., 2026; Wang et al., 2026b). This role becomes particularly critical in unfamiliar environments, where parametric knowledge alone may be insufficient, making interaction-derived evidence in the context essential for subsequent reasoning (Chen et al., 2026; Tajwar et al., 2025). This has recently motivated growing interest in *context engineering* (Mei et al., 2025; Zhang et al., 2025), which manages the information provided to agents during inference. However, as long-horizon interactions unfold, useful evidence becomes increasingly entangled with redundant or irrelevant information in rapidly expanding interaction histories, while context capacity remains limited (Wu et al., 2025a; Zhao et al., 2026). Therefore, deciding which evidence to include within the limited context window has become a significant challenge in the context management (Wan et al., 2026; Ghulyani et al., 2026).

Recent efforts to address this challenge mainly manage interaction histories in two ways. Some methods compress or summarize long histories to fit limited context budgets (Pan et al., 2024; Wu et al., 2025a; Wan et al., 2026), while others organize past interactions into memories or experiences that agents can retrieve and update during reasoning (Zhao et al., 2024; Zhang et al., 2025; Yu et al., 2026). These approaches provide effective mechanisms for reducing, organizing, and accessing growing histories. However, evidence selection raises a more fundamental question: ***what makes a piece of evidence worth retaining for subsequent reasoning and decisions?*** Recent studies suggest

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

that semantic relevance and compactness capture important aspects of context quality, but may not fully reflect how individual evidence contributes to subsequent reasoning and decisions (Jiang et al., 2026; Jin et al., 2026). This motivates a different perspective: since agentic reasoning continually depends on environment feedback to guide subsequent decisions, we consider evidence valuable when it reduces the agent's uncertainty about such subsequent feedback.

Motivated by this perspective, we propose **S**electing **E**vidence by **E**ntropy **R**eduction (**SEER**), a novel and theoretically-grounded context management framework that selects evidence by its contribution to reducing the agent's uncertainty (quantified by entropy). Specifically, SEER formulates context management as an Information Bottleneck (IB) problem, which explicitly balances the effectiveness and compactness of contexts. Based on this formulation, SEER evaluates each evidence by its marginal reduction in the agent's entropy over subsequent environment feedback and dynamically constructs compact contexts enriched with information that supports subsequent reasoning.

Our Main Contributions. (1) We introduce a new perspective for context management in agentic reasoning by formulating evidence selection as an IB objective, which provides a principled criterion to balance context compactness and effectiveness. (2) We develop SEER, an entropy-driven evidence selection framework that prioritizes evidence according to its marginal reduction in the agent's uncertainty over subsequent environment feedback. (3) We design an efficient evidence evaluation mechanism that estimates uncertainty reduction without repeatedly executing candidate-induced actions, enabling practical context construction during long-horizon interactions. (4) We conduct extensive experiments on ExCyTIn-Bench, AppWorld, SWE-Bench, and WebArena with Qwen3-8B, Qwen3-32B, Llama-3.1-8B-Instruct, and GPT-4.1-nano. SEER significantly outperforms all baselines across multiple model scales and families, improving over the strongest baselines by up to **9.4 points**.

2 Preliminaries

Interactive Episodes and Notations. An LLM-based agent solves a task (described by q) through a multi-step interaction episode. At step t, the agent receives environment feedback x_t ($x_1 = q$) and then produces an action a_t. The episode lasts for T steps or ends earlier upon termination. Let the interaction history up to step t be $h_t = (q, a_1, x_2, a_2, \ldots, x_t)$. At step t, the agent uses its base LLM to generate the action a_t conditioning on a context C_t derived from h_t.

Cost-Aware Context Construction. We quantify context compactness by a length measure $\mathrm{cost}(C_t)$ (e.g., token count). A convenient abstraction is to view context construction as a mapping $C_t = \phi(h_t)$, where $\phi(\cdot)$ outputs a concise representation of the interaction history. In practice, a hard context window can be enforced by truncation if needed.

Information-Theoretic Background. For a discrete random variable U with probability mass function $p(U)$, the Shannon entropy is $H(U) = -\sum_u p(u) \log p(u)$. For a pair (U, V), the conditional entropy is $H(U \mid V) = -\sum_v p(v) \sum_u p(u \mid v) \log p(u \mid v)$. The mutual information (MI) is $I(U;V) = H(U) - H(U \mid V) = H(V) - H(V \mid U)$. The Information Bottleneck (IB) principle (Tishby et al., 2000) formalizes learning a representation C that preserves information about a relevance variable Y while controlling representational capacity. A classical Lagrangian form is

$$\max_{C} \; I(C;Y) - \beta\, I(C;X), \tag{1}$$

where $\beta > 0$ controls the relevance-compression trade-off.

3 Selecting Evidence by Entropy Reduction

The key idea of SEER is to retain evidence according to how much it reduces the agent's uncertainty about subsequent environment feedback relative to the context length it introduces. We first use the Information Bottleneck (IB) principle to formalize the trade-off between preserving useful information and keeping the context compact. This formulation naturally leads to an evidence-selection criterion based on marginal entropy reduction per unit length. To evaluate this criterion efficiently at test time, SEER predicts the proxy-outcome distribution under a hypothetical next action, allowing it to estimate the value of candidate evidence without repeatedly interacting with the environment.

The remainder of this section develops SEER in four parts. **Section 3.1** derives the cost-aware IB formulation and motivates proxy entropy as the quantity for measuring evidence value. **Section 3.2**

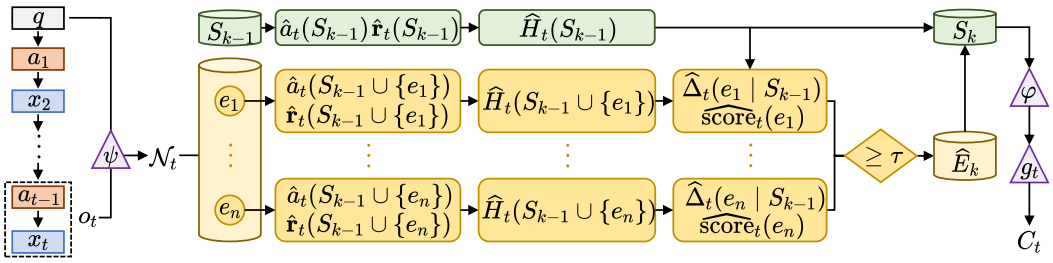


Figure 1: **Illustration of SEER at step t.** Given the history $h_t = (q, a_1, x_2, \ldots, a_{t-1}, x_t)$, SEER extracts new evidence items $\mathcal{N}_t$ from the task description q and latest interaction $o_t = (a_{t-1}, x_t)$ using ψ. For each candidate pool S, it computes a hypothetical action $\hat{a}_t(S)$, score vector $\hat{\mathbf{r}}_t(S)$, and proxy uncertainty $\widehat{H}_t(S)$. SEER uses these estimates to score each evidence item by marginal uncertainty reduction per cost, perform thresholded admission, update the evidence pool, and construct the context C_t using an optional curator φ and formatter g_t. Appendix E.4 reports inference-time costs, while Appendices E.5 and E.6 analyze robustness to taxonomy design and prompt variants.

turns context optimization into online evidence selection and derives the gain-per-cost admission rule. **Section 3.3** introduces the action-conditioned uncertainty estimator used to evaluate candidate evidence before execution. Finally, **Section 3.4** provides a concise theoretical characterization of thresholded admission and estimation error. **Figure 1** illustrates the overall framework.

3.1 Cost-Aware Information Bottleneck Formulation

At each step t, an effective context $C_t = \phi(h_t)$ should preserve the evidence needed for subsequent decisions while remaining compact (Mei et al., 2025). We formalize this trade-off through the Information Bottleneck (IB) principle (Tishby et al., 2000), which naturally captures this goal by preserving information relevant to a target variable while compressing the original representation. This view follows the standard belief-state formulation of partially observable decision making, where a compact state can summarize the interaction history without losing information needed for optimal action selection (Kaelbling et al., 1998). Recent LLM-agent studies similarly represent interaction histories through task-sufficient belief or world states for subsequent planning (Kim & Hwang, 2025; Kim et al., 2025). Motivated by this view, we make the following assumption.

Assumption 3.1 (Decision sufficiency (Kaelbling et al., 1998))**.** There exists a latent relevance variable Y_t that is sufficient for optimal decision-making at step t, such that the optimal action depends on the interaction history h_t only through Y_t.

Under Assumption 3.1, a desirable context should preserve information about Y_t while remaining compact. In the classical IB formulation, compactness is controlled by $I(C_t; h_t)$, which measures the information rate or description complexity retained from the input (Tishby et al., 2000; Zaidi et al., 2020). For explicit textual contexts, however, the directly constrained resource is the finite context window, and prior work commonly measures this resource by the number of retained tokens (Jiang et al., 2024; Li et al., 2025). Thus, we use token length as a computable context cost and obtain

$$\max_{\phi} \; I(C_t; Y_t) - \lambda \,\mathrm{cost}(C_t) \quad \text{s.t.} \quad C_t = \phi(h_t), \tag{2}$$

where $\lambda > 0$ controls the relevance-compactness trade-off. The relevance term determines *what information* the context should preserve, while $\mathrm{cost}(C_t)$ controls how much context it occupies.

From latent relevance to an observable proxy. The latent relevance variable Y_t is unavailable at test time. We therefore use an observable proxy outcome $\widetilde{Y}_{t+1} \in \mathcal{Y}$ that summarizes the functional type of subsequent environment feedback after the step-t action. For example, $\widetilde{Y}_{t+1}$ can distinguish successful execution, empty results, schema errors, and other functional outcomes. Appendix B.1.2 provides the full taxonomy, and Appendix E.5 studies its robustness.

Replacing Y_t with $\widetilde{Y}_{t+1}$ gives $I(C_t; \widetilde{Y}_{t+1}) - \lambda \,\mathrm{cost}(C_t)$. Following the standard mutual-information decomposition, $I(C_t; \widetilde{Y}_{t+1}) = H(\widetilde{Y}_{t+1}) - H(\widetilde{Y}_{t+1} \mid C_t)$. At a fixed step, $H(\widetilde{Y}_{t+1})$ remains constant with respect to the context mapping ϕ, so maximizing proxy mutual information reduces to minimizing

its conditional entropy:

$$\min_{\phi} \quad H(\widetilde{Y}_{t+1} \mid C_t) + \lambda \,\mathrm{cost}(C_t) \quad \text{s.t.} \quad C_t = \phi(h_t), \tag{3}$$

which favors compact contexts that reduce the agent's uncertainty over subsequent environment feedback, i.e., $H(\widetilde{Y}_{t+1} \mid C_t)$.

Why reduce proxy entropy? Agentic reasoning repeatedly uses environment feedback to update its understanding of the environment and guide subsequent decisions, as commonly modeled in partially observable sequential decision making (Kaelbling et al., 1998). From an information-theoretic perspective, information from the past is particularly useful when it reduces uncertainty about the future (Creutzig et al., 2009). Accordingly, when the proxy-outcome distribution remains diffuse, the current context leaves several qualitatively different subsequent outcomes plausible. Evidence that concentrates this distribution resolves such uncertainty and provides the agent with more predictive information for subsequent reasoning.

Importantly, SEER reduces uncertainty by selecting *evidence for the context*, rather than by selecting the action with the most predictable outcome. It *does not* optimize an action-level objective such as $\min_a H(\widetilde{Y}_{t+1} \mid C_t, a)$. Instead, SEER only changes the information provided to the agent through the context. After SEER constructs the context, the agent generates its actual action normally and remains free to explore when useful.

▷ **Empirical Support.** We provide three complementary analyses that support proxy-uncertainty reduction as the criterion for evidence selection. **(i) Outcome analysis.** Across step-level and episode-level evaluations, lower proxy entropy consistently accompanies stronger downstream outcomes. The same trend remains within different horizon and difficulty groups, reducing the possibility that it arises simply from shorter or easier tasks (Appendix E.1). **(ii) Matched-budget evidence selection.** As analyzed in Section 4.3, we compare different evidence selectors under nearly identical retained-evidence budgets. On ExCyTIn-Bench with Qwen3-32B, replacing Length-Matched Random with SEER reduces realized proxy entropy from 1.38 to 1.13 while improving ACC from 16.9 to 25.7. A retrospective Rollout Oracle that selects evidence using realized rollout outcomes further improves both quantities. The consistent ordering from Random to SEER and then to the Rollout Oracle supports the connection between lower proxy entropy and more useful evidence selection. **(iii) Admission-threshold analysis.** We further vary the admission threshold to directly control how much evidence SEER retains. The default threshold τ achieves both the lowest proxy entropy (1.13) and the strongest ACC/RWD ($25.7/26.1$), while either admitting more evidence (0.50τ) or filtering more aggressively (1.50τ) increases proxy entropy and degrades performance. This shows that the gains arise from selecting evidence by uncertainty reduction rather than simply retaining more or less context. Section 4.3 reports the complete results across Qwen3-8B and Qwen3-32B.

3.2 From Context Optimization to Online Evidence Selection

Eq. (3) optimizes over unrestricted context mappings $C_t = \phi(h_t)$. To make the problem tractable, we restrict C_t to a formatted subset of evidence extracted from the interaction history. At step t, SEER maintains an episode-level evidence pool S_{t-1} and incrementally updates it to S_t. Let $o_t = (a_{t-1}, x_t)$ denote the latest interaction. We construct the context as $C_t = g_t(S_t) \triangleq g(q, o_t, S_t)$, where $g(\cdot)$ is a deterministic formatter and $g_t(S) \triangleq g(q, o_t, S)$ fixes (q, o_t) at step t. For evidence selection, we define $\mathrm{cost}(S) \triangleq \sum_{e \in S} \mathrm{cost}(e)$, where $\mathrm{cost}(e)$ measures the token length contributed by e. Since q and o_t remain fixed across candidate pools, their token lengths remain constant during candidate comparison.

Evidence extraction. We apply an extractor ψ to the latest interaction and obtain a set of candidate evidence $\mathcal{N}_t = \psi(q, o_t)$. Each $e \in \mathcal{N}_t$ is a separate piece of evidence with $\mathrm{cost}(e) > 0$ that SEER can retain or discard independently. We implement ψ with the base LLM, following the general practice of extracting compact natural-language knowledge from agent experience (Zhao et al., 2024). Appendix B.1.1 describes the evidence taxonomy, while Appendices B.2 and E.6 provide the prompt and robustness analysis.

Marginal entropy reduction. Different candidate contexts may induce different next actions, and these actions in turn affect subsequent environment feedback in interactive decision processes (Kaelbling et al., 1998; Yao et al., 2022). We therefore evaluate each candidate context together with the action it induces. Let A_t denote the step-t action variable, and let $a_t(S)$ denote the action generated

by the agent under the candidate context $g_t(S)$. The agent's original policy maps each candidate context to its corresponding action, so the context remains the quantity optimized by SEER while the induced action determines the relevant proxy-outcome distribution. We define the corresponding action-conditioned proxy uncertainty as $U_t^{\mathrm{act}}(S) \triangleq H\Big(\widetilde{Y}_{t+1} \mid C_t = g_t(S), A_t = a_t(S)\Big)$. At the current interaction state, this gives the following cost-regularized objective over evidence subsets:

$$F_t^{\lambda}(S) \triangleq -U_t^{\mathrm{act}}(S) - \lambda\,\mathrm{cost}(S). \tag{4}$$

For a candidate $e \in \mathcal{N}_t$, we measure how much adding it reduces proxy uncertainty relative to a baseline pool S:

$$\Delta_t^{\mathrm{act}}(e \mid S) \triangleq U_t^{\mathrm{act}}(S) - U_t^{\mathrm{act}}(S \cup \{e\}). \tag{5}$$

We then score the candidate by its uncertainty reduction per unit length:

$$\mathrm{score}_t^{\mathrm{act}}(e \mid S) \triangleq \Delta_t^{\mathrm{act}}(e \mid S)/\mathrm{cost}(e). \tag{6}$$

Thresholded evidence selection. Marginal gain-per-cost criteria and threshold rules are widely used for cost-aware and streaming subset selection (Leskovec et al., 2007; Badanidiyuru et al., 2014). For our objective, the threshold rule follows directly from the marginal change in Eq. (4). For any $e \notin S$, $F_t^{\lambda}(S \cup \{e\}) - F_t^{\lambda}(S) = \Delta_t^{\mathrm{act}}(e \mid S) - \lambda\,\mathrm{cost}(e)$. Thus, setting $\tau = \lambda$ gives

$$\mathrm{score}_t^{\mathrm{act}}(e \mid S) \geq \tau \quad \Longleftrightarrow \quad F_t^{\lambda}(S \cup \{e\}) \geq F_t^{\lambda}(S). \tag{7}$$

The threshold rule thus retains evidence whose marginal reduction in proxy uncertainty justifies its additional context cost. Given $\mathcal{N}_t$, SEER evaluates every candidate against the same baseline pool S_{t-1}. This common baseline makes candidate scores directly comparable and independent of the order in which candidates are evaluated. The resulting oracle admission rule is

$$E_t \triangleq \Big\{e \in \mathcal{N}_t : \mathrm{score}_t^{\mathrm{act}}(e \mid S_{t-1}) \geq \tau\Big\}, \qquad S_t \leftarrow S_{t-1} \cup E_t. \tag{8}$$

SEER discards the remaining candidates in $\mathcal{N}_t$ after step t. Eq. (7) characterizes the marginal contribution of each candidate relative to this common baseline. Section 3.4 further studies how these individual gains combine under simultaneous admission when candidates contain overlapping information.

Curator-based consolidation. Long interaction histories can still accumulate redundant or verbose information, motivating periodic compression or consolidation in agent memory and context management (Wu et al., 2025a; Park et al., 2023). After every preset number of newly admitted candidates, SEER invokes a curator φ to consolidate the pool: $S_t \leftarrow \varphi(S_t)$. We instruct the curator to merge repetitive evidence and shorten verbose descriptions while preserving the information retained in the pool. Finally, SEER constructs $C_t = g_t(S_t)$ and passes it to the agent for the next action. Appendix B.2 provides the curator prompt, and Appendix E.6 studies its robustness.

3.3 Efficient Action-Conditioned Proxy Uncertainty Estimation

Section 3.2 evaluates evidence through the action-conditioned proxy uncertainty $U_t^{\mathrm{act}}(S)$. Estimating this quantity directly from the environment would require counterfactual executions for every candidate context and repeated rollouts when estimating the corresponding outcome distribution. Such candidate-wise interaction introduces substantial additional cost during online reasoning. We instead estimate the proxy-outcome distribution before execution.

Hypothetical action and outcome prediction. Predicting future outcomes conditioned on candidate actions is widely used in model-based planning to evaluate decisions before environment interaction (Hafner et al., 2019). For any candidate pool S, SEER provides the base LLM with the task q, the latest interaction $o_t = (a_{t-1}, x_t)$, and the evidence in S. The LLM jointly produces (i) a hypothetical next action $\hat{a}_t(S)$ and (ii) an unnormalized score vector $\hat{\mathbf{r}}_t(S) \in \mathbb{R}^{|\mathcal{Y}|}$ over the proxy outcomes:

$$(\hat{a}_t(S), \hat{\mathbf{r}}_t(S)) \leftarrow \mathrm{LLM}(q, o_t, S). \tag{9}$$

The hypothetical action provides a concrete action condition for predicting the subsequent environment feedback under the candidate context. We use the resulting scores only to evaluate the candidate before environment interaction.

We normalize the scores into an estimated proxy-outcome distribution:

$$\hat{p}_t(y \mid S, o_t) \triangleq \frac{\exp(\hat{\mathbf{r}}_t(S)[y])}{\sum_{y' \in \mathcal{Y}} \exp(\hat{\mathbf{r}}_t(S)[y'])}, \qquad y \in \mathcal{Y}. \tag{10}$$

Because the model scores every proxy outcome, the resulting distribution can represent uncertainty by assigning probability mass to multiple plausible outcomes. This explicit score elicitation is also consistent with prior findings that LLMs can provide useful uncertainty estimates through verbalized confidence scores (Tian et al., 2023). Section 4.3 further shows that this structured proxy-outcome prediction provides a stronger evidence-selection signal than uncertainty derived from token-level log-probabilities.

Proxy-entropy estimation. We quantify the uncertainty of the estimated proxy-outcome distribution using its entropy $\widehat{H}_t(S) \triangleq -\sum_{y \in \mathcal{Y}} \hat{p}_t(y \mid S, o_t) \log \hat{p}_t(y \mid S, o_t)$. Here, $\widetilde{Y}_{t+1}$ specifies the proxy outcome whose uncertainty we measure, while $\widehat{H}_t(S)$ measures the uncertainty predicted under the hypothetical action proposed from S. A concentrated distribution yields low entropy, whereas a diffuse distribution over several plausible outcomes yields high entropy.

Estimated marginal gains. We use $\widehat{H}_t(\cdot)$ to estimate the marginal uncertainty reduction in Eq. (5). For every candidate $e \in \mathcal{N}_t$, we compute

$$\widehat{\Delta}_t(e \mid S_{t-1}) \triangleq \widehat{H}_t(S_{t-1}) - \widehat{H}_t(S_{t-1} \cup \{e\}), \tag{11}$$

and its estimated gain-per-cost score

$$\widehat{\mathrm{score}}_t(e) \triangleq \widehat{\Delta}_t(e \mid S_{t-1})/\mathrm{cost}(e). \tag{12}$$

All candidates share the same baseline estimate $\widehat{H}_t(S_{t-1})$, which SEER computes once at step t. SEER then evaluates every candidate against this common baseline and admits

$$\widehat{E}_t \triangleq \Big\{ e \in \mathcal{N}_t : \widehat{\mathrm{score}}_t(e) \geq \tau \Big\}, \qquad S_t \leftarrow S_{t-1} \cup \widehat{E}_t. \tag{13}$$

This estimator therefore converts the action-conditioned uncertainty objective in Section 3.2 into a practical evidence score using only LLM predictions, avoiding candidate-wise counterfactual interactions with the environment.

3.4 Properties of Entropy-Guided Selection

We next examine how the evidence-selection rule relates to the cost-regularized objective in Section 3.2. Under oracle uncertainty, the threshold rule exactly identifies candidates with nonnegative marginal gains in the objective. When SEER estimates this uncertainty at test time, bounded estimation error preserves this property up to a controlled additive error. Together, these results establish a direct objective-to-algorithm connection for SEER. Appendix C provides the complete theoretical analysis, proofs, and empirical diagnostics.

Thresholding aligns with the cost-regularized objective. Recall from Section 3.2 that $F_t^\lambda(S) = -U_t^{\mathrm{act}}(S) - \lambda\,\mathrm{cost}(S)$. For any candidate e, we have $F_t^\lambda(S \cup \{e\}) - F_t^\lambda(S) = \Delta_t^{\mathrm{act}}(e \mid S) - \lambda\,\mathrm{cost}(e)$. Therefore, with $\tau = \lambda$, any candidate satisfying $\Delta_t^{\mathrm{act}}(e \mid S)/\mathrm{cost}(e) \geq \tau$ gives a nonnegative marginal gain in the oracle cost-regularized objective.

Bounded estimation error preserves near-improvement. At test time, SEER replaces $U_t^{\mathrm{act}}(S)$ with $\widehat{H}_t(S)$ from Section 3.3. Suppose that, for the candidate pools evaluated at step t, $|\widehat{H}_t(S) - U_t^{\mathrm{act}}(S)| \leq \varepsilon_H$. Then, for every $e \in \mathcal{N}_t$, we have $|\widehat{\Delta}_t(e \mid S_{t-1}) - \Delta_t^{\mathrm{act}}(e \mid S_{t-1})| \leq 2\varepsilon_H$. Consequently, if SEER admits e because $\widehat{\Delta}_t(e \mid S_{t-1})/\mathrm{cost}(e) \geq \tau$ with $\tau = \lambda$, then $F_t^\lambda(S_{t-1} \cup \{e\}) - F_t^\lambda(S_{t-1}) \geq -2\varepsilon_H$. Thus, the practical admission rule retains the oracle objective-improvement property up to the entropy-estimation error. Appendix C extends these per-candidate results to simultaneous admission under controlled evidence redundancy and further analyzes proxy confidence and action-level behavior.

4 Experiments

4.1 Experimental Settings

Benchmarks and Evaluation Protocol. We focus on challenging benchmarks with complex environments, where unfamiliar and evolving feedback limits the usefulness of parametric knowledge

Table 1: **Performance of SEER and baselines on ExCyTIn-Bench and AppWorld.** Since the default minimal implementation on AppWorld is ReAct, we do not report Direct on AppWorld. Additional results on more models and benchmarks are provided in Appendix E.2, paired statistical robustness analyses in Appendix E.3, and detailed inference-time cost statistics in Appendix E.4.

Methods	Qwen3-8B						Qwen3-32B					
	ExCyTIn-Bench		AppWorld-N		AppWorld-C		ExCyTIn-Bench		AppWorld-N		AppWorld-C	
	ACC	RWD	TGC	SGC	TGC	SGC	ACC	RWD	TGC	SGC	TGC	SGC
Direct	8.7	10.2	–	–	–	–	11.9	13.1	–	–	–	–
LLMLingua-2	5.9	8.1	10.1	1.8	1.4	0.0	15.6	17.2	13.1	3.6	4.8	1.4
ReAct	10.9	11.8	17.3	5.4	4.8	0.7	15.8	17.3	18.4	7.1	9.6	0.7
ExpeL	12.2	13.4	17.9	5.4	5.7	1.3	16.2	17.7	19.1	7.1	8.9	2.2
ACE	14.6	16.0	17.9	3.6	3.8	0.7	16.3	17.7	18.4	7.1	11.5	2.2
SEER	**23.0**	**24.5**	**21.4**	**8.9**	**8.9**	**2.9**	**25.7**	**26.1**	**24.6**	**8.9**	**13.2**	**5.8**

and makes agents highly dependent on interaction contexts. We evaluate SEER on two challenging interactive-agent benchmarks: ExCyTIn-Bench (Wu et al., 2025b) and AppWorld (Trivedi et al., 2024), with additional results on SWE-Bench (Yang et al., 2025b) and WebArena (Zhou et al., 2023) in Appendix E.2. **ExCyTIn-Bench** evaluates long-horizon cyber threat investigation through interactive MySQL queries over noisy enterprise security logs, requiring agents to accumulate sparse evidence and track constraints across long interaction histories. We report final-answer accuracy (ACC) and verifiable reward (RWD). **AppWorld** evaluates stateful multi-step API interaction in a controllable app ecosystem, where agents must iteratively act on evolving application states; AppWorld-C further tests generalization to unseen apps. We report Task Goal Completion (TGC) and Scenario Goal Completion (SGC). Further benchmark details are provided in Appendix A.1.

Models and Inference Configuration. We use Qwen3-8B and Qwen3-32B (Yang et al., 2025a) as base LLMs, with additional results using Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and GPT-4.1-nano (OpenAI, 2025) in Appendix E.2. For fair comparison, all methods use the same decoding setup with temperature 0.0, following (Wu et al., 2025b; Trivedi et al., 2024). Unless otherwise specified, we use each model's default chat template and the benchmark-specific stop conditions provided by each environment. Further details are provided in Appendix A.2.

Baselines and Fairness. We compare SEER with five baselines covering direct inference, prompt compression, reasoning-acting prompting, and experience-/context-adaptation. **Direct** uses the raw interaction history without additional context engineering. **LLMLingua-2** (Pan et al., 2024) compresses prompts with a BERT-based token classifier. **ReAct** (Yao et al., 2022) interleaves reasoning traces and tool actions. **ExpeL** (Zhao et al., 2024) augments ReAct with retrieved successful experiences and distilled insights from past trajectories. **ACE** (Zhang et al., 2025) iteratively generates, reflects, and curates an evolving playbook-style context. For fairness, all methods share the same tool set, maximum interaction steps, and context-length budget, and each baseline is implemented within its intended design scope without extra mechanisms. Further details are provided in Appendix A.3.

Implementation of SEER. For each benchmark, we define 3 evidence types and 7 outcome-proxy types, with taxonomy details and robustness analyses in Appendices B.1 and E.5. At each step, we extract up to 5 evidence units, and trigger curator consolidation whenever 10 new evidence units have been added since the last consolidation. The extractor ψ, curator φ, hypothetical action $\hat{a}_t$, and score vector $\hat{\mathbf{r}}_t$ are implemented with the agent's base model using different prompts. Prompt templates for these components and the formatter g are given in Appendix B.2, with prompt robustness analyses in Appendix E.6. Experiments are conducted either on 4 or 8 A100/A800 GPUs or via LLM APIs.

4.2 Main Results

Table 1 reports the main results on ExCyTIn-Bench and AppWorld with Qwen3-8B and Qwen3-32B, with all metrics scaled to $[0, 100]$. First, both backbones still achieve relatively low absolute performance on these challenging interactive benchmarks, highlighting the difficulty of long-horizon agentic reasoning in complex environments. Second, SEER consistently outperforms all baselines across benchmarks, model scales, and metrics. The improvements are particularly large on ExCyTIn-Bench: compared with the strongest baselines, SEER improves ACC/RWD by 8.4/8.5 points with Qwen3-8B and 9.4/8.4 points with Qwen3-32B, supporting the effectiveness of entropy-reducing evidence selection for long-horizon agentic reasoning. SEER also yields consistent improvements

on AppWorld, including both the normal split and the more challenging unseen-app split. Third, LLMLingua-2 often underperforms other context-engineering baselines and can even trail Direct on ExCyTIn-Bench, suggesting that compression alone may discard decision-critical evidence. Finally, ExpeL and ACE generally improve over ReAct by incorporating experience- or reflection-driven context refinement, but still remain below SEER, suggesting that explicitly selecting evidence by its estimated marginal uncertainty reduction makes more effective use of the limited context budget.

Performance under matched inference budgets. Since different LLM calls can vary substantially in sequence length and runtime, we further compare methods under matched total-token and wall-clock budgets on AppWorld-N. Table 2 shows two consistent observations. (1) Under matched total-token budgets, SEER achieves the strongest performance on both backbones, reaching 20.8/7.1 TGC/SGC with Qwen3-8B and 5.4/1.8 with GPT-4.1-nano, compared with 17.9/5.4 and 4.2/0.0 for the strongest competing results, respectively. (2) The advantage persists under matched wall-clock budgets: SEER reaches 19.0/8.9 with Qwen3-8B and 5.6/3.6 with GPT-4.1-nano, outperforming ExpeL and ACE in both settings. These results show that the gains persist after controlling for both total token usage and runtime, rather than arising solely from additional inference-time computation.

Table 2: **Performance under matched inference budgets on AppWorld-N.** We report TGC/SGC. All methods use the same per-episode total-token budget in Panel (a) and wall-clock budget in Panel (b).

Backbone	Budget	ExpeL	ACE	SEER
(a) Matched total-token budget				
Qwen3-8B	240K tokens	17.9/5.4	16.8/3.6	**20.8/7.1**
GPT-4.1-nano	65K tokens	4.2/0.0	2.6/0.0	**5.4/1.8**
(b) Matched wall-clock budget				
Qwen3-8B	25 s	14.8/3.6	17.9/3.6	**19.0/8.9**
GPT-4.1-nano	4.5 s	4.2/0.0	3.0/1.8	**5.6/3.6**

Additional Analyses. Beyond the mechanism analyses in Section 4.3, we provide further complementary results in the appendix. **Appendix E.1** examines the relationship between proxy entropy and downstream outcomes through step-level, episode-level, horizon-stratified, and difficulty-stratified diagnostics. **Appendix E.2** extends evaluation to SWE-Bench, WebArena, Llama-3.1-8B-Instruct, and GPT-4.1-nano, while **Appendix E.3** reports paired statistical analyses across the evaluated benchmarks. **Appendix E.4** analyzes inference-time cost, and **Appendix E.7** examines evidence-pool growth on difficult and long-horizon episodes. **Appendices E.5, E.6, and E.8** study robustness to proxy-taxonomy design, prompt variants, and heterogeneous module assignments. **Appendices E.9 and E.10** analyze intermediate-stage reliability, error propagation, proxy calibration, and hypothetical-action consistency. Finally, **Appendix D** presents a representative AppWorld case study.

4.3 Ablations and Insights

Table 3 compares controlled variants of SEER. We make the following observations. (1) Replacing the evidence pool with recent-history summaries substantially degrades performance. *Summarized*, which keeps only a summary of the latest interaction, achieves 13.4/14.3 and 14.8/15.3 ACC/RWD on Qwen3-8B and Qwen3-32B, respectively; *Recent-5 Summary*, which summarizes the five most recent interactions, improves over it but still remains far below SEER. This suggests that short recent-history summaries cannot reliably preserve useful evidence accumulated throughout long interactions. (2) The evidence-admission criterion is also critical. *Length-Matched Random* randomly admits evidence while matching SEER in retained-evidence length and admission count within 2%, *LLM Judge* directly judges candidate usefulness using the same base model and candidate-level scoring budget, and *Flooded* retains all extracted evidence. SEER consistently outperforms all three variants; on Qwen3-32B, it reaches 25.7/26.1 ACC/RWD, compared with 16.9/18.1 for *Length-Matched*

Table 3: **Controlled variants of SEER on ExCyTIn-Bench with Qwen3-8B and Qwen3-32B.** The bold font indicates the best result and an underline indicates the second-best result.

Variant	Qwen3-8B		Qwen3-32B	
	ACC	RWD	ACC	RWD
SEER	**23.0**	**24.5**	**25.7**	**26.1**
Summarized	13.4	14.3	14.8	15.3
Recent-5 Summary	15.8	17.0	16.9	17.8
Length-Matched Random	16.1	17.4	16.9	18.1
LLM Judge	18.1	19.6	<u>18.9</u>	<u>20.1</u>
Flooded	18.7	<u>20.5</u>	17.5	18.4
Probabilized	17.8	18.5	16.7	17.5
Anchored	<u>19.0</u>	20.0	18.1	18.9

Table 4: **Admission-threshold sweep on ExCyTIn-Bench with Qwen3-32B.** The default threshold τ achieves the lowest realized proxy entropy together with the strongest ACC/RWD.

Threshold	Context Tokens / Ep.	Admitted Evidence / Ep.	Proxy Entropy ↓	ACC / RWD
0.50τ	3.14K	25.9	1.20	24.7 / 25.0
0.75τ	2.62K	21.6	1.15	25.3 / 25.8
τ	2.18K	17.8	**1.13**	**25.7 / 26.1**
1.25τ	1.82K	14.4	1.20	24.8 / 25.1
1.50τ	1.51K	11.9	1.29	23.8 / 25.0

Random, 18.9/20.1 for *LLM Judge*, and 17.5/18.4 for *Flooded*. Moreover, *Flooded* uses over three times as much retained evidence, showing that the gains come from entropy-guided selection rather than retaining more evidence or spending more scoring calls. (3) The uncertainty estimator further contributes to performance. *Probabilized*, which replaces the prompted proxy-outcome distribution with token-level log-probabilities, drops to 17.8/18.5 and 16.7/17.5 on Qwen3-8B and Qwen3-32B, respectively. *Anchored*, which evaluates all candidates under the same proposed action, performs better but still remains below SEER, especially on Qwen3-32B (18.1/18.9 vs. 25.7/26.1). These results support explicitly predicting proxy-outcome uncertainty while allowing different candidate contexts to induce different proposed actions.

4.3.1 Understanding Entropy-Guided Selection

We first examine whether lower proxy entropy corresponds to better evidence selection. Fig. 2 compares three selectors under nearly identical retained-evidence budgets: *Length-Matched Random* randomly selects evidence while matching SEER in retained length, whereas *Rollout Oracle* retrospectively selects evidence using realized rollout outcomes. We make two observations. (1) SEER consistently lowers proxy entropy and improves ACC over Random on both backbones; for Qwen3-32B, entropy drops from 1.38 to 1.13 while ACC rises from 16.9 to 25.7. (2) Rollout Oracle further improves both quantities, supporting the connection between lower proxy entropy and more useful evidence selection.

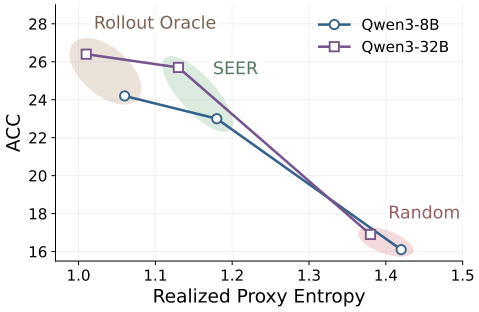


Figure 2: **Matched-budget evidence selection on ExCyTIn-Bench.** Under nearly matched retained-evidence budgets, lower realized proxy entropy consistently corresponds to stronger downstream performance across Length-Matched Random, SEER, and the Retrospective Rollout Oracle.

We next study the admission threshold in Table 4. Lower thresholds admit more evidence, while higher thresholds retain less; neither direction monotonically improves performance. The default τ achieves both the lowest proxy entropy (1.13) and the best ACC/RWD (25.7/26.1), while either admitting too much (0.50τ) or filtering too aggressively (1.50τ) degrades both uncertainty and performance. This supports selecting evidence by uncertainty reduction rather than simply retaining more or less context.

5 Related Work

Due to the 9-page limit for the main paper at submission, we place the related work in Appendix F. If accepted, the limit increases to 10 pages, and we will move the related work into the main paper.

6 Conclusion

We propose SEER, a theoretically grounded context management framework that selects evidence by how much it reduces the agent's uncertainty about subsequent environment feedback. Starting from an Information Bottleneck formulation, SEER balances context effectiveness and compactness through marginal entropy reduction, and efficiently estimates this quantity using hypothetical actions without additional environment executions. Our theoretical analysis connects the resulting thresholded admission rule to the underlying cost-regularized objective. Experiments on challenging benchmarks show that SEER significantly outperforms all baselines across multiple model scales and families in terms of accuracy and task completion.

AI USE STATEMENT

We used generative AI tools to polish author-written text for grammar, clarity, and readability. The authors reviewed all AI-assisted edits and take full responsibility for the final content of this paper. The LLMs evaluated or invoked in our experiments are objects of study and components of the agent systems investigated in this work, rather than generative AI tools used to produce the scientific content of the paper.

ETHICS STATEMENT

We conduct this work in accordance with the ICLR Code of Ethics. Our experiments evaluate LLM-based agents in existing controlled benchmark environments and do not involve human-subject studies or the collection of new personal data. Improving the reliability of tool-using agents may have dual-use implications, particularly in sensitive settings such as cybersecurity. Our method does not provide agents with additional tools, permissions, or external access beyond those defined by the evaluation environments. Practical deployment in sensitive settings should therefore incorporate appropriate access control, privacy protection, logging, and human oversight.

REPRODUCIBILITY STATEMENT

We provide detailed information to support reproducibility throughout the paper and appendices. Section 4.1 and Appendix A describe the benchmarks, evaluation protocols, backbone models, inference settings, baselines, and comparison controls; Appendix B provides the implementation details of SEER, including the evidence and proxy-outcome taxonomies and prompt templates; Appendix C provides the theoretical assumptions, derivations, and proofs; and Appendix E reports additional evaluations, statistical analyses, inference-time costs, and robustness studies. The experimental setup also reports the key implementation choices and computational resources used in our experiments.

REFERENCES

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017.

Anthropic. The future of ai at work: Introducing cowork. `https://www.anthropic.com/webinars/future-of-ai-at-work-introducing-cowork`, January 2026. Accessed: 2026-09-17.

Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 671–680, 2014.

Arthur Chen, Zuxin Liu, Jianguo Zhang, Akshara Prabhakar, Zhiwei Liu, Shelby Heinecke, Silvio Savarese, Victor Zhong, and Caiming Xiong. Test-time adaptation for llm agents via environment interaction. In *International Conference on Learning Representations*, volume 2026, pp. 39256–39288, 2026.

Sadia Sultana Chowa, Riasad Alvi, Subhey Sadi Rahman, Md Abdur Rahman, Mohaimenul Azam Khan Raiaan, Md Rafiqul Islam, Mukhtar Hussain, and Sami Azam. From language to action: a review of large language models as autonomous agents and tool users. *Artificial Intelligence Review*, 59(2):71, 2026.

Felix Creutzig, Amir Globerson, and Naftali Tishby. Past-future information bottleneck in dynamical systems. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 79(4):041925, 2009.

Gelei Deng, Yi Liu, Yuekang Li, Ruozhao Yang, Xiaofei Xie, Jie Zhang, Han Qiu, and Tianwei Zhang. What makes a good llm agent for real-world penetration testing? *arXiv preprint arXiv:2602.17622*, 2026.

Manu Ghulyani, Arunabh Singh, Karan Bharadwaj, Ankit Nath, and Suranjan Goswami. Pacms: Submodular context selection as a pluggable engine for llm agents. *arXiv preprint arXiv:2606.20047*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. Magma: A multi-graph based agentic memory architecture for ai agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36848–36865, 2026.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024.

Yiqiao Jin, Kartik Sharma, Vineeth Rakesh, Yingtong Dou, Menghai Pan, Mahashweta Das, and Srijan Kumar. Sara: Selective and adaptive retrieval-augmented generation with context compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14508–14528, 2026.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. Reflact: World-grounded decision making in llm agents via goal-state reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33421–33453, 2025.

Minsoo Kim and Seung-won Hwang. Coex-co-evolving world-model and exploration. In *EMNLP (Findings)*, pp. 21629–21651, 2025.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 420–429, 2007.

Xiaochuan Li, Ryan Ming, Meng Chu, Shuai Shao, Rong Jin, and Chenyan Xiong. Acm: Agentic context management for long horizon tasks. *arXiv preprint arXiv:2607.23809*, 2026.

Zongqian Li, Yixuan Su, and Nigel Collier. 500xcompressor: Generalized prompt compression for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25081–25091, 2025.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, et al. A survey of context engineering for large language models. *arXiv preprint arXiv:2507.13334*, 2025.

OpenAI. Introducing gpt-4.1 in the api. `https://openai.com/index/gpt-4-1/`, April 2025. Accessed: 2026-05-06.

OpenAI. Chatgpt is now a partner for your most ambitious work. `https://openai.com/index/chatgpt-for-your-most-ambitious-work/`, July 2026. Accessed: 2026-09-17.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *ACL (Findings)*, 2024.

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

Fahim Tajwar, Yiding Jiang, Abitha Thankaraj, Sumaita Sadia Rahman, J Zico Kolter, Jeff Schneider, and Ruslan Salakhutdinov. Training a generally curious agent. *arXiv preprint arXiv:2502.17543*, 2025.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5433–5442, 2023.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pp. 1–5. Ieee, 2015.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024.

Guangya Wan, Mingyang Ling, Xiaoqi Ren, Rujun Han, Sheng Li, and Zizhao Zhang. Compass: Enhancing agent long-horizon reasoning with evolving context. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3360–3380, 2026.

Haoran Wang, Zhenyu Hou, Yao Wei, Jie Tang, and Yuxiao Dong. Swe-dev: Building software engineering agents with training and inference scaling. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 3742–3761, 2025a.

Jihong Wang, Jiamu Zhou, Weiming Zhang, Teng Wang, Weiwen Liu, Zhuosheng Zhang, Xingyu Lou, Weinan Zhang, Huarong Deng, and Jun Wang. Colorbrowseragent: Complex long-horizon browser agent with adaptive knowledge evolution. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, pp. 665–680, 2026a.

Tong Wang, Pei Xu, Shiyue Cao, Likun Yang, Daipeng Li, Jianbin Jiao, and Kaiqi Huang. Samem: State-aware memory as a fine-grained memory for llm agents in decision-making. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14691–14710, 2026b.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pp. 65882–65919, 2025b.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In *ICML 2025 Workshop on Computer Use Agents*, 2025.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, et al. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025a.

Yiran Wu, Mauricio Velazco, Andrew Zhao, Manuel Raúl Meléndez Luján, Srisuma Movva, Yogesh K Roy, Quang Nguyen, Roberto Rodriguez, Qingyun Wu, Michael Albada, et al. Excytin-bench: Evaluating llm agents on cyber threat investigation. *arXiv preprint arXiv:2507.14201*, 2025b.

Yong Wu, YanZhao Zheng, TianZe Xu, ZhenTao Zhang, YuanQiang Yu, JiHuai Zhu, Chao Ma, BinBin Lin, BaoHua Dong, HangCheng Zhu, et al. Contextbudget: Budget-aware context management for long-horizon search agents. *arXiv preprint arXiv:2604.01664*, 2026.

Weikai Xu, Chengrui Huang, Shen Gao, and Shuo Shang. Llm-based agents for tool learning: A survey: W. xu et al. *Data Science and Engineering*, 10(4):533–563, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents, 2025b. URL https://arxiv.org/abs/2504.21798.

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*, 2022.

Pei Yingjun and Hou Xinwen. Learning representations in reinforcement learning: An information bottleneck approach. *arXiv preprint arXiv:1911.05695*, 2019.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, 2026.

Abdellatif Zaidi, Iñaki Estella-Aguerri, and Shlomo Shamai. On the information bottleneck problems: Models, connections, applications and information theoretic views. *Entropy*, 22(2):151, 2020.

Minxing Zhang, Yi Yang, Roy Xie, Bhuwan Dhingra, Shuyan Zhou, and Jian Pei. Generalizability of large language model-based agents: A comprehensive survey. *ACM Computing Surveys*, 58(10): 1–44, 2026.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Beidi Zhao, Yaoqi Chen, Yuru Feng, Menghao Li, Qianxi Zhang, Baotong Lu, Jianan Lu, Zhirui Wang, Xinjiang Wang, Shusen Xu, et al. Mesa: Task-adaptive multi-structure evidence selection for long-horizon agent memory. *arXiv preprint arXiv:2608.10108*, 2026.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

A Experimental Details

A.1 Benchmarks and Evaluation Protocol

We evaluate our method on two challenging benchmarks that necessitate managing long-horizon interactions, noisy observations, and domain-specific tool usage.

- **ExCyTIn-Bench (Wu et al., 2025b)** assesses agents on end-to-end cyber threat investigation tasks. It simulates a realistic security operations center (SOC) environment comprising 57 log tables and recording 8 simulated multi-stage attack campaigns. Agents must perform interactive SQL querying to sift through noisy logs and identify indicators of compromise (IoCs). This setting also serves as a long-context stress test for test-time context construction. Unlike short single-turn tasks, an ExCyTIn-Bench episode requires the agent to accumulate information over a growing interaction history, including explored schemas, returned log entries, failed SQL attempts, unavailable tables or columns, and intermediate investigation findings. Without explicit context construction, such histories can quickly become verbose and noisy, forcing the agent to rely on truncation or coarse compression and increasing the risk of losing sparse but decision-critical evidence. SEER addresses this setting by maintaining an episode-level pool of concise evidence units and selectively preserving information that yields high marginal proxy-uncertainty reduction per unit cost.

  Metric: We report the reward (RWD), which assigns a score of 1.0 for a correct final answer and awards discounted partial credit for intermediate findings based on ground-truth investigation paths. We also report *Accuracy (ACC)*.

  Split: Evaluation is performed on the standard test split containing 589 questions.

- **AppWorld (Trivedi et al., 2024)** is a high-fidelity environment simulating a user's digital life across 9 controllable apps (e.g., Amazon, Venmo, Gmail) populated with realistic database records. In contrast to simple function calling, AppWorld requires agents to iteratively write *complex executable Python code* to invoke over 450 APIs and manage intricate state dependencies.

  Metric: Evaluation relies on robust *state-based unit tests* that verify the final database state rather than performing text matching. We report *Task Goal Completion (TGC)* and *Scenario Goal Completion (SGC)*.

  Split: We evaluate on the combined test set of 585 tasks, stratified into *AppWorld-N* (Test-N) and the more demanding *AppWorld-C* (Test-C). AppWorld-C specifically tests generalization by requiring the agent to utilize APIs from "unseen" apps absent from the few-shot exemplars.

A.2 Models and Inference Configuration

Base models. We use Qwen3-8B and Qwen3-32B (Yang et al., 2025a) as the base LLMs. We load the official HuggingFace checkpoints: `Qwen/Qwen3-8B`[1] and `Qwen/Qwen3-32B`[2]. We use the official tokenizer and the default chat template shipped with the checkpoints. For all methods, we run inference with the same decoding configuration for fair comparison, using a temperature of 0.0 as adopted in (Wu et al., 2025b; Trivedi et al., 2024).

A.3 Baselines and Fairness

Baselines. We compare SEER against five representative baselines.

- **Direct.** The backbone LLM executes actions using standard task instructions and tool schemas, with interaction history and tool outputs formatted without modification.
- **LLMLingua-2 (Pan et al., 2024).** A state-of-the-art prompt compression method utilizing a BERT-based token classification model. Instead of naive character-level truncation, LLMLingua-2 compresses the full prompt (instruction, schema, and history) to fit within the

[1] `https://huggingface.co/Qwen/Qwen3-8B`
[2] `https://huggingface.co/Qwen/Qwen3-32B`

effective token limits. This baseline evaluates whether task-agnostic semantic pruning offers advantages over simple truncation.

- **ReAct (Yao et al., 2022).** This baseline prompts the model to interleave natural language reasoning traces (Thoughts) with tool actions. We adopt the canonical `Thought-Action-Observation` format.
- **ExpeL (Zhao et al., 2024).** ExpeL equips the agent with an experiential learning module that maintains a pool of successful trajectories and distilled insights. At inference time, it retrieves the top-k most relevant insights based on vector similarity to augment the prompt.
- **ACE (Zhang et al., 2025).** ACE treats context as an evolving playbook optimized via a Generator, Reflector, and Curator. We initialize the playbook using the training split and the playbook is updated sequentially across benchmark tasks.

Fairness and evaluation controls. To ensure a rigorous comparison, we standardize the following settings: (1) **Backbone & Environment:** All methods utilize the same backbone model, tool definitions, API return formats, and maximum interaction steps per episode. (2) **Hyperparameters:** For baselines requiring specific parameters, we tune them on a held-out development set and freeze them for the final evaluation. (3) **Decoding:** We employ greedy decoding with a temperature of 0.0 to ensure deterministic reproducibility.

B Implementation of Our Method

B.1 Taxonomy Design

We use lightweight functional taxonomies to structure evidence units and proxy outcomes in SEER. Although these taxonomies are specified for each benchmark to match its action and feedback semantics, our robustness analysis in Appendix E.5 shows that **the method is not tied to a particular taxonomy granularity or benchmark-specific label set**. In particular, SEER remains effective under coarser and finer proxy-outcome partitions, and also remains stable when transferring the core proxy-taxonomy structure from ExCyTIn-Bench to AppWorld. Below, we describe the default taxonomy used in the main experiments.

B.1.1 Evidence Three-Way Classification

SEER categorizes extracted evidence into three types (see below), each with distinct semantics and priority. The design is motivated by three considerations. (1) **Efficient retrieval:** grouping by type helps the model quickly locate the required information without scanning the entire context. (2) **Priority ordering:** constraints are surfaced first to clarify "what cannot be done," reducing ineffective exploration and preventing repeated invalid actions. (3) **Curation efficiency:** evidence units within the same type tend to be semantically closer, which makes curator-based merging and condensation more reliable and efficient.

- **Constraint** ([CONSTRAINT]):
  - ExCyTIn-Bench: Limitations or constraints on future actions, including schema restrictions, permission limitations, or unavailable resources.
  - AppWorld: API limitations, parameter restrictions, or unavailable resources.
- **FactLink** ([FACT]):
  - ExCyTIn-Bench: Confirmed state facts or data, including table structures, data values, and counts.
  - AppWorld: Confirmed application state, return values, or data relationships.
- **AttemptTrace** ([TRACE]):
  - ExCyTIn-Bench: Historical records of action executions, including successful queries and failure reasons.
  - AppWorld: Historical records of code executions, including successful API calls and failure reasons.

B.1.2 Outcome Proxy Seven-Way Classification

SEER predicts possible feedback from the environment by discretizing the outcome space into seven mutually exclusive categories as follows.

- ExCyTIn-Bench:
  - SUCCESS: Query succeeds and returns data.
  - EMPTY_RESULT: Query succeeds but returns empty.
  - SYNTAX_ERROR: SQL syntax error.
  - SCHEMA_ERROR: Table or column does not exist.
  - PERMISSION_ERROR: Insufficient permissions.
  - TIMEOUT/LIMIT: Timeout or resource limit.
  - LOGIC_ERROR: Logical error.
- AppWorld:
  - SUCCESS: Code executes successfully with output.
  - EMPTY_RESULT: Code succeeds but returns empty/None.
  - PARAMETER_ERROR: Invalid or missing parameters.
  - RESOURCE_ERROR: Resource not found or unavailable.
  - AUTH_ERROR: Authentication or authorization failure.
  - SCHEMA_ERROR: API or method does not exist.
  - EXECUTION_ERROR: Runtime execution error.

B.2 Prompt Templates

SEER uses structured prompts for evidence extraction, action-conditioned outcome prediction, evidence curation, and final action generation. Such prompt dependence is not unique to SEER: frozen-LLM agent pipelines that rely on intermediate reasoning traces or structured outputs generally face similar sensitivity to prompt phrasing, format adherence, and parsing reliability. At the same time, the prompts used in SEER are structured but not heavily hand-crafted. In our implementation, they are generated automatically from a simple meta-prompt, without manual prompt search or repeated hand tuning.

Our prompt-robustness analysis in Appendix E.6 further shows that **SEER is not dependent on a single carefully tuned prompt template**. Specifically, we generate independent prompt variants for the main structured-output stages and use them directly without manual filtering or editing. The results remain stable across variants, suggesting that the performance of SEER is primarily driven by the framework design rather than prompt-specific tuning. Below, we provide the default prompt templates used in the main experiments.

The prompts are organized into four categories:

- **Evidence Extraction:** Extracting structured evidence from the latest interaction.
- **Action Generation & Outcome Prediction:** Generating the hypothetical next-step action and predicting its outcome distribution, which is used for entropy computation.
- **Evidence Curation:** Refining the evidence pool through de-duplication, merging, and condensation.
- **Agent Action Generation:** Generating the next-step action conditioned on the constructed context.

Next, using ExCyTIn-Bench as a running example, we present the corresponding prompt templates.

Evidence Extraction. This prompt template is used after each SQL execution, using the latest interaction as input. The output should be a JSON array where each element has the structure *{"type": ..., "content": ...}*.

Prompt Template: Evidence Extraction (ExCyTIn-Bench)

```
You are a cybersecurity evidence extractor for security incident
investigation.  Your goal is to extract STRICTLY FACTUAL evidence
from SQL query results.

## CRITICAL RULE: NO HALLUCINATION
- ONLY extract information that is EXPLICITLY present in the
Observation
- DO NOT guess table names, column names, or data values
- If observation is empty/whitespace, report it as empty result
- If observation shows an error, extract the error EXACTLY as written
- For errors:  use TRACE type with format "ACTION failed:  ERROR"

## Output Quantity Limit
- Extract AT MOST 5 evidence items total (0 to 5 items)
- Prioritize quality over quantity - only extract the most important
facts
- If observation contains no useful information, output NOTHING (empty
response)

## Input
Task:  {task}
Action:  {action}
Observation:  {observation}

## Output Format - CRITICAL: MUST FOLLOW EXACTLY

Your output will be parsed as JSON. You MUST output valid JSON format.

**Rules:**
- Output a JSON array of evidence objects
- Each object has:  "type" (CONSTRAINT/FACT/TRACE) and "content"
(string)
- AT MOST 5 items total - choose only the most important ones
- If no useful information, output an empty array:  []

**Format Pattern:**
[ {"type":  "CONSTRAINT", "content":  "exact description from
observation"}, {"type":  "FACT", "content":  "exact fact from
observation"}, {"type":  "TRACE", "content":  "exact action and
result"} ]
**How Your Output Will Be Parsed:**
- Output must be valid JSON array
- "type" must be exactly:  "CONSTRAINT", "FACT", or "TRACE"
- "content" can be any string (including newlines, special characters)
- Invalid JSON will be rejected and NO evidence will be extracted

## Examples

Example 1 - Empty result:
Action:  SELECT AccountName FROM DeviceProcessEvents WHERE
ProcessCommandLine LIKE '%malware%'
Observation:  "Query executed successfully.  0 rows returned."
Output:  ["type":  "TRACE", "content":  "Executed query on
DeviceProcessEvents but returned 0 rows"]

Example 2 - Schema error:
Action:  SELECT * FROM DeviceEvents
Observation:  "Error:  Table 'DeviceEvents' doesn't exist"
Output:  ["type":  "TRACE", "content":  "SELECT * FROM DeviceEvents
failed:  Table 'DeviceEvents' doesn't exist"]
```

```
Example 3 - Successful query:
Observation:  "Column names:  DeviceName, AccountName, TimeGenerated.
2 rows found."
Output:  [ "type":  "FACT", "content":  "Table has columns:
DeviceName, AccountName, TimeGenerated", "type":  "FACT", "content":
"Found 2 rows in the result" ]

Example 4 - No useful information:
Action:  SELECT 1
Observation:  "1"
Output:  []

Remember:  Extract AT MOST 5 items.  Output valid JSON array only.

Extract now:
```

Action Generation and Outcome Prediction. At each step, this prompt template is used to generate the hypothetical next-step action and predict its outcome distribution. The output should be a JSON object containing '*action*' (SQL string) and '*probabilities*' (array of length 7).

Action Generation and Outcome Prediction (ExCyTIn-Bench)

```
You are an expert database analyst.  Your task is to determine the
next SQL action AND predict its execution outcome.

## Task
{task}

## Last Action
{last_action}
## Context
{context_block}
## Instructions
Think through this systematically as an expert:
1.  **Consider ALL possibilities**:  Even successful queries can fail
unexpectedly
2.  **Question assumptions**:  Tables you assume exist may not;
permissions you have may be limited
3.  **Be honest about uncertainty**:  If you're unsure, distribute
probability across multiple categories

## Outcome Categories
0:  SUCCESS - Query executes successfully and returns data
1:  EMPTY_RESULT - Query executes but returns 0 rows (syntax valid, no
matching data)
2:  SYNTAX_ERROR - SQL syntax is malformed (missing quotes, unmatched
parentheses, invalid keywords)
3:  SCHEMA_ERROR - Table or column doesn't exist in database
4:  PERMISSION_ERROR - Insufficient privileges to access the data
5:  TIMEOUT - Query takes too long or hits resource limits
6:  LOGIC_ERROR - Query has logical contradictions or invalid
operations

## UNCERTAINTY GUIDELINES
Database exploration is HIGHLY uncertain:
- You don't know the actual schema
- You don't know your permissions
- You don't know the data patterns

**CRITICAL**:  You must think twice before outputting probabilities:
1.  **First**:  Decide what action to take
```

```
2. **Second**: Based on YOUR specific action, predict what outcome
is most likely

For example:
- If you query a table you've never seen before -> SCHEMA_ERROR should
be HIGH
- If you query without knowing permissions -> PERMISSION_ERROR should
be CONSIDERED
- If your query has complex joins -> SYNTAX_ERROR might be possible
- If you've seen this table work before -> SUCCESS might be higher

**Probability Assignment Rules**:
- If completely unsure about THIS specific action: use uniform
distribution [0.14, 0.14, 0.14, 0.14, 0.14, 0.14, 0.14]
- If somewhat confident: give higher probability to 2-3 likely
outcomes, keep others > 0.05
- NEVER put all probability on a single outcome unless you have
overwhelming evidence

## Task
Output your response as a JSON object with TWO fields:
- "action": The SQL query you would execute next (ONLY the SQL, no
explanation)
- "probabilities": An array of 7 numbers (0-1) that sum to 1.0

The 7 probabilities correspond to: [SUCCESS, EMPTY_RESULT,
SYNTAX_ERROR, SCHEMA_ERROR, PERMISSION_ERROR, TIMEOUT, LOGIC_ERROR]
**Think step-by-step**:
1. What action will I take?
2. For THIS action, what are all possible outcomes?
3. How likely is each outcome? (be honest about uncertainty)

Example format:
"action": "SELECT * FROM Users WHERE AccountName='admin'",
"probabilities": [0.6, 0.2, 0.05, 0.1, 0.03, 0.01, 0.01]

Another example (high uncertainty - never seen this table):
"action": "SHOW TABLES", "probabilities": [0.3, 0.3, 0.1, 0.15,
0.05, 0.05, 0.05]

Your response (JSON only, no extra text):
```

Evidence Curation. As soon as 10 new evidence units are added to the evidence pool, we apply this prompt template to curate the pool.

Evidence Curation (ExCyTIn-Bench)

```
You are an Evidence Curator for cybersecurity investigation. Your
goal is to refine and consolidate a collection of evidence items to
make them more useful for AI agents.
## Your Task

You will receive a list of evidence items gathered during SQL database
investigation. Each item is tagged as:
- **[CONSTRAINT]**: Schema limitations, missing tables/columns,
permissions
- **[FACT]**: Confirmed data values, relationships, counts
- **[TRACE]**: Actions taken and their results (especially failures)

## Curator Responsibilities

### 1. Identify Redundancies
```

```
Evidence items are redundant if they:
- Convey the same factual information (even if worded differently)
- Describe the same schema constraint
- Report overlapping action results

### 2. Consolidate Similar Evidence Merge evidence that:
- Refers to the same table/column/entity
- Describes related aspects of the same discovery
- Can be combined into a more comprehensive statement

### 3. Prioritize Critical Information Preserve:
- **Schema facts** (table/column existence) - highest priority
- **Security-relevant facts** (malicious IPs, suspicious accounts)
- **Failed attempts** that reveal what doesn't work
- **Constraints** that prevent wasted exploration

Remove or simplify:
- Repetitive trace logs
- Overly verbose descriptions
- Low-value observational details

### 4. Improve Clarity
Rewrite evidence to be:
- **Concise**: Use minimal words to convey maximum information
- **Precise**: Use exact table/column names from observations
- **Actionable**: Help the AI avoid repeating mistakes

## Input Evidence

{evidence_list}
## Output Format

Return ONLY the refined evidence list, following this EXACT format:

[CONSTRAINT] <refined constraint description> [FACT] <refined fact
description> [TRACE] <refined trace description>
**Rules:**
- Each evidence on a separate line
- Start with tag: [CONSTRAINT], [FACT], or [TRACE]
- No explanations, no numbering, no markdown
- Lines not matching this pattern will be IGNORED

## Examples

**Input:**
[FACT] Table DeviceEvents has 5 columns
[FACT] Found that DeviceEvents contains columns: EventID, DeviceName,
AccountName, TimeGenerated
[TRACE] Tried SELECT * FROM DeviceEvents but got empty result

**Output:**
[FACT] Table DeviceEvents has columns: EventID, DeviceName,
AccountName, TimeGenerated
[TRACE] SELECT * FROM DeviceEvents returned 0 rows

**Input:**
[CONSTRAINT] Table Users doesn't exist [CONSTRAINT] Error: table
'Users' not found in database [FACT] Found table called Users_v2
instead
**Output:**
[CONSTRAINT] Table 'Users' doesn't exist, but 'Users_v2' exists

Curate now:
```

Agent Action Generation. At each step, we use this prompt template to generate the next-step action conditioned on the constructed context.

Agent Action Generation (ExCyTIn-Bench)

```
You are a security analyst.  You need to interact with a database
with MYSQL queries to collect info and then answer a given security
question.  The logs are stored in a MySQL database, you can use SQL
queries to retrieve entries as needed.  Note there are more than 20
tables in the database, so you may need to explore the schema or check
example entries to understand the database structure.

{evidence_section}

[LAST_ACTION]
{last_action_section}

[LATEST_OBSERVATION]
{observation}

In your response, you should give your thoughts and actions.  The
reasoning process and action are enclosed within <answer> </answer>
tags.
You should ONLY have ONE action per response in the <answer> </answer>
block, it can be one of the following based on your reasoning:
(1) <answer>execute[<your sql query>]</answer>, which executes the SQL
query.  For example, execute[SHOW TABLES]. You should give sql queries
to explore the schema and acquire information.
(2) <answer>submit[<your answer>]</answer>, which submits the final
answer to the question.  When you believe you have enough information
to answer the question, you can submit your answer.

Please do not do excessive reasoning.  Briefly reason about the
current situation and then give your action quickly.  DO NOT make
assumptions about the data that are not observed in the logs.  Be
concise and precise in your response.
```

C Theoretical Details

This appendix provides the complete theoretical analysis underlying the concise characterization in Section 3.4. We first formalize the action-conditioned oracle objective and its connection to thresholded evidence selection. We then analyze simultaneous admission under controlled evidence redundancy, quantify the effect of LLM-based proxy-uncertainty estimation, and derive additional consequences for proxy confidence and action-level behavior. We also provide empirical diagnostics for the conditions used in the analysis.

C.1 Setup and Action-Conditioned Oracle Objective

Step-t view. Fix a step t and the latest interaction $o_t = (a_{t-1}, x_t)$. Within this step, the selected evidence pool S determines the context $C_t = g_t(S)$ and consequently the action proposed under that context. Let $a_t(S)$ denote the action induced by $g_t(S)$. The proxy outcome $\widetilde{Y}_{t+1} \in \mathcal{Y}$ is realized after executing this action, so its distribution depends on both the selected context and the induced action.

Action-conditioned proxy uncertainty. We define

$$U_t^{\mathrm{act}}(S) \triangleq H\Big(\widetilde{Y}_{t+1} \mid g_t(S), a_t(S)\Big), \qquad f_t^{\mathrm{act}}(S) \triangleq -U_t^{\mathrm{act}}(S). \tag{14}$$

Here, $f_t^{\mathrm{act}}(S)$ increases as the uncertainty over the subsequent proxy outcome decreases. During practical scoring, SEER does not execute $a_t(S)$ in the environment. Instead, Section 3.3 uses the hypothetical action $\hat{a}_t(S)$ and the predicted proxy-outcome distribution to estimate this quantity.

For an evidence unit e, define its oracle marginal uncertainty reduction as

$$\Delta_t^{\mathrm{act}}(e \mid S) \triangleq U_t^{\mathrm{act}}(S) - U_t^{\mathrm{act}}(S \cup \{e\}) = f_t^{\mathrm{act}}(S \cup \{e\}) - f_t^{\mathrm{act}}(S), \tag{15}$$

and its gain-per-cost score as

$$\mathrm{score}_t^{\mathrm{act}}(e \mid S) \triangleq \frac{\Delta_t^{\mathrm{act}}(e \mid S)}{\mathrm{cost}(e)}. \tag{16}$$

C.2 Thresholding as Cost-Regularized Evidence Selection

To connect the local admission rule with the relevance-cost trade-off in Section 3.1, define

$$F_{t,\mathrm{act}}^{\tau}(S) \triangleq f_t^{\mathrm{act}}(S) - \tau\,\mathrm{cost}(S), \qquad \mathrm{cost}(S) \triangleq \sum_{e \in S} \mathrm{cost}(e). \tag{17}$$

For any $e \notin S$,

$$F_{t,\mathrm{act}}^{\tau}(S \cup \{e\}) - F_{t,\mathrm{act}}^{\tau}(S) = \Delta_t^{\mathrm{act}}(e \mid S) - \tau\,\mathrm{cost}(e). \tag{18}$$

Therefore, thresholding $\mathrm{score}_t^{\mathrm{act}}(e \mid S) \geq \tau$ is exactly equivalent to requiring a nonnegative marginal improvement in the oracle cost-regularized objective.

Proposition C.1 (Per-item improvement under thresholding)**.** *Fix a step t and a baseline pool S. If an evidence unit e satisfies $\mathrm{score}_t^{\mathrm{act}}(e \mid S) \geq \tau$, then*

$$\Delta_t^{\mathrm{act}}(e \mid S) \geq \tau\,\mathrm{cost}(e), \tag{19}$$

and

$$F_{t,\mathrm{act}}^{\tau}(S \cup \{e\}) \geq F_{t,\mathrm{act}}^{\tau}(S). \tag{20}$$

Proof. By definition, $\mathrm{score}_t^{\mathrm{act}}(e \mid S) = \Delta_t^{\mathrm{act}}(e \mid S)/\mathrm{cost}(e)$. Thus, $\mathrm{score}_t^{\mathrm{act}}(e \mid S) \geq \tau$ implies $\Delta_t^{\mathrm{act}}(e \mid S) \geq \tau\,\mathrm{cost}(e)$. Substituting this inequality into Eq. (18) proves the second statement. □

C.3 Simultaneous Admission under Controlled Redundancy

Why simultaneous admission requires additional structure. SEER scores all candidates in $\mathcal{N}_t$ against the same baseline S_{t-1} and admits qualified candidates simultaneously. This design avoids order effects within a step. However, the joint gain of several admitted units need not equal the sum of their individual gains because different units may carry overlapping information. We characterize this interaction through a step-dependent redundancy factor.

Assumption C.2 (Controlled redundancy)**.** Fix a step t. There exists $\gamma_t \in (0, 1]$ such that for any baseline pool S and any finite candidate set E,

$$f_t^{\mathrm{act}}(S \cup E) - f_t^{\mathrm{act}}(S) \geq \gamma_t \sum_{e \in E} \Delta_t^{\mathrm{act}}(e \mid S). \tag{21}$$

The factor γ_t quantifies how much of the sum of individual baseline gains remains after admitting the candidates jointly. When $\gamma_t = 1$, their joint gain matches or exceeds the summed baseline gains. Smaller positive values allow overlapping information to reduce the realized joint improvement. This condition does not require global submodularity of the uncertainty objective.

Empirical diagnostic. We evaluate the realized redundancy ratio

$$\frac{f_t^{\mathrm{act}}(S \cup E) - f_t^{\mathrm{act}}(S)}{\sum_{e \in E} \Delta_t^{\mathrm{act}}(e \mid S)} \tag{22}$$

on 214 randomly sampled ExCyTIn-Bench steps. As shown in Figure 3a, the ratio lies in $[0.08, 1]$ for 98% of the measured cases. Singleton admissions have ratio 1 by construction. This diagnostic suggests that the redundancy factor remains positive and non-vanishing for most evaluated steps.

Theorem C.3 (One-step improvement under simultaneous oracle admission)**.** *Fix a step t and define*

$$E_t^{\mathrm{act}} = \left\{ e \in \mathcal{N}_t : \mathrm{score}_t^{\mathrm{act}}(e \mid S_{t-1}) \geq \tau \right\}, \qquad S_t = S_{t-1} \cup E_t^{\mathrm{act}}. \tag{23}$$

Under Assumption C.2,

$$f_t^{\mathrm{act}}(S_t) - f_t^{\mathrm{act}}(S_{t-1}) \geq \gamma_t \tau \,\mathrm{cost}(E_t^{\mathrm{act}}), \tag{24}$$

or equivalently,

$$U_t^{\mathrm{act}}(S_t) \leq U_t^{\mathrm{act}}(S_{t-1}) - \gamma_t \tau \,\mathrm{cost}(E_t^{\mathrm{act}}). \tag{25}$$

Moreover,

$$F_{t,\mathrm{act}}^{\tau}(S_t) - F_{t,\mathrm{act}}^{\tau}(S_{t-1}) \geq (\gamma_t - 1)\tau \,\mathrm{cost}(E_t^{\mathrm{act}}). \tag{26}$$

Proof. For every $e \in E_t^{\mathrm{act}}$, Proposition C.1 gives

$$\Delta_t^{\mathrm{act}}(e \mid S_{t-1}) \geq \tau \,\mathrm{cost}(e). \tag{27}$$

Summing over the admitted set gives

$$\sum_{e \in E_t^{\mathrm{act}}} \Delta_t^{\mathrm{act}}(e \mid S_{t-1}) \geq \tau \,\mathrm{cost}(E_t^{\mathrm{act}}). \tag{28}$$

Applying Assumption C.2 yields

$$\begin{aligned} f_t^{\mathrm{act}}(S_t) - f_t^{\mathrm{act}}(S_{t-1}) &\geq \gamma_t \sum_{e \in E_t^{\mathrm{act}}} \Delta_t^{\mathrm{act}}(e \mid S_{t-1}) \\ &\geq \gamma_t \tau \,\mathrm{cost}(E_t^{\mathrm{act}}), \end{aligned}$$

which proves Eq. (24). Since $f_t^{\mathrm{act}} = -U_t^{\mathrm{act}}$, the uncertainty bound follows immediately. Finally,

$$\begin{aligned} F_{t,\mathrm{act}}^{\tau}(S_t) - F_{t,\mathrm{act}}^{\tau}(S_{t-1}) &= f_t^{\mathrm{act}}(S_t) - f_t^{\mathrm{act}}(S_{t-1}) - \tau \,\mathrm{cost}(E_t^{\mathrm{act}}) \\ &\geq (\gamma_t - 1)\tau \,\mathrm{cost}(E_t^{\mathrm{act}}), \end{aligned}$$

which proves Eq. (26). □

Proposition C.4 (Local τ-stopping property)**.** *Fix a step t and baseline pool S_{t-1}. If no candidate is admitted, then for every $e \in \mathcal{N}_t$,*

$$\Delta_t^{\mathrm{act}}(e \mid S_{t-1}) < \tau \,\mathrm{cost}(e), \tag{29}$$

and

$$F_{t,\mathrm{act}}^{\tau}(S_{t-1} \cup \{e\}) < F_{t,\mathrm{act}}^{\tau}(S_{t-1}). \tag{30}$$

Proof. If no candidate passes the threshold, then $\mathrm{score}_t^{\mathrm{act}}(e \mid S_{t-1}) < \tau$ for every $e \in \mathcal{N}_t$. The first result follows directly from the score definition, and the second follows from Eq. (18). □

C.4 Robustness to Proxy-Uncertainty Estimation

Section 3.3 evaluates only the baseline pool and the single-addition pools corresponding to newly extracted evidence. Therefore, the analysis only requires the estimator to approximate the oracle uncertainty on this restricted family, rather than on all possible evidence subsets.

Define

$$\mathcal{S}_t \triangleq \{S_{t-1}\} \cup \{S_{t-1} \cup \{e\} : e \in \mathcal{N}_t\}. \tag{31}$$

Assumption C.5 (Restricted entropy estimation error)**.** Fix a step t. There exists $\varepsilon_H \geq 0$ such that

$$\max_{S \in \mathcal{S}_t} \left| \widehat{H}_t(S) - U_t^{\mathrm{act}}(S) \right| \leq \varepsilon_H. \tag{32}$$

Empirical diagnostic. We compare $\widehat{H}_t(S)$ with a post-hoc estimate of $U_t^{\mathrm{act}}(S)$ on the candidate pools evaluated at 237 randomly sampled ExCyTIn-Bench steps. Figure 3b shows that the absolute error remains below 1.75 in all measured cases, with a mean error of 0.9185.

Proposition C.6 (Marginal-gain error)**.** *Under Assumption C.5, for every $e \in \mathcal{N}_t$,*

$$\left|\widehat{\Delta}_t(e \mid S_{t-1}) - \Delta_t^{\mathrm{act}}(e \mid S_{t-1})\right| \le 2\varepsilon_H. \tag{33}$$

Proof. By the definitions of the estimated and oracle marginal gains,

$$\begin{aligned}&\widehat{\Delta}_t(e \mid S_{t-1}) - \Delta_t^{\mathrm{act}}(e \mid S_{t-1})\\ =\ &\left[\widehat{H}_t(S_{t-1}) - U_t^{\mathrm{act}}(S_{t-1})\right]\\ &- \left[\widehat{H}_t(S_{t-1}\cup\{e\}) - U_t^{\mathrm{act}}(S_{t-1}\cup\{e\})\right].\end{aligned}$$

The triangle inequality and Assumption C.5 bound each term by ε_H, giving Eq. (33). □

Corollary C.7 (Near-threshold oracle improvement)**.** *Under Assumption C.5, if SEER admits $e \in \mathcal{N}_t$ because $\widehat{\mathrm{score}}_t(e) \ge \tau$, then*

$$\Delta_t^{\mathrm{act}}(e \mid S_{t-1}) \ge \tau\,\mathrm{cost}(e) - 2\varepsilon_H, \tag{34}$$

and

$$F^{\tau}_{t,\mathrm{act}}(S_{t-1}\cup\{e\}) - F^{\tau}_{t,\mathrm{act}}(S_{t-1}) \ge -2\varepsilon_H. \tag{35}$$

Proof. The admission rule gives $\widehat{\Delta}_t(e \mid S_{t-1}) \ge \tau\,\mathrm{cost}(e)$. Proposition C.6 then implies

$$\Delta_t^{\mathrm{act}}(e \mid S_{t-1}) \ge \widehat{\Delta}_t(e \mid S_{t-1}) - 2\varepsilon_H \ge \tau\,\mathrm{cost}(e) - 2\varepsilon_H. \tag{36}$$

Substituting this bound into Eq. (18) proves the second result. □

Corollary C.8 (Simultaneous admission with estimated scores)**.** *Fix a step t and let*

$$\widehat{E}_t = \left\{e \in \mathcal{N}_t : \widehat{\mathrm{score}}_t(e) \ge \tau\right\}, \qquad S_t = S_{t-1}\cup\widehat{E}_t. \tag{37}$$

Under Assumptions C.2 and C.5,

$$f_t^{\mathrm{act}}(S_t) - f_t^{\mathrm{act}}(S_{t-1}) \ge \gamma_t\left(\tau\,\mathrm{cost}(\widehat{E}_t) - 2|\widehat{E}_t|\varepsilon_H\right), \tag{38}$$

and

$$F^{\tau}_{t,\mathrm{act}}(S_t) - F^{\tau}_{t,\mathrm{act}}(S_{t-1}) \ge (\gamma_t - 1)\tau\,\mathrm{cost}(\widehat{E}_t) - 2\gamma_t|\widehat{E}_t|\varepsilon_H. \tag{39}$$

Proof. For every $e \in \widehat{E}_t$, Corollary C.7 gives

$$\Delta_t^{\mathrm{act}}(e \mid S_{t-1}) \ge \tau\,\mathrm{cost}(e) - 2\varepsilon_H. \tag{40}$$

Summing over $\widehat{E}_t$ yields

$$\sum_{e\in\widehat{E}_t} \Delta_t^{\mathrm{act}}(e \mid S_{t-1}) \ge \tau\,\mathrm{cost}(\widehat{E}_t) - 2|\widehat{E}_t|\varepsilon_H. \tag{41}$$

Applying Assumption C.2 proves Eq. (38). Subtracting $\tau\,\mathrm{cost}(\widehat{E}_t)$ then gives Eq. (39). □

C.5 From Proxy Uncertainty to Confidence

The previous results characterize how evidence selection changes proxy uncertainty. We next give a direct distributional interpretation of this quantity.

Let $p_t(\cdot \mid S, o_t)$ denote the oracle conditional distribution of $\widetilde{Y}_{t+1}$ under $(g_t(S), a_t(S))$, so that

$$U_t^{\mathrm{act}}(S) = H(p_t(\cdot \mid S, o_t)). \tag{42}$$

Proposition C.9 (Entropy-to-confidence)**.** *For any pool S,*

$$\max_{y\in\mathcal{Y}} p_t(y \mid S, o_t) \ge \exp\left(-U_t^{\mathrm{act}}(S)\right), \tag{43}$$

where the entropy uses natural logarithms.

Proof. Let $p_{\max} = \max_y p_t(y \mid S, o_t)$. Since $p_t(y \mid S, o_t) \le p_{\max}$ for every y,

$$-\log p_t(y \mid S, o_t) \ge -\log p_{\max}. \tag{44}$$

Therefore,

$$\begin{aligned} U_t^{\mathrm{act}}(S) &= -\sum_y p_t(y \mid S, o_t) \log p_t(y \mid S, o_t) \\ &\ge -\sum_y p_t(y \mid S, o_t) \log p_{\max} \\ &= -\log p_{\max}. \end{aligned}$$

Rearranging proves Eq. (43). □

This result gives a proxy-level interpretation of entropy reduction: as $U_t^{\mathrm{act}}(S)$ decreases, the lower bound on the probability assigned to the most likely functional outcome increases. For proxy types that represent coarse execution states, the selected context therefore supports a more concentrated prediction over subsequent environment feedback.

C.6 Optional Action-Level Interpretation

The previous results concern uncertainty over proxy outcomes and do not require any assumption about the agent's action distribution. To connect proxy predictability with action-level decisiveness, we consider the following additional condition.

Let $A_t(S)$ denote the random action generated by the agent under context $g_t(S)$ when sampling from its action distribution.

Assumption C.10 (Proxy-to-action decisiveness)**.** Fix a step t. There exists a non-decreasing function $\eta_t : \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$ such that for any pool S,

$$H(A_t(S) \mid g_t(S), o_t) \le \eta_t\big(U_t^{\mathrm{act}}(S)\big). \tag{45}$$

This condition states that lower uncertainty over the subsequent proxy outcome corresponds to a non-increasing upper bound on the entropy of the induced action distribution. It does not require a linear or Lipschitz relationship.

Empirical diagnostic. We estimate action entropy by sampling ten actions with temperature 1.0 for each evaluated state and estimate $U_t^{\mathrm{act}}(S)$ using post-hoc proxy-outcome entropy. We then group the sampled states by proxy uncertainty. Figure 3c shows that lower-uncertainty groups generally exhibit lower average action entropy, providing empirical support for the monotone relationship captured above.

Corollary C.11 (Proxy uncertainty and action decisiveness)**.** *Under Assumption C.10, a decrease in $U_t^{\mathrm{act}}(S)$ yields a non-increasing upper bound on the entropy of the induced action distribution through η_t. In particular, the uncertainty reductions characterized in Theorem C.3 and Corollary C.8 tighten this upper bound.*

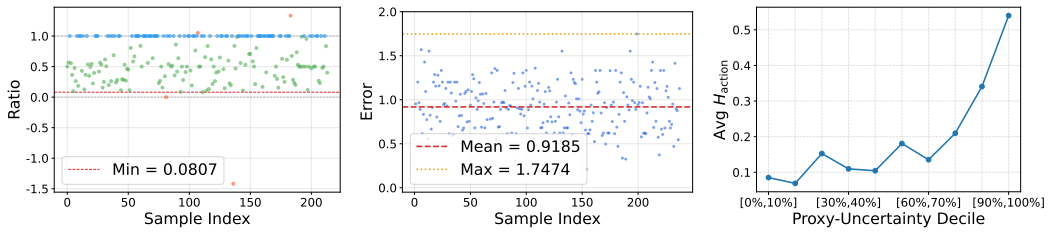
Proof. The result follows directly from Assumption C.10 and the monotonicity of η_t. □

C.7 Empirical Diagnostics for the Theoretical Conditions

Figure 3 summarizes the empirical diagnostics used above. These analyses do not form part of the algorithm; they examine how the conditions used in the theoretical characterization behave on ExCyTIn-Bench.

D Case Study: AppWorld Example

Here we present a representative case study on AppWorld with Qwen3-32B to illustrate how SEER improves interactive tool use in practice. We focus on Task ID 552869a_3 and report the step-by-step interaction logs of SEER and ACE on the same task. While ACE fails to complete this task, SEER succeeds, and the comparison highlights several decision-critical moments where SEER constructs a cleaner, decision-relevant context by extracting and selecting compact evidence, leading to more stable actions and more effective recovery from failures. We attribute this to the following factors.



(a) Redundancy ratio γ_t (b) Restricted entropy error ε_H (c) Proxy uncertainty vs. action entropy

Figure 3: **Empirical diagnostics for the theoretical analysis on ExCyTIn-Bench.** Left: the realized joint uncertainty reduction remains a positive fraction of the summed individual gains for most evaluated steps. Middle: the LLM-based proxy-entropy estimator exhibits bounded error on the candidate pools actually evaluated during scoring. Right: lower proxy uncertainty generally accompanies lower entropy of the sampled action distribution.

- **ACE leveraged prior experience** — it correctly identifies SHOW_TRANSACTIONS as the relevant API from its playbook, but incorrectly assumes that the QUERY parameter performs exact filtering rather than text-search ranking. This misunderstanding causes it to aggregate many irrelevant transactions (4392 vs. the ground truth 165).
- **SEER employed evidence-driven interpretation** — it extracts the API documentation as evidence, recognizes (via entropy-based selection) that QUERY is used for search rather than filtering, and then implements exact string matching in Python to filter the retrieved results.
- **Key distinction** — ACE's playbook is helpful for recalling which API to call, but it can also encode semantic misconceptions about API behavior. In contrast, the IB-driven, per-task evidence selection in SEER encourages accurate interpretation from the current interaction signals, preventing the agent from committing to an incorrect assumption.

Log for Task 552869a_3 (SEER)

```
Task ID: 552869a_3
Instruction:  How much have I paid in internet bill on venmo this year
so far?
Supervisor:
Martin Blackburn
mar_blac@gmail.com
3539110281

agent (step #1)
```python
# Step 1:  Get Venmo account password from Supervisor app
venmo_password = [p["password"] for p in apis.supervisor.

show_account_passwords() if p["account_name"] == "venmo"][0]

# Step 2:  Log in to Venmo to get access token
venmo_login_result = apis.venmo.login(username='mar_blac@gmail.com',
password=venmo_password)
access_token = venmo_login_result["access_token"]

# Step 3:  Get API spec for Venmo transactions
print(apis.api_docs.sho_api_descriptions(app_name='venmo'))
```

environment (step #2)
[{
"name": "show_account",
```

```
"description": "Show your account information. Unlike show_profile,
this includes private information."},
{
"name": "signup",
"description": "Sign up to create account."},
{
"name": "delete_account",
"description": "Delete your account."},

... ...

{
"name": "show_transactions",
"description": "Search or show a list of your transactions."},

... ...

{ "name": "delete_notification",
"description": "Delete a notification."},
{
"name": "mark_notification",
"description": "Mark a notification as read or unread."}]

agent (step #2)
"'python
# Step 4: Get API spec for Venmo transactions
print(apis.api_docs.show_api_doc(app_name='venmo',
api_name='show_transactions'))
"'

environment (step #3)
{
"app_name": "venmo",
"api_name": "show_transactions",
"path": "/venmo/transactions",
"method": "GET",
"description": "Search or show a list of your transactions.",
"parameters": [
{
"name": "access_token",
"type": "string",
"required": true,
"description": "Access token obtained from venmo app login.",
"default": null,
"constraints": []
},
{
"name": "min_created_at",
"type": "string",
"required": false,
"description": "Minimum created_at datetime to filter transactions in
YYYY-MM-DD format.",
"default": "1500-01-01",
"constraints": []
},
{
"name": "max_created_at",
"type": "string",
"required": false,
"description": "Maximum created_at datetime to filter transactions in
YYYY-MM-DD format.",
"default": "3000-01-01",
"constraints": []
```

```
},

...  ...

{
"name":  "page_limit",
"type":  "integer",
"required":  false,
"description":  "The maximum number of results to return per page.",
"default":  5,
"constraints":  [
"value >= 1, <= 20"
]
},
{
"name":  "sort_by",
"type":  "string",
"required":  false,
"description":  "The attribute to sort the transactions by prefixed
with +/- to reflect ascending/descending.  Valid attributes:
created_at, like_count and amount.  If both query and sort_by are
given and non-empty, results will be first ranked by query relevance,
then paginated, and will then be sorted by the given attribute within
each page.",
"default":  null,
"constraints":  []
}
],
"response_schemas":  {
"success":  [
{ "transaction_id":  1,
"amount":  0.0,
"description":  "string",
"created_at":  "2019-01-01T00:00:00",
"updated_at":  "2019-01-01T00:00:00",
"private":  true,
"like_count":  1,
"payment_card_digits":  "string",
"comment_count":  1,
"sender":  {
"name":  "string",
"email":  "user@example.com"
},
"receiver":  {
"name":  "string",
"email":  "user@example.com"
},
"liked":  true
}
],
"failure":  {
"message":  "string"
}
}
}

agent (step #3)
"'python
from datetime import datetime

# Step 5:  Get current date for filtering transactions
today = datetime.now()
current_year = today.year
```

```
min_created_at = f"current_year-01-01"
max_created_at = today.strftime("%Y-%m-%d")

# Step 6:  Fetch all sent transactions for this year
page_index = 0
transactions = []
while True:
transaction_page = apis.venmo.show_transactions(
access_token=access_token,
min_created_at=min_created_at,
max_created_at=max_created_at,
direction='sent',
page_index=page_index,
page_limit=20
)
if not transaction_page:
break
transactions.extend(transaction_page)
page_index += 1

# Step 7:  Filter transactions with "internet" in description and sum
amounts
internet_bill_total = sum(t["amount"] for t in transactions if
"internet" in t["description"].lower())

# Step 8:  Complete task with the calculated total
apis.supervisor.complete_task(answer=internet_bill_total)
"'

environment (step #4)
Execution successful.
```

Log for Task 552869a_3 (ACE)

```
Task ID: 552869a_3
Instruction:  How much have I paid in internet bill on venmo this year
so far?
Supervisor:
Martin Blackburn
mar_blac@gmail.com
3539110281

agent (step #1)
"'python
# Step 1:  Retrieve Venmo credentials from supervisor app
venmo_password = [acc["password"] for acc in apis.supervisor.
show_account_passwords() if acc["account_name"] == "venmo"][0]

# Step 2:  Login to Venmo to get access token
venmo_login_result = apis.venmo.login(username='mar_blac@gmail.com',
password=venmo_password)
access_token = venmo_login_result["access_token"]

# Step 3:  Check show_transactions API specs
print(apis.api_docs.show_api_doc(app_name='venmo',
api_name='show_transactions'))
"'

environment (step #1)
{
"app_name":  "venmo",
"api_name":  "show_transactions",
```

```
"path": "/venmo/transactions",
"method": "GET",
"description": "Search or show a list of your transactions.",
"parameters": [
{
"name": "access_token",
"type": "string",
"required": true,
"description": "Access token obtained from venmo app login.",
"default": null,
"constraints": []
},
{
"name": "query",
"type": "string",
"required": false,
"description": "Search query string.",
"default": "",
"constraints": []
},
... ...

{
"name": "page_index",
"type": "integer",
"required": false,
"description": "The index of the page to return.",
"default": 0,
"constraints": [
"value >= 0"
]
},
{
"name": "page_limit",
"type": "integer",
"required": false,
"description": "The maximum number of results to return per page.",
"default": 5,
"constraints": [
"value >= 1, <= 20"
]
}
],
"response_schemas": {
"success": [
{
"transaction_id": 1,
"amount": 0.0,
"description": "string",
"created_at": "2019-01-01T00:00:00",
"updated_at": "2019-01-01T00:00:00",
"private": true,
"like_count": 1,
"payment_card_digits": "string",
"comment_count": 1,
"sender": {
"name": "string",
"email": "user@example.com"
},
"receiver": {
"name": "string",
"email": "user@example.com"
},
```

```
"liked": true
}
],
"failure": {
"message": "string"
}
}
}

agent (step #2)
from datetime import datetime

# Step 4: Fetch all relevant transactions (sent, this year, with
"internet" or "bill" in description)
start_of_year = datetime.now().replace(month=1,
day=1).strftime('%Y-%m-%d')
total_paid = 0.0
page_index = 0

while True:
transactions = apis.venmo.show_transactions(
access_token=access_token,
min_created_at=start_of_year,
direction='sent',
query='internet bill',
page_index=page_index,
page_limit=20
)

if not transactions:
break

for transaction in transactions:
total_paid += transaction["amount"]

page_index += 1

# Step 5: Complete task with the total amount paid
apis.supervisor.complete_task(answer=total_paid)

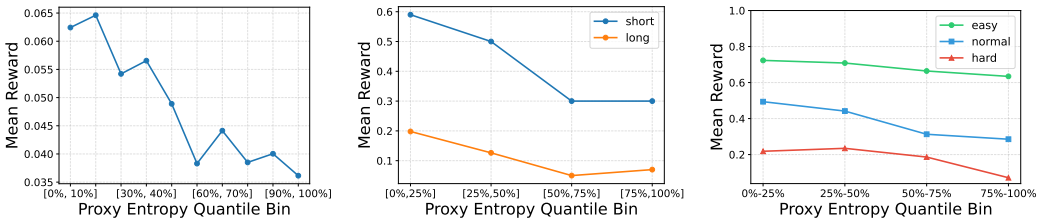
environment (step #2)
Execution successful, but the answer 4392 does not match the ground
truth 165.
```

E More Experiments

E.1 Proxy Entropy Is Associated with Downstream Task Outcomes

A central premise of SEER is that the estimated proxy entropy used for context construction is empirically aligned with downstream task quality. We emphasize that this analysis concerns *context construction* rather than direct action selection: SEER uses proxy entropy to decide which evidence should be included in the context, instead of forcing the agent to choose actions with the most predictable immediate feedback. Thus, the relevant question is whether contexts with lower estimated proxy uncertainty are associated with better downstream outcomes in actual trajectories. We study this question on ExCyTIn-Bench through step-level, episode-level, horizon-stratified, and difficulty-stratified diagnostics. These analyses are intended as association evidence rather than causal identification, since both entropy and reward may be affected by task difficulty, episode length, and the state reached by the agent.

Step-level association. For each step under SEER, we record the estimated proxy entropy $\widehat{H}_t$ after evidence admission. Since ExCyTIn-Bench provides reward at the episode level, we define a



(a) Step-level (b) By horizon (c) By difficulty

Figure 4: **Proxy entropy is associated with downstream task outcomes.** On ExCyTIn-Bench, we analyze the relationship between the estimated proxy entropy optimized by SEER and task reward. (a) At the step level, we sort steps by the post-admission proxy entropy $\widehat{H}_t$ and group them into ten empirical entropy-quantile bins; lower-entropy bins exhibit higher mean step-level reward. (b) At the episode level, we compute $\bar{H}^{\mathrm{ep}} = \frac{1}{T}\sum_{t=1}^{T} \widehat{H}_t$ and stratify episodes by horizon; both short- and long-horizon groups show a negative entropy–reward trend. (c) We further stratify episodes by external task difficulty; the same overall trend remains visible within easy, normal, and hard groups. These results suggest that lower proxy entropy is consistently aligned with better downstream outcomes, while serving as association evidence rather than causal identification.

step-level average reward by evenly distributing the final episode reward across its steps. Specifically, for a step belonging to an episode of length T with final reward RWD, we assign the step-level average reward as RWD/T. We then sort all steps by $\widehat{H}_t$ and partition them into ten empirical entropy-quantile bins, corresponding to the ranges 0–10%, 10–20%, …, 90–100%. For each bin, we compute the mean step-level average reward. As shown in Figure 4a, lower-entropy bins achieve higher average reward, suggesting that lower estimated proxy uncertainty under the constructed context is associated with better downstream task quality.

Episode-level association under different horizons. We next examine the relationship at the episode level. For an episode of length T, we compute its mean proxy entropy as $\bar{H}^{\mathrm{ep}} = \frac{1}{T}\sum_{t=1}^{T} \widehat{H}_t$. We sort episodes by $\bar{H}^{\mathrm{ep}}$ and partition them into four empirical entropy-quantile bins. For each bin, we report the mean final reward RWD. To examine whether the trend is merely an artifact of episode length, we further split episodes into short- and long-horizon groups and perform the same binning within each group. As shown in Figure 4b, both short- and long-horizon episodes exhibit a negative entropy-reward trend: episodes with lower mean proxy entropy tend to obtain higher final reward. The short-horizon curve lies above the long-horizon curve, which is expected because shorter episodes often correspond to easier tasks. Nevertheless, the downward trend remains visible within both groups, indicating that the association is not merely driven by episode length.

Difficulty-stratified association. We further examine whether the entropy-reward association is explained solely by intrinsic task difficulty. To obtain an external difficulty label, we use publicly available trajectories from three reference models with ordered capability GPT-5 $>$ o4-mini $>$ GPT-4.1-nano. A task is labeled *easy* if it is solved by all three models, *normal* if it is solved by two models, *hard* if it is solved by one model, and *challenging* if it is solved by none. Since the challenging subset yields very sparse successful outcomes for small open-source models, we visualize the easy, normal, and hard groups. Within each difficulty group, we compute $\bar{H}^{\mathrm{ep}}$ for every episode, partition episodes into four empirical entropy-quantile bins, and report the mean final reward in each bin. As shown in Figure 4c, the same overall negative entropy-reward trend remains visible within each difficulty group. This suggests that the association between lower proxy entropy and better downstream outcomes is not explained solely by task difficulty.

Overall, these diagnostics support the use of proxy entropy as a meaningful test-time signal for evidence selection. While they do not establish that entropy reduction causally improves reward, they show that the quantity optimized by SEER is consistently aligned with downstream task outcomes across step-level, episode-level, horizon-stratified, and difficulty-stratified views.

Table 5: **Performance of SEER and baselines on additional benchmarks with Qwen3-8B and Qwen3-32B.** For SWE-Bench, we evaluate on the Verified subset, whose instances are manually validated and therefore provide a reliable testbed for agentic software-engineering tasks. For WebArena, we evaluate on the shopping and gitlab subsets due to evaluation cost and deployment constraints; these two subsets together account for 48.8% of the full WebArena benchmark. The bold font indicates the best result and an underline indicates the second-best result. All results are scaled to the range $[0, 100]$.

Methods	Qwen3-8B			Qwen3-32B		
	SWE-Bench	WebArena		SWE-Bench	WebArena	
	Verified	shopping	gitlab	Verified	shopping	gitlab
ExpeL	9.8	<u>10.4</u>	<u>5.4</u>	11.0	<u>15.1</u>	<u>6.8</u>
ACE	<u>10.8</u>	7.3	4.4	<u>12.8</u>	13.9	6.7
SEER	**11.4**	**13.9**	**6.9**	**13.6**	**19.8**	**8.3**

Table 6: **Performance of SEER and baselines with additional model families.** We evaluate Llama-3.1-8B-Instruct and GPT-4.1-nano on ExCyTIn-Bench and AppWorld-N. For AppWorld, we use the normal split because it is more tractable for smaller models, whereas the challenge split is largely unsolved by these models and therefore provides limited comparative signal. The bold font indicates the best result and an underline indicates the second-best result. All results are scaled to the range $[0, 100]$.

Model	Methods	ExCyTIn-Bench		AppWorld-N	
		ACC	RWD	TGC	SGC
Llama-3.1-8B-Instruct	ExpeL	7.8	8.8	<u>3.3</u>	<u>1.8</u>
	ACE	<u>8.0</u>	<u>9.7</u>	2.4	0.0
	SEER	**13.8**	**14.3**	**4.8**	**3.6**
GPT-4.1-nano	ExpeL	12.6	13.1	<u>4.2</u>	0.0
	ACE	<u>14.2</u>	<u>17.5</u>	3.0	<u>1.8</u>
	SEER	**16.2**	**18.1**	**5.6**	**3.6**

E.2 More Benchmarks and Models

To further examine the generality of SEER, we conduct additional evaluations along two complementary axes. First, we extend the benchmark coverage beyond ExCyTIn-Bench and AppWorld by evaluating on SWE-Bench (Yang et al., 2025b) and WebArena (Zhou et al., 2023). Second, we evaluate SEER with additional model families, including Llama-3.1-8B-Instruct and GPT-4.1-nano, to test whether the observed gains are specific to Qwen3. Across these additional settings, SEER consistently outperforms the strongest baselines, suggesting that the effectiveness of information-bottleneck-driven context selection is not tied to a particular benchmark type or model family.

Table 5 reports the results on SWE-Bench and WebArena. On SWE-Bench Verified, SEER improves over the best baseline from 10.8 to 11.4 with Qwen3-8B, and from 12.8 to 13.6 with Qwen3-32B. On WebArena, SEER also achieves the best performance on both evaluated subsets and both model scales. For example, with Qwen3-32B, SEER improves the score on the shopping subset from 15.1 to 19.8, and on the gitlab subset from 6.8 to 8.3. These results indicate that the gains of SEER extend to both software-engineering and web-interaction environments, which differ substantially from the SQL-based investigation and API-based task settings used in the main experiments.

Table 6 reports the results with Llama-3.1-8B-Instruct and GPT-4.1-nano. The improvements remain consistent across both open-source and API-based models. On ExCyTIn-Bench, SEER improves ACC from 8.0 to 13.8 with Llama-3.1-8B-Instruct and from 14.2 to 16.2 with GPT-4.1-nano. On AppWorld-N, SEER also achieves the best TGC and SGC for both models. These results suggest that the benefit of SEER does not come from model-specific behavior of Qwen3, but from a more general mechanism for selecting compact and decision-relevant context during test-time agent execution.

Table 7: **Paired statistical robustness analysis across benchmarks.** We report the improvement of SEER over the strongest baseline, paired bootstrap 95% confidence intervals, and win/tie/loss statistics. We use $10{,}000$ bootstrap replicates and pair all comparisons on the same evaluation instances. All performance differences are reported on the same $[0, 100]$ scale as the corresponding benchmark metrics.

Benchmark	Backbone	Metric	Strongest Baseline	ΔPerformance	Paired Bootstrap 95% CI	Win/Tie/Loss
ExCyTIn-Bench	Qwen3-8B	ACC	ACE	$+8.4$	$[+5.6, +11.2]$	$15.2\%/78.0\%/6.8\%$
		RWD	ACE	$+8.5$	$[+5.9, +11.1]$	$36.4\%/35.8\%/27.8\%$
ExCyTIn-Bench	Qwen3-32B	ACC	ACE	$+9.4$	$[+6.6, +12.2]$	$15.8\%/77.8\%/6.4\%$
		RWD	ExpeL / ACE	$+8.4$	$[+5.8, +11.0]$	$35.9\%/36.7\%/27.4\%$
AppWorld-N	Qwen3-8B	TGC	ExpeL / ACE	$+3.5$	$[+1.2, +5.8]$	$13.1\%/77.3\%/9.6\%$
		SGC	ExpeL / ReAct	$+3.5$	$[+1.3, +5.7]$	$7.1\%/89.3\%/3.6\%$
AppWorld-C	Qwen3-8B	TGC	ExpeL	$+3.2$	$[+0.7, +5.7]$	$9.1\%/85.0\%/5.9\%$
		SGC	ExpeL	$+1.6$	$[-0.1, +3.3]$	$2.9\%/95.8\%/1.3\%$
AppWorld-N	Qwen3-32B	TGC	ExpeL	$+5.5$	$[+2.9, +8.1]$	$15.5\%/74.5\%/10.0\%$
		SGC	ExpeL / ACE / ReAct	$+1.8$	$[+0.1, +3.5]$	$5.4\%/91.0\%/3.6\%$
AppWorld-C	Qwen3-32B	TGC	ACE	$+1.7$	$[-0.4, +3.8]$	$8.4\%/84.9\%/6.7\%$
		SGC	ExpeL / ACE	$+3.6$	$[+1.2, +6.0]$	$5.8\%/92.0\%/2.2\%$
SWE-Bench Verified	Qwen3-8B	Resolved	ACE	$+0.6$	$[-1.4, +2.6]$	$5.0\%/90.6\%/4.4\%$
SWE-Bench Verified	Qwen3-32B	Resolved	ACE	$+0.8$	$[-1.2, +2.8]$	$5.8\%/89.2\%/5.0\%$
WebArena-shopping	Qwen3-8B	Success	ExpeL	$+3.5$	$[+0.4, +6.6]$	$9.8\%/83.9\%/6.3\%$
WebArena-gitlab	Qwen3-8B	Success	ExpeL	$+1.5$	$[-1.0, +4.0]$	$5.4\%/90.7\%/3.9\%$
WebArena-shopping	Qwen3-32B	Success	ExpeL	$+4.7$	$[+1.3, +8.1]$	$12.2\%/80.3\%/7.5\%$
WebArena-gitlab	Qwen3-32B	Success	ExpeL	$+1.5$	$[-1.1, +4.1]$	$6.1\%/89.3\%/4.6\%$

E.3 Statistical Robustness and Paired Bootstrap Analysis

Aggregate benchmark scores may obscure whether performance gains are broadly distributed across instances or driven by a small number of favorable cases. We therefore conduct paired statistical analyses on ExCyTIn-Bench, both AppWorld splits, SWE-Bench Verified, and the evaluated WebArena subsets. For each comparison, we evaluate SEER and the strongest baseline on the same instances and use $10{,}000$ paired bootstrap replicates to estimate 95% confidence intervals for their performance difference.

We match the resampling unit to the evaluation structure of each benchmark. For ExCyTIn-Bench, we resample questions within each attack campaign. For AppWorld, we resample scenarios while keeping all three tasks in each scenario together, and recompute both TGC and SGC. For SWE-Bench Verified and WebArena, we resample software issues and web tasks, respectively. We compute win/tie/loss statistics at the question level for ExCyTIn-Bench, the task level for AppWorld TGC, SWE-Bench, and WebArena, and the scenario level for AppWorld SGC.

Table 7 complements the aggregate results with paired instance-level comparisons. We make three observations. (1) On ExCyTIn-Bench, the confidence intervals are strictly positive for both ACC and RWD at both model scales, consistent with the substantial aggregate gains of 8.4–9.4 points. The win/tie/loss statistics also show that SEER improves more instances than it degrades. (2) On AppWorld, the paired analysis supports the gains on AppWorld-N for both backbones, as well as AppWorld-C TGC with Qwen3-8B and SGC with Qwen3-32B. The smaller AppWorld-C SGC gain with Qwen3-8B and TGC gain with Qwen3-32B have confidence intervals that include zero, so we interpret these two comparisons as directionally favorable rather than statistically resolved. (3) On the additional benchmarks, WebArena-shopping shows strictly positive intervals for both backbones, whereas the intervals for SWE-Bench Verified and WebArena-gitlab include zero. These results indicate that the strongest gains are broadly supported by paired statistical analysis, while several smaller differences should be interpreted more conservatively.

E.4 Inference-Time Cost Analysis

Although SEER is training-free, it introduces additional inference-time computation through modular evidence extraction, scoring, and context curation. To better understand this cost, we report a compute analysis on AppWorld-N using Qwen3-8B and GPT-4.1-nano. We compare SEER with ExpeL and ACE in terms of task goal completion (TGC), wall-clock time per episode, total LLM tokens per episode, and total LLM calls per episode. Wall-clock time is reported in seconds. Since GPT-4.1-nano

Table 8: **Inference-time cost analysis on AppWorld-N.** We report TGC, wall-clock time per episode, total LLM tokens per episode, and total LLM calls per episode for SEER and baselines. Wall-clock time is measured in seconds. The bold font indicates the best TGC within each backbone.

Backbone	Methods	TGC	Wall-clock Time / Ep.	Total LLM Tokens / Ep.	Total LLM Calls / Ep.
Qwen3-8B	ExpeL	17.9	54.8	235,149	23.3
	ACE	17.9	**16.0**	**309,979**	**30.0**
	SEER	**21.4**	24.4	315,928	57.4
GPT-4.1-nano	ExpeL	4.2	4.4	**62,501**	**11.4**
	ACE	3.0	**3.5**	116,083	15.0
	SEER	**5.6**	4.3	112,091	27.8

Table 9: **Robustness to proxy taxonomy granularity on ExCyTIn-Bench.** We evaluate SEER with different proxy taxonomies using Qwen3-32B. The bold font indicates the best result and an underline indicates the second-best result. All results are scaled to the range $[0, 100]$.

Model	Method	ACC	RWD
Qwen3-32B	ACE (best-performing baseline)	16.3	17.7
	SEER (3-way)	23.9	25.3
	SEER (7-way)	**25.7**	**26.1**
	SEER (15-way)	25.0	25.3

is evaluated via API while Qwen3-8B is evaluated under a different serving setup, absolute runtime numbers are not directly comparable across backbones; therefore, runtime comparisons are most meaningful within the same backbone.

Table 8 shows that SEER achieves the best TGC on both backbones. On Qwen3-8B, SEER reaches 21.4 TGC, improving over both ExpeL and ACE (17.9) by 3.5 points. Although SEER uses more LLM calls and slightly more tokens than ACE, its wall-clock time remains substantially lower than ExpeL (24.4 vs. 54.8 seconds), while providing markedly stronger task performance. Compared with ACE, SEER incurs additional latency (24.4 vs. 16.0 seconds) in exchange for the 3.5-point TGC gain.

On GPT-4.1-nano, SEER similarly achieves the best TGC of 5.6, compared with 4.2 for ExpeL and 3.0 for ACE. Its wall-clock time is slightly lower than ExpeL (4.3 vs. 4.4 seconds) and moderately higher than ACE (3.5 seconds). In terms of total tokens, SEER uses fewer tokens than ACE (112,091 vs. 116,083), although more than ExpeL. These results show that the additional computation introduced by SEER yields consistent performance gains without causing disproportionate increases in end-to-end latency.

We also emphasize that total LLM call count should not be interpreted as a direct proxy for effective computational overhead. For SEER, the larger number of calls mainly comes from modular internal operations such as evidence extraction and proxy-outcome scoring, which are typically shorter and more structured than full agent-environment interaction turns. For example, on Qwen3-8B, SEER nearly doubles the number of calls relative to ACE (57.4 vs. 30.0), while their total token usage remains similar (315,928 vs. 309,979). Likewise, on GPT-4.1-nano, SEER uses substantially more calls than ACE but fewer total tokens. Therefore, wall-clock time and total token usage provide a more faithful view of inference-time cost than call count alone.

E.5 Robustness to Proxy Taxonomy Design

SEER uses a discrete proxy-outcome space to estimate the uncertainty of candidate context states. In the main experiments, this proxy-outcome space is instantiated with benchmark-specific functional categories, such as successful progress, empty-but-valid results, and different types of execution or reasoning failures. This design raises a natural question: whether the performance of SEER depends strongly on a particular hand-designed taxonomy. To examine this issue, we conduct two robustness

Table 10: **Cross-benchmark proxy taxonomy transfer on AppWorld-N.** We evaluate whether the proxy taxonomy structure used for ExCyTIn-Bench can be transferred to AppWorld under Qwen3-32B. The bold font indicates the best result and an underline indicates the second-best result. All results are scaled to the range $[0, 100]$.

Model	Method	TGC	SGC
Qwen3-32B	ExpeL (best-performing baseline)	19.1	7.1
	SEER (AppWorld taxonomy)	**24.6**	**8.9**
	SEER (ExCyTIn-Bench taxonomy)	<u>24.0</u>	**8.9**

studies. First, we vary the granularity of the proxy taxonomy on ExCyTIn-Bench. Second, we evaluate whether a taxonomy structure designed for ExCyTIn-Bench can be transferred to AppWorld.

Table 9 reports the results with 3-way, 7-way, and 15-way proxy taxonomies on ExCyTIn-Bench. The 3-way taxonomy corresponds to a coarse partition of proxy outcomes, such as success, error, and empty result. The 7-way taxonomy is the default setting used in the main experiments. The 15-way taxonomy further refines the outcome space into more fine-grained categories. The default 7-way taxonomy achieves the strongest result of $25.7/26.1$ ACC/RWD, while both the coarser 3-way taxonomy ($23.9/25.3$) and the finer 15-way taxonomy ($25.0/25.3$) remain close. Notably, even the 3-way variant substantially outperforms the best-performing baseline, ACE, by 7.6 points on both ACC and RWD. These results indicate that SEER is robust to substantial changes in proxy-taxonomy granularity, although the default intermediate granularity provides the strongest overall performance.

Table 10 further examines cross-benchmark transfer of the proxy taxonomy. Since ExCyTIn-Bench and AppWorld have different action and feedback semantics, directly reusing the exact surface category names is not meaningful. Instead, we perform a structure-preserving transfer: we keep the core functional roles of the ExCyTIn-Bench taxonomy, including successful progress, empty-but-valid outcomes, and several distinct failure modes, and instantiate these roles on AppWorld. This setting is stricter than designing a new AppWorld-specific taxonomy from scratch, because the high-level decomposition of proxy outcomes is inherited from ExCyTIn-Bench.

The transferred taxonomy remains highly stable on AppWorld-N. With Qwen3-32B, SEER obtains $24.0/8.9$ TGC/SGC using the transferred ExCyTIn-Bench taxonomy, compared with $24.6/8.9$ using the AppWorld-specific taxonomy. Both variants clearly outperform the best-performing baseline, ExpeL, which obtains $19.1/7.1$. These results suggest that SEER does not require a narrowly benchmark-specific label set, but instead benefits from a reasonable functional partition of proxy outcomes that generalizes across environments.

E.6 Prompt Robustness Analysis

SEER contains several structured-output stages, including evidence extraction, action generation and outcome prediction, and evidence curation. These stages are implemented with frozen LLMs and therefore may be affected by prompt phrasing, output-format adherence, and parsing reliability. This sensitivity is not unique to SEER, since many frozen-LLM agent pipelines rely on intermediate reasoning traces or structured outputs. Nevertheless, prompt robustness is important for evaluating whether the observed gains come from the overall framework design or from a carefully tuned prompt template.

In our implementation, the prompts used by SEER are structured but not heavily hand-crafted. Rather than performing manual prompt search, we generate the prompts from a simple meta-prompt and use constrained structured outputs, such as JSON objects and typed fields, to reduce parsing ambiguity. To further evaluate robustness, we conduct a prompt-variation study on ExCyTIn-Bench with Qwen3-8B. For each major prompt-sensitive stage, we keep the same meta-prompt and ask ChatGPT to generate two independent prompt variants. These variants are then directly inserted into the SEER pipeline without manual filtering, editing, or repeated tuning. We vary one stage at a time while keeping the remaining stages unchanged.

Table 11 shows that SEER remains stable under independently generated prompt variants. The default prompt obtains an ACC/RWD score of $23.0/24.5$. Across all six prompt variants, ACC ranges

Table 11: **Prompt robustness analysis on ExCyTIn-Bench.** We evaluate SEER with independently generated prompt variants for different structured-output stages using Qwen3-8B. All variants are used directly without manual filtering or editing. All results are scaled to the range $[0, 100]$.

Stage	Prompt Version	ACC	RWD
–	Default version	23.0	24.5
Evidence Extraction	v1	23.3	24.1
	v2	22.6	24.4
Action Generation & Outcome Prediction	v1	23.4	24.8
	v2	23.2	24.0
Evidence Curation	v1	22.8	24.7
	v2	**23.7**	**25.1**

Table 12: **Terminal evidence-pool sizes on ExCyTIn-Bench with Qwen3-32B.** We report the mean, 95th percentile (P95), and maximum pool sizes of SEER, together with the mean pool size of the *Flooded* variant that retains all extracted evidence.

Task Subset	Avg. Steps	SEER Pool Units Mean / P95 / Max	SEER Pool Tokens Mean / P95 / Max	Flooded Mean Units / Tokens
All tasks	17.3	11.8 / 22 / 29	384 / 716 / 903	39.6 / 1,283
Hard tasks	22.8	15.9 / 27 / 33	526 / 842 / 1,087	52.8 / 1,748
Longest 25%	28.6	18.7 / 31 / 37	624 / 964 / 1,241	61.7 / 2,061

from 22.6 to 23.7, while RWD ranges from 24.0 to 25.1. All variants therefore remain close to the default, with maximum absolute deviations of only 0.7 points in ACC and 0.6 points in RWD. No stage exhibits a consistent performance degradation under alternative prompt templates, and several variants slightly improve one or both metrics. For example, the evidence-curation v2 prompt achieves the strongest result of $23.7/25.1$.

E.7 Evidence-Pool Growth on Long-Horizon Episodes

Although SEER selectively admits evidence, the retained evidence pool can still grow as interactions become longer. We therefore examine how large the pool becomes on difficult and long-horizon episodes. In SEER, thresholded admission prevents evidence with insufficient uncertainty reduction per unit cost from entering the pool, while the curator periodically consolidates the retained evidence by merging redundancy and shortening verbose descriptions. In our implementation, curation is triggered after every 10 newly admitted evidence units. For comparison, we use the *Flooded* variant from Section 4.3, which uses the same evidence extractor but retains all extracted evidence.

Table 12 reports terminal evidence-pool sizes on ExCyTIn-Bench with Qwen3-32B. We consider all episodes, difficult episodes from the hard and challenging groups, and the longest 25% of episodes ranked by interaction steps.

Table 12 shows that the evidence pool grows moderately with trajectory length. Even in the longest 25% of episodes, SEER retains only 18.7 evidence units and 624 tokens on average, compared with 61.7 units and 2,061 tokens for *Flooded*. Across all evaluated episodes, the largest observed terminal pool contains 37 units and 1,241 tokens. Moreover, the mean pool size of SEER remains approximately 30% of that of *Flooded* across all three subsets, showing that thresholded admission and periodic consolidation substantially limit evidence accumulation during long-horizon reasoning.

E.8 Robustness to Heterogeneous Module Assignments

By default, SEER uses the agent's base LLM for evidence extraction, curation, proxy scoring, and action generation, avoiding the need to deploy additional models. However, the framework itself does not require these components to share the same model. We therefore examine heterogeneous model assignments on ExCyTIn-Bench to understand how different components depend on model capability.

Table 13: **Robustness to heterogeneous module assignments on ExCyTIn-Bench.** We vary the models used for the agent action, evidence extractor/curator, and proxy scorer. ECE measures calibration of the predicted proxy-outcome distribution, with lower values indicating better calibration. ACC and RWD are scaled to $[0, 100]$.

Action Model	Extractor / Curator	Proxy Scorer	ECE ↓	ACC	RWD
Qwen3-8B	Qwen3-8B	Qwen3-8B	0.126	23.0	24.5
Qwen3-8B	Qwen3-8B	Qwen3-32B	**0.084**	**23.8**	**25.3**
Qwen3-32B	Qwen3-32B	Qwen3-32B	**0.084**	**25.7**	**26.1**
Qwen3-32B	Qwen3-8B	Qwen3-32B	**0.084**	25.2	25.6
Qwen3-32B	Qwen3-32B	Qwen3-8B	0.126	24.4	24.8
Qwen3-32B	Qwen3-32B	Llama-3.1-8B-Instruct	0.145	23.9	24.2
Qwen3-32B	Qwen3-32B	GPT-4.1-nano	0.099	25.1	25.4

Table 14: **Parsing and semantic errors of intermediate stages.** Parsing success measures whether the generated output satisfies the required structured schema. Semantic error rates are measured by manual or environment-based verification of successfully parsed outputs.

Stage	ExCyTIn Parsing Success	AppWorld-N Parsing Success	Semantic Error Rate: ExCyTIn / AppWorld-N
Evidence extraction	99.6%	99.4%	3.1% / 3.5%
Proxy scoring	99.2%	99.0%	27.6% / 24.2%
Evidence curation	99.8%	99.6%	2.2% / 2.8%

Since the proxy scorer predicts a distribution over proxy outcomes, we additionally evaluate its calibration using multiclass expected calibration error (ECE). We compute ECE on a common set of cloned environment states: for each state, we execute the scorer's hypothetical action in the cloned environment and compare the predicted proxy distribution with the realized outcome. Lower ECE indicates better calibration.

Table 13 reveals two observations. (1) Changing the extractor and curator has a relatively small effect: replacing Qwen3-32B with Qwen3-8B for these two modules reduces ACC/RWD only from $25.7/26.1$ to $25.2/25.6$. (2) The proxy scorer is more sensitive to model capability. Using Qwen3-32B as the scorer improves the Qwen3-8B agent from $23.0/24.5$ to $23.8/25.3$, whereas replacing the Qwen3-32B scorer with Qwen3-8B or Llama-3.1-8B-Instruct yields higher ECE and lower task performance. GPT-4.1-nano achieves intermediate calibration and correspondingly competitive performance. These results suggest that scorer quality depends not only on probability calibration, but also on how well its hypothetical actions align with the behavior of the final agent.

E.9 Intermediate-Stage Reliability and Error Propagation

SEER relies on several LLM-generated intermediate outputs, including extracted evidence, proxy-outcome predictions, and curated evidence. Errors in these stages may affect subsequent evidence-selection decisions or propagate through the persistent evidence pool. We therefore analyze three aspects of intermediate-stage reliability: structured-output parsing, proxy calibration, and semantic-error propagation. We conduct the analysis on ExCyTIn-Bench and AppWorld-N using the default Qwen3-32B configuration.

Parsing and semantic errors. We randomly audit outputs from each generated stage. The parser checks required fields, data types, probability ranges, and probability normalization, and rejects malformed outputs before downstream use. Table 14 shows that parsing succeeds in more than 99% of cases across all stages. Semantic errors are more common, particularly for proxy scoring, where the predicted top proxy outcome can differ from the realized outcome.

Proxy calibration. We further evaluate whether the predicted proxy-outcome distributions provide meaningful uncertainty estimates. For each saved environment state, we execute the scorer's hypothetical action in a cloned environment and compare the predicted distribution with the realized proxy outcome. Table 15 reports top-1 proxy accuracy, multiclass expected calibration error (ECE), and Brier score.

Table 15: **Calibration of proxy-outcome predictions.** Lower ECE and Brier scores indicate better calibration.

Benchmark	Top-1 Proxy Accuracy	ECE ↓	Brier Score ↓
ExCyTIn-Bench	72.4%	0.084	0.157
AppWorld-N	75.8%	0.054	0.143

Table 16: **Propagation of intermediate semantic errors.** Percentages are computed among identified errors of each type. Parsing errors are excluded because malformed outputs are rejected before downstream use.

Error Type	Immediate Pool Changed	Persisted After Later Steps	Final Success Changed
Unsupported or inaccurate evidence	29.6%	10.7%	6.4%
Incorrect proxy-outcome prediction	24.3%	8.9%	5.1%
Curation deletion or distortion	34.8%	13.0%	8.7%

Table 17: **Consistency between hypothetical and final actions.** The candidate-context comparison uses the hypothetical action generated during evidence scoring, whereas the same-final-context comparison generates both actions from the final selected context. Functional agreement ignores superficial wording or formatting differences.

Action Comparison	Metric	ExCyTIn-Bench	AppWorld-N
Candidate-context hypothetical vs. final	Exact match	39.8%	34.7%
	Functional agreement	67.9%	61.8%
Same-final-context hypothetical vs. final	Exact match	48.6%	41.9%
	Functional agreement	81.7%	76.4%

The predicted distributions are informative but not perfectly calibrated. Top-1 proxy accuracy exceeds 70% on both benchmarks, while the nonzero ECE and Brier scores indicate remaining calibration error. These measurements evaluate the quality of the full proxy distribution and complement the entropy-estimation analysis in Section 4.3.

Error propagation. To measure whether semantic errors affect later reasoning, we perform controlled correction-and-replay experiments. For each identified semantic error, we replace only the erroneous intermediate output with a manually corrected version, restore the same environment state, and replay the remaining deterministic pipeline. Table 16 reports how often this correction changes the immediate evidence pool, persists after subsequent steps, and changes final task success.

Table 16 shows that most identified semantic errors do not ultimately change task success. Curation errors propagate most often because they directly modify the persistent evidence pool, while proxy-prediction errors are less persistent because they typically affect only individual admission decisions and can be offset by evidence observed later. Overall, the analysis indicates that structured-output failures are rare, while the main remaining reliability limitations arise from semantic errors in evidence extraction, proxy prediction, and curation.

E.10 Consistency Between Hypothetical and Final Actions

SEER estimates proxy uncertainty by conditioning on a hypothetical action generated from each candidate context. Since the final agent action is generated only after evidence selection, the two actions may differ, potentially introducing error into the action-conditioned uncertainty estimate. We therefore analyze their consistency on ExCyTIn-Bench and AppWorld-N.

During candidate scoring, SEER generates the hypothetical action from $S_{t-1} \cup \{e\}$, whereas the final action uses the selected pool S_t. We compare the two actions using normalized textual exact match and functional agreement, where two actions functionally agree if they perform the same operation on the same target despite wording or formatting differences. To separate action inconsistency from the effect of different contexts, we additionally regenerate a hypothetical action from the same final context used by the agent.

Table 18: **Proxy accuracy under hypothetical–final action agreement and divergence.** We partition same-final-context cases according to functional agreement between the hypothetical and final actions.

Benchmark	Aligned	Divergent
ExCyTIn-Bench	74.9%	61.2%
AppWorld-N	79.8%	62.8%

Table 17 shows that candidate-context hypothetical actions functionally agree with final actions in 67.9% and 61.8% of cases on ExCyTIn-Bench and AppWorld-N, respectively. When both actions use the same final context, agreement increases to 81.7% and 76.4%. Thus, a substantial part of the apparent divergence comes from the fact that candidate scoring and final action generation use different evidence pools.

We further examine how same-context action divergence affects proxy prediction. For each case, we compare the predicted top proxy outcome with the realized outcome after executing the final action.

Table 18 shows that action divergence reduces proxy accuracy, confirming it as one source of estimation error. Nevertheless, proxy accuracy remains above 60% even in divergent cases. Moreover, weighting the aligned and divergent groups by their observed frequencies recovers the overall proxy accuracies of 72.4% and 75.8% reported in Appendix E.9. These results show that hypothetical-action conditioning remains informative while also identifying action mismatch as a measurable limitation of the estimator.

F Related Work

Long-Horizon Context Management. As LLM agents interact with environments over extended horizons, their growing interaction histories increasingly challenge bounded context windows, making context management an important component of agentic systems (Mei et al., 2025). Existing methods address this challenge through summarization, context evolution, explicit editing, and adaptive compression. ReSum (Wu et al., 2025a) periodically condenses interaction histories into compact reasoning states, while COMPASS (Wan et al., 2026) maintains evolving progress briefs through a dedicated context manager. ACE (Zhang et al., 2025) iteratively generates, reflects on, and curates an evolving playbook-style context from agent experience. Other methods make context management itself adaptive: ContextBudget (Wu et al., 2026) learns when and how much history to compress under a token budget, while ACM (Li et al., 2026) allows agents to actively edit their context and offload information to external memory. These methods show that effective long-horizon context management can involve not only compression, but also selective retention, reflection, reorganization, and externalization of interaction information. SEER differs mainly in how it defines the value of retained evidence: starting from an Information Bottleneck formulation, it evaluates each evidence item by its marginal reduction in the agent's uncertainty over subsequent environment feedback relative to the context length it introduces.

Agent Memory and Evidence Selection. A closely related line of work externalizes interaction history into memories, experiences, or structured representations that can be selectively retrieved or updated during reasoning (Zhao et al., 2024; Zhang et al., 2025; Yu et al., 2026). ExpeL (Zhao et al., 2024) retrieves relevant successful experiences and distilled insights from past trajectories, while Agentic Memory (Yu et al., 2026) learns to manage long- and short-term memories for agents. Recent methods further introduce state-aware or structured mechanisms for memory organization and selection. SAMem (Wang et al., 2026b) conditions experiential memory on the agent's current state, MAGMA (Jiang et al., 2026) organizes memory through multiple structural graphs, and MESA (Zhao et al., 2026) dynamically selects complementary memory structures according to task demands. Closer to evidence selection, PACMS (Ghulyani et al., 2026) formulates context assembly as budget-constrained submodular selection based on semantic relevance and coverage. Thus, selecting useful information itself is not unique to SEER. The key distinction is the selection criterion: SEER assigns value to interaction-derived evidence according to its marginal reduction

in uncertainty over subsequent environment feedback per unit context cost, providing an explicit information-theoretic connection between evidence selection and subsequent reasoning.

Information Bottleneck. The Information Bottleneck (IB) principle provides a classic information-theoretic framework for learning compact representations that preserve information relevant to a target variable while discarding irrelevant information from the input (Tishby et al., 2000). Building on this principle, prior work develops tractable IB objectives, including variational formulations that make the relevance–compression trade-off compatible with deep networks (Alemi et al., 2017), and uses IB to analyze how deep models balance information preservation and compression (Tishby & Zaslavsky, 2015). IB-inspired objectives have also been studied in reinforcement learning to learn compact state representations by filtering task-irrelevant information (Yingjun & Xinwen, 2019). These approaches typically use IB as a **training-time** objective to shape internal representations. In contrast, our work uses IB as a **training-free**, **test-time** principle for context management in interactive agents. We operationalize relevance with an observable proxy outcome that summarizes subsequent environment feedback, and select evidence according to its marginal reduction in the agent's uncertainty over such feedback per unit context length. To make this criterion practical, we estimate action-conditioned proxy uncertainty using hypothetical actions without additional environment executions.