UPLIFTMEM: LEARNING SET-LEVEL UPLIFT FOR AGENT MEMORY RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents reuse external memory to guide new tasks, but effective retrieval requires learning which memory sets improve execution. Such learning relies on costly outcome feedback: ordinary retrieval observes only executed sets, while evaluating alternatives requires additional rollouts. We introduce UPLIFTMEM, which learns memory retrieval from set-level execution uplift relative to the same executor without memory. A theoretical analysis of how retrieval preferences restrict feedback coverage motivates targeted probing of alternative memory sets. Probe selection follows an expected value of sample information (EVSI) criterion, derived in closed form under a correlated Gaussian model, to allocate limited training rollouts according to their expected improvement in local retrieval decisions. The shared scorer is trained with a frozen executor and selects memory sets without test-time probes. Across ALFWorld, WebShop, and BigCodeBench, UPLIFTMEM achieves the best success rates among evaluated baselines on the main evaluation sets. Controlled fixed-store and matched probe budget evaluations further demonstrate improved memory-use decisions and more effective use of execution feedback. Code is available at `https://anonymous.4open.science/r/upliftmem-main-DB1B`.

1 INTRODUCTION

LLM-based agents retrieve past experience to guide new tasks (Shinn et al., 2023; Zhao et al., 2024), but relevant memories do not necessarily improve execution. Figure 1(a) illustrates this mismatch: when asked to find a waterproof backpack under \$60, the agent succeeds without memory, yet follows a retrieved recommendation to purchase an \$89 item. This failure motivates learning retrieval from execution feedback, with supervision that reflects whether supplying memory improves the same agent's execution on the same task.

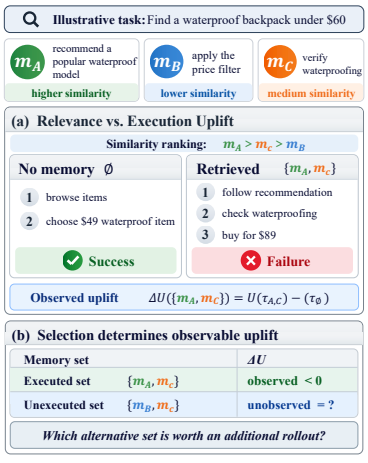

Figure 1. Learning memory retrieval from set-level execution feedback. (a) A relevant memory set may reduce utility relative to no-memory execution. (b) Only executed sets receive direct uplift labels; evaluating alternatives requires additional rollouts.

Building on outcome-aware and uplift-based retrieval (Zhang et al., 2026; Qu et al., 2025; Tu et al., 2026), we define execution uplift as the expected utility difference between execution with complete memory and execution without memory under the same executor. Observed utility differences supply the signal, distinguishing additional memory benefit from success attributable to the executor alone. Memories may provide complementary, redundant, or conflicting guidance (Xu et al., 2025b). A memory's effect on execution can thus depend on the other memories used alongside it. To capture these interactions, we model execution uplift at the level of complete memory sets.

Obtaining this supervision, however, requires costly agent executions. Each new label requires executing a candidate memory set and comparing its utility with a no-memory reference. Ordinary retrieval supplies direct feedback only for the set it executes. In Figure 1(b), the negative uplift observed for $\{m_A, m_C\}$ provides no direct label for $\{m_B, m_C\}$. In other words, *if a memory set is*

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.

never selected, its actual execution uplift remains unverified by direct observations. Consequently, an underestimated set may remain rarely selected and receive little feedback to correct that estimate. Obtaining labels for untested sets requires additional rollouts, but the number of feasible combinations makes exhaustive labeling impractical under a limited budget.

The goal is therefore to learn effective memory retrieval by allocating a limited rollout budget to useful execution feedback. To address this problem, we introduce UPLIFTMEM, which couples set-level uplift learning with execution-feedback acquisition under a limited rollout budget. Our theoretical analysis characterizes how retrieval preferences restrict direct-feedback coverage. This finding motivates additional training executions, called probes, that provide feedback for alternative memory sets ordinary retrieval may leave untested. For local memory-set comparisons, the analysis further yields a closed-form expression for the expected value of sample information (EVSI) under a correlated Gaussian surrogate (Frazier et al., 2009). The resulting decision-aware acquisition rule combines resolvable uncertainty with predicted utility differences to prioritize observations expected to improve retrieval decisions. Correlations in the surrogate allow one observation to inform multiple set comparisons. The acquired outcomes then supervise a shared value model that evaluates each candidate addition in the context already selected, while no-memory execution serves as both the reference and an explicit alternative.

With executors backed by Qwen3-4B and Qwen3-8B (Yang et al., 2025), UPLIFTMEM achieves the best success rates among the evaluated systems on both ALFWorld splits (Shridhar et al., 2021), WebShop (Yao et al., 2022), and the overall BigCodeBench evaluation set (Zhuo et al., 2025). Fixed-store evaluations on MemSyco-Bench (Xiang et al., 2026) isolate the contribution of retrieval by holding memory content and the executor unchanged. Relative to native retrieval, decision accuracy averaged over five memory stores and the three scenarios assessing when to use memory increases from 41.7% to 66.5% with Qwen3-4B and from 44.4% to 68.8% with Qwen3-8B.

2 RELATED WORK

2.1 EXPERIENCE-BASED AGENT MEMORY

Agent memory has evolved from retaining execution feedback and trajectories to distilling reusable procedural and reasoning guidance (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024; Fang et al., 2026b; Ouyang et al., 2026). Recent systems further organize, revise, and selectively maintain stored experience using later interactions or learned memory-management policies (Xu et al., 2025a; Fang et al., 2026a; Zhou et al., 2025; Yan et al., 2026). As memory becomes an adaptive component of the agent, its benefit increasingly depends on which parts of accumulated experience are supplied to the executor for a particular task.

2.2 LEARNING TO RETRIEVE AGENT MEMORY

Memory retrieval has correspondingly moved beyond semantic relevance toward downstream usefulness. Reasoning-aware and utility-based methods learn which memories are likely to improve task performance, while uplift-style and paired-execution approaches estimate contribution relative to behavior without external guidance (Li et al., 2026a; Zhang et al., 2026; Qu et al., 2025; Tu et al., 2026). When several memories are used together, their effects can also depend on the surrounding context, motivating shared-context attribution and joint adaptation of retrieved knowledge (Xu et al., 2025b; Li et al., 2026b).

Outcome-based retrieval further changes how training evidence is obtained. Direct outcome labels are available only for memory sets that are executed, while evaluating alternatives requires additional rollouts. Related work on counterfactual learning and active acquisition studies how selectively observed or costly feedback should be used for learning and decision making (Swaminathan & Joachims, 2015; Roy & McCallum, 2001; Houlsby et al., 2011; Kirsch et al., 2019; Frazier et al., 2009). UpliftMem formulates these coupled challenges as a learning problem, where complete memory sets are valued by execution uplift and additional rollouts are allocated according to their expected value for improving retrieval decisions.

3 PRELIMINARIES

3.1 INTERACTIVE AGENT WITH EXTERNAL MEMORY

Following prior memory-augmented agent architectures (Sumers et al., 2024; Zhao et al., 2024; Wang et al., 2024), we consider an interactive agent comprising a memory controller and an executor. The controller manages an external pool of reusable experience through memory retrieval and updates, while the executor interacts with the environment to complete tasks. Within the controller, the retriever selects memory sets to support execution.

At episode t, the retriever receives a task context c and a memory pool $\mathcal{M}_t$ of self-contained experience units m. It selects a subset $S \subseteq \mathcal{M}_t$ as auxiliary context for the executor. The controller then uses the resulting trajectory and execution feedback to write new memory candidates, revise them, and determine their admission to the next pool $\mathcal{M}_{t+1}$.

3.2 PROBLEM FORMULATION

Task performance alone cannot distinguish equally successful agent executions with substantially different resource requirements (Kapoor et al., 2024). We therefore incorporate execution cost to make resource efficiency an explicit component of utility. For an execution trajectory τ, let $R(\tau) \in [0, 1]$ denote task performance and $C(\tau) \in [0, C_{\max}]$ denote the cost assigned by the benchmark protocol, with $C_{\max} > 0$ as the cap. We define execution utility as

$$U(\tau) = R(\tau) \left[1 + \gamma \mathbb{I}[R(\tau) \geq \rho] \left(1 - \frac{C(\tau)}{C_{\max}}\right)\right], \tag{1}$$

where $\mathbb{I}[\cdot]$ is the indicator function, $\gamma \geq 0$ weights the efficiency bonus, and $\rho \in [0, 1]$ sets the performance threshold. Performance determines the base utility, and lower cost increases the bonus once the threshold is met.

For task c executed with memory set S, the expected utility is $V(c, S) = \mathbb{E}[U(\tau(c, S))]$. The retriever uses a learnable set-value model $G_\theta(c, S)$ to predict the expected utility gain relative to no-memory execution. Its output $S_\theta(c, \mathcal{M}_t)$ satisfies the retrieval constraints, with the empty set allowed. For a given memory pool $\mathcal{M}_t$, the downstream learning objective is

$$\theta^* \in \arg\max_\theta \mathbb{E}\left[V\big(c, S_\theta(c, \mathcal{M}_t)\big)\right]. \tag{2}$$

4 METHODOLOGY

As shown in Figure 2, our trainable retriever is parameterized by $G_\theta(c, S)$, which scores complete memory sets by their predicted execution uplift. Similarity-based recall supplies candidate memories, and conditional gains computed from G_θ guide sequential set construction. Because set values are learned from execution outcomes, retrieval also determines which memory sets receive direct feedback, leaving disfavored alternatives poorly observed unless additional rollouts are performed. Under a limited rollout budget, UPLIFTMEM therefore uses expected value of sample information (EVSI) to probe alternative sets whose outcomes are expected to improve retrieval decisions.

The retrieved set and selected probes are executed by the same frozen agent and compared with no-memory execution to obtain uplift observations. Retrieved executions supervise both set value and the sampled retrieval path, whereas probes provide additional value observations. After value-only warm-up, these signals jointly update the shared uplift model. Completed trajectories are also distilled into memory candidates, which are validated and revised when needed before admission to the memory pool. Complete training and inference procedure is shown in Algorithm 1.

4.1 DECISION VALUE OF MEMORY SETS

Motivated by outcome-aware retrieval methods that measure the benefit of external content relative to model-only behavior (Qu et al., 2025; Tu et al., 2026), we use the same executor without external memory as the reference and define the uplift of a memory set S as

$$\Delta(c, S) = V(c, S) - V(c, \emptyset). \tag{3}$$

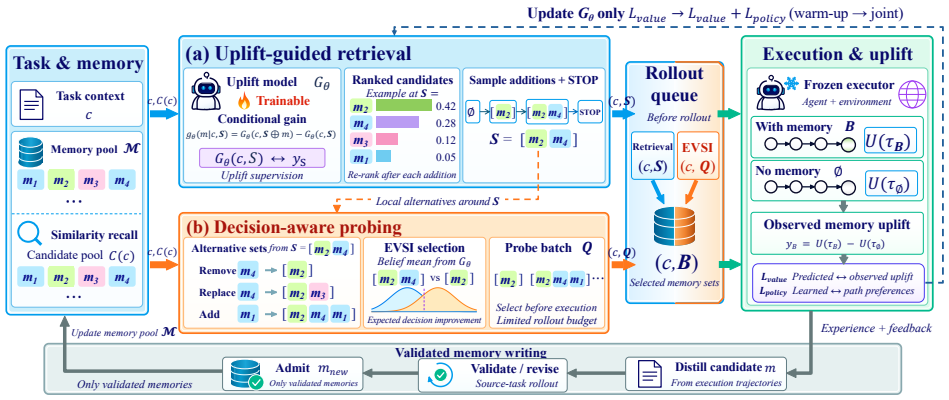

Figure 2: Overview of UPLIFTMEM. A shared scorer G_θ guides retrieval and EVSI-based probing, trained with value and policy supervision while the executor remains frozen. Validated writing supplies additional labels and expands the memory pool.

Accordingly, we train a shared uplift model $G_\theta(c, S)$ to predict expected set uplift, supervised by the observed utility difference from paired executions with and without S,

$$y_S = U\big(\tau(c, S)\big) - U\big(\tau(c, \emptyset)\big). \tag{4}$$

Because $V(c, \emptyset)$ is independent of θ, replacing V with Δ in Eq. 2 does not change the optimal parameters. The learned model G_θ approximates this uplift, with $G_\theta(c, \emptyset) = 0$ as the no-memory reference. Each observation corresponds to the complete memory set used during execution. The set-valued formulation follows consistent memory construction and evaluation rules, with the search and scoring details provided in Appendices B.1 and C.1. Appendix A.4 distinguishes uncertainty about expected uplift from variability in its observed labels.

4.2 SET CONSTRUCTION AND EXECUTION FEEDBACK

For each task context c, similarity-based recall first constructs a candidate memory pool $\mathcal{C}(c) \subseteq \mathcal{M}_t$. Starting from $S = \emptyset$, the retriever evaluates an additional memory through its conditional gain,

$$g_\theta(m \mid c, S) = G_\theta(c, S \cup \{m\}) - G_\theta(c, S), \qquad g_\theta(\texttt{STOP} \mid c, S) = 0. \tag{5}$$

The action set $\mathcal{A}(c, S)$ contains feasible memories in $\mathcal{C}(c) \setminus S$ and the stopping action `STOP`, which returns the current set. Retrieval also terminates at the maximum memory count or when no feasible addition remains. During training, the next action $a \in \mathcal{A}$ is sampled according to the conditional gains:

$$\pi_\theta(a \mid c, S) = \frac{\exp(g_\theta(a \mid c, S)/\tau_{\text{tr}})}{\sum_{a' \in \mathcal{A}(c,S)} \exp(g_\theta(a' \mid c, S)/\tau_{\text{tr}})}, \tag{6}$$

where $\tau_{\text{tr}} > 0$ controls the sampling concentration. At test time, deterministic beam search of width two returns the highest-valued set retained across depths, including $\emptyset$. It can retain intermediate sets with negative conditional gains so that later additions can reveal complementarity. Appendix B.1 specifies feasibility, deduplication, and stopping.

Under this retrieval strategy, only the selected terminal set receives direct execution feedback, while alternative memory sets may remain unobserved and lack corrective supervision. The resulting dependence between retrieval preference and feedback coverage is characterized by the following proposition.

Proposition 1 (Selective retrieval suppresses alternative observations)**.** *Fix a task c, its candidate pool, retrieval constraints, and model parameters. Let $\mathcal{P}(B)$ denote all feasible retrieval paths that return a memory set B. For each path $\mathbf{a} = (a_1, \ldots, a_L)$ of L sampled actions, let a_t denote its t-th action and S_{t-1} the preceding memory set. Define the cumulative action deficit as*

$$d(\mathbf{a}) = \sum_{t=1}^{L} \left[\max_{a \in \mathcal{A}(c, S_{t-1})} g_\theta(a \mid c, S_{t-1}) - g_\theta(a_t \mid c, S_{t-1}) \right]. \tag{7}$$

The probability p_B that ordinary retrieval returns B satisfies

$$p_B \le \sum_{\mathbf{a}\in\mathcal{P}(B)} \exp\left(-\frac{d(\mathbf{a})}{\tau_{\mathrm{tr}}}\right). \tag{8}$$

The bound shows how locally disfavored actions suppress direct feedback for the sets they construct, limiting correction of their estimated values. Appendices A.1–A.3 extend this result to evidence arrival, identification, and changing policies, motivating targeted executions of informative alternatives.

4.3 Decision-Aware Probing

To address limited feedback for sets disfavored by retrieval (Section 4.2), UpliftMem probes alternative memory sets during training. Under a limited rollout budget, an observation is valuable when it can improve a retrieval comparison, following the value-of-information principle (Howard, 1966; Chaloner & Verdinelli, 1995).

Because memory-set values are statistically related, one observation can reduce uncertainty across several retrieval comparisons. Following correlated-belief knowledge-gradient models (Frazier et al., 2009), we use a local Gaussian surrogate conditioned on the collected uplift observations $\mathcal{D}$,

$$\widetilde{G}(c,S) = G_\theta(c,S) + \varphi(c,S)^\top \xi, \qquad \xi \sim \mathcal{N}(0,\Sigma). \tag{9}$$

Here G_θ is the conditional mean uplift prediction given $\mathcal{D}$, $\varphi(c,S)\in\mathbb{R}^d$ is a task-set feature vector of dimension d, $\xi\in\mathbb{R}^d$ is a zero-mean Gaussian latent vector, and $\Sigma\in\mathbb{R}^{d\times d}$ is its positive-semidefinite covariance matrix. The surrogate describes uncertainty about expected uplift, while a separate noise model describes variability in execution labels. Appendix A.4 defines the measurement model, including shared no-memory references.

For a local retrieval decision j comparing two candidate sets S_j^a and S_j^b, we define their uncertain uplift difference as $\widetilde{g}_j = \widetilde{G}(c,S_j^a) - \widetilde{G}(c,S_j^b)$, with conditional mean $\mu_j = \mathbb{E}[\widetilde{g}_j \mid \mathcal{D}]$. A prospective probe collection Q would produce additional observations y_Q. The uncertainty these observations can resolve about decision j is

$$s_j^2(Q) = \mathrm{Var}(\widetilde{g}_j \mid \mathcal{D}) - \mathrm{Var}(\widetilde{g}_j \mid \mathcal{D}, y_Q), \tag{10}$$

where $\mu_j' = \mathbb{E}[\widetilde{g}_j \mid \mathcal{D}, y_Q]$ is the posterior mean used to revise the comparison. Appendix A.4 derives their pre-execution distribution under Gaussian conditioning.

The following theorem gives the expected improvement in choosing between sets in closed form, using both resolved uncertainty $s_j(Q)$ and the current comparison margin μ_j.

Theorem 2 (Value of a probe for a decision)**.** *Under the joint Gaussian model, for $s_j(Q) > 0$, the expected value of sample information (EVSI) for comparison j is*

$$E_j(Q) = \mathbb{E}_{y_Q\mid\mathcal{D}}\left[\left(\mu_j'\right)_+\right] - (\mu_j)_+ = s_j(Q)\psi\left(\frac{|\mu_j|}{s_j(Q)}\right), \qquad \psi(t) = \phi(t) - t\Phi(-t), \tag{11}$$

where $(x)_+ = \max\{x,0\}$, ϕ and Φ are the standard normal density and cumulative distribution functions, and $E_j(Q) = 0$ when $s_j(Q) = 0$.

This score combines resolvable uncertainty with the current predicted utility difference and measures expected decision improvement in utility units. It can therefore compare prospective observations without simulating their outcomes. Appendix A.6 relates binary EVSI to information gain and distinguishes local comparisons from selecting a single set among multiple alternatives.

We aggregate EVSI over the local comparison family $\mathcal{G}$ as $H(Q) = \sum_{j\in\mathcal{G}} E_j(Q)$ to select a probe batch Q from the feasible probe pool. For each candidate q, its marginal acquisition value is

$$V_t(q) = \sum_j E_j(Q_t\cup\{q\}) - \sum_j E_j(Q_t), \tag{12}$$

where $Q_t \subseteq Q$ denote the probes selected after t steps. The next probe is sampled according to a softmax over $V_t(q)$, with the marginal values updated after each selection. Appendix A.7 derives this prospective increment and its relationship to subsequent learning, while Appendix B.2 gives the candidate construction and batch procedure.

4.4 LEARNING FROM EXECUTION FEEDBACK

The retrieved set S, the selected probe sets Q, and validated memory candidates are executed against the same no-memory reference to provide uplift observations for training G_θ.

Regression supervision fits the uplift magnitude:

$$\mathcal{L}_{\text{reg}} = \mathbb{E}[w_{\text{mag}}(y_S)\texttt{Huber}_{\delta_H}\left(G_\theta(c,S) - y_S\right)], \tag{13}$$

where w_{mag} weights observations by uplift magnitude.

Since the no-memory baseline also determines whether a memory set is beneficial or harmful, we explicitly supervise the prediction sign while excluding near-zero observations:

$$\mathcal{L}_{\text{sign}} = \mathbb{E}_{|y_S|>\epsilon_{\text{sign}}}[\texttt{BCEWithLogits}\left(G_\theta(c,S), \mathbb{I}[y_S > 0]\right)], \tag{14}$$

where $\texttt{BCEWithLogits}(\cdot)$ denotes binary cross-entropy applied directly to the predicted uplift as a logit.

Retrieval further requires distinguishing competing sets for the same task, so we impose pairwise supervision on their relative uplift. For pairs satisfying $|y_{S_a} - y_{S_b}| > \epsilon_{\text{rank}}$,

$$\mathcal{L}_{\text{pair}} = \mathbb{E}_{\text{pair}}[\texttt{softplus}\left(-\texttt{sign}(y_{S_a} - y_{S_b})\left[G_\theta(c,S_a) - G_\theta(c,S_b)\right]\right)]. \tag{15}$$

Here, $\texttt{sign}(y_{S_a} - y_{S_b})$ indicates which set has the larger observed uplift.

The overall value objective combines these complementary signals

$$\mathcal{L}_{\text{value}} = \lambda_{\text{reg}}\mathcal{L}_{\text{reg}} + \lambda_{\text{sign}}\mathcal{L}_{\text{sign}} + \lambda_{\text{pair}}\mathcal{L}_{\text{pair}}. \tag{16}$$

Appendix C.2 specifies the weighting and sample-eligibility rules.

The same parameters θ also determine the retrieval policy π_θ through the conditional gains in Eqs. 5 and 6. For path i on task c_i, let $L_i \geq 1$ be its number of sampled actions, $a_{i,t}$ its t-th action, and $S_{i,t-1}$ the preceding set. Using paths whose terminal sets are executed, we optimize

$$\mathcal{L}_{\text{policy}} = -\mathbb{E}_i\left[\frac{\texttt{sg}(\omega_i)}{L_i}\sum_{t=1}^{L_i}\log\pi_\theta(a_{i,t} \mid c_i, S_{i,t-1})\right]. \tag{17}$$

The expectation averages eligible paths, ω_i is a signed and clipped weight derived from their observed uplift, and $\texttt{sg}(\cdot)$ denotes the stop-gradient operator. Positive weights reinforce sampled actions, negative weights suppress them, and zero weights contribute no policy gradient. Value supervision also uses observations without a recorded retrieval path. Appendix C.3 specifies path eligibility and outcome weights.

After value-only warm-up, both objectives jointly update θ by minimizing $\mathcal{L}_{\text{value}} + \mathcal{L}_{\text{policy}}$, while the executor remains frozen. Memory writing extracts and validates new candidates from completed trajectories (Zhao et al., 2024; Cao et al., 2026), contributing a value label even when the selected candidate is not admitted. Appendix B.3 gives the revision and admission rules.

5 EXPERIMENTS

We evaluate task performance across complete agent workflows and then hold memory stores fixed to examine retrieval decisions. Component ablations test the learning and acquisition mechanisms. We then compare probe-acquisition strategies under increasing rollout budgets to evaluate feedback efficiency, followed by retrieval-budget experiments that examine sensitivity to the amount of available context.

5.1 EXPERIMENTAL SETUP

We evaluate ALFWorld (Shridhar et al., 2021) using success rate (SR) and execution steps, WebShop (Yao et al., 2022) using reward, SR, and interaction steps, and BigCodeBench (BCB) (Zhuo et al., 2025) using overall and hard-subset SR. MemSyco-Bench (Xiang et al., 2026) evaluates memory-use behavior under a dialogue-grouped 70:30 train/test split with seed 42.

Table 1: End-to-end performance on ALFWorld, WebShop, and BCB with Qwen3-4B and Qwen3-8B. SR is reported in percent. Bold and underlined values indicate the **best** and runner-up distinct results in each column, respectively.

	ALFWorld								WebShop						BCB			
	Seen SR↑		Seen Steps↓		Unseen SR↑		Unseen Steps↓		Reward↑		SR↑		Steps↓		SR↑		Hard SR↑	
Method	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**
ReAct	43.13	57.14	22.35	20.36	50.00	76.87	20.98	16.59	63.20	68.39	38.00	41.60	7.09	5.80	31.87	40.35	15.91	20.45
Random	47.14	65.00	18.54	19.46	49.25	71.64	21.73	17.80	62.22	52.87	39.20	29.80	7.26	6.13	30.70	**42.69**	13.64	13.64
BM25-RAG	59.28	75.71	18.67	15.31	52.23	70.15	19.75	17.13	57.88	64.87	37.60	41.80	7.24	6.55	33.33	40.35	11.36	15.91
Dense-RAG	50.00	67.14	20.82	17.02	52.99	68.66	21.05	17.43	56.97	64.17	34.80	38.60	7.40	6.61	34.50	39.47	15.91	11.36
MemP	50.00	54.29	19.81	18.64	58.21	49.25	19.15	20.09	52.42	58.87	35.40	37.00	6.71	7.74	34.21	39.47	18.18	11.36
MemRL	54.29	72.14	19.69	16.35	58.96	78.35	19.36	15.44	53.17	60.78	35.00	36.40	6.86	7.38	32.46	39.77	18.18	13.64
Mem0	75.71	70.00	15.77	16.57	67.16	77.61	16.82	**15.04**	70.26	58.36	43.60	33.80	5.59	**5.03**	35.09	40.64	20.45	11.36
MemGPT	57.14	52.86	19.37	21.30	65.67	64.18	19.03	20.27	67.26	65.65	40.20	40.40	6.33	5.49	38.01	41.52	18.18	11.36
A-MEM	57.86	58.57	20.29	20.99	58.96	61.19	21.15	22.33	65.14	55.48	39.60	33.20	**5.54**	5.92	33.04	38.30	18.18	18.18
LightMem	64.29	66.43	19.81	18.99	67.91	77.61	16.74	16.74	66.28	61.41	40.40	37.80	5.78	6.86	35.96	41.81	**22.73**	18.18
UpliftMem	**76.86**	**81.43**	**14.01**	**13.61**	**70.91**	**79.10**	**15.81**	16.53	**72.51**	**70.20**	**45.60**	**44.00**	5.57	5.56	**39.77**	**42.69**	20.45	**22.73**

The frozen executors are Qwen3-4B and Qwen3-8B. For each executor, the retriever is parameterized by G_θ, which is initialized from Qwen3-Reranker-0.6B (Zhang et al., 2025) and adapted separately. Baselines include ReAct (Yao et al., 2023), conventional retrieval methods, MemP (Fang et al., 2026b), MemRL (Zhang et al., 2026), Mem0 (Chhikara et al., 2025), MemGPT (Packer et al., 2023), A-MEM (Xu et al., 2025a), and LightMem (Fang et al., 2026a). End-to-end comparisons evaluate complete workflows, whereas MemSyco-Bench compares retrieval within identical memory stores. Appendix D.2 specifies these controls and the implementation-evidence view used for BCB.

We recall 50 candidate memories and cap retrieval at $K = 5$ for ALFWorld and WebShop and $K = 3$ for BCB. MemSyco-Bench uses $K = 5$, with additional tests at $K \in \{3, 10\}$. Appendix D.4 reports the model configuration. Each configuration is trained once and evaluated repeatedly with the same checkpoint, following the statistical protocol in Appendix D.5.

5.2 RQ1: End-to-End Agent Performance

Table 1 complete agent workflows across three benchmarks. UpliftMem attains the highest success rates on both ALFWorld splits and WebShop with both executors. With Qwen3-4B, it achieves 76.9% and 70.9% success on ALFWorld seen and unseen tasks, respectively, 45.60% on WebShop, and 39.77% on BCB. With Qwen3-8B, it achieves 81.43% and 79.10% on ALFWorld, 44.00% on WebShop, and 42.69% on BCB, matching the highest overall BCB result. It also obtains the highest WebShop reward with both executors and the fewest steps on ALFWorld seen tasks. We next hold memory stores fixed to examine the contribution of retrieval decisions.

5.3 RQ2: Controlled Retrieval Evaluation

MemSyco-Bench evaluates when memory should guide a response through Objective Fact, Scope Control, and Evidence Conflict, and how it should be used through Personalized Use and Valid Selection. Each scenario reports response accuracy alongside a behavioral measure, namely sycophancy (Syco) for the first three, correct-memory use (Correct) for Personalized Use, and outdated-memory use (Outdated) for Valid Selection.

To isolate retrieval effects, we replace each memory system's native retriever with UpliftMem while holding its post-extraction memory store and executor fixed. Figure 3 shows higher accuracy and lower sycophancy across Objective Fact, Scope Control, and Evidence Conflict for all five memory stores and both executors. Scope Control accuracy increases by 23.7–62.7 percentage points with Qwen3-4B, while Evidence Conflict accuracy increases by 29.1–54.6 points with Qwen3-8B. In Valid Selection, accuracy improves in all ten store–executor comparisons, with outdated-memory use decreasing in nine. These results show that retrieval decisions affect both when agents rely on memory and which memories they use, even with identical available memory content. Table 9 reports the corresponding absolute scores and inference-run variability for each memory store.

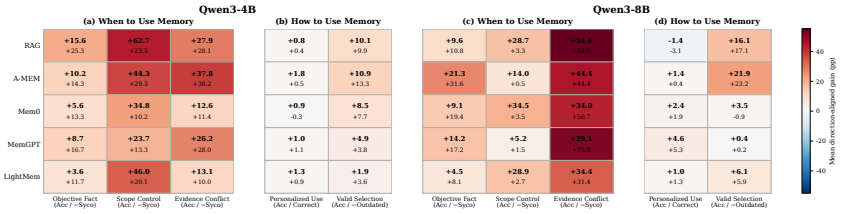

Figure 3: Retrieval gains on MemSyco-Bench with fixed memory stores and $K = 5$. Each cell shows the change from native retrieval in accuracy (top) and the indicated behavioral metric (bottom), in percentage points. Changes in sycophancy and outdated-memory use are sign-reversed so that positive values indicate improvement.

Table 2: Component ablations on ALFWorld, WebShop, and BCB with both executors.

	ALFWorld								WebShop						BCB			
	Seen SR↑		Seen Steps↓		Unseen SR↑		Unseen Steps↓		Reward↑		SR↑		Steps↓		SR↑		Hard SR↑	
Variant	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**	**4B**	**8B**
Base reranker	68.57	65.00	15.52	17.31	63.43	64.93	17.12	17.33	67.12	66.81	41.80	42.00	6.32	6.44	33.63	38.89	18.18	18.18
Warm-up only	64.76	66.19	16.69	16.86	61.19	67.16	17.36	16.65	68.82	66.66	43.60	43.20	5.71	6.46	34.80	41.23	18.18	18.18
Random probing	66.07	72.85	16.18	15.32	63.43	76.12	17.28	16.85	68.03	70.16	41.60	43.60	6.08	5.59	37.72	40.94	20.45	20.45
w/o policy loss	71.42	70.71	16.98	14.85	67.16	70.90	16.85	17.43	67.88	69.54	41.80	43.40	5.99	6.78	38.89	41.23	**25.00**	18.18
w/o value loss	70.00	75.00	15.21	14.34	68.65	73.13	16.74	17.15	68.61	68.25	43.40	43.20	5.72	6.01	37.72	41.23	20.45	18.18
UpliftMem (Full)	**76.86**	**81.43**	**14.01**	**13.61**	**70.91**	**79.10**	**15.81**	**16.53**	**72.51**	**70.20**	**45.60**	**44.00**	**5.57**	**5.56**	**39.77**	**42.69**	20.45	**22.73**

5.4 RQ3: Ablation Study

We examine the contributions of value adaptation, probe selection, and the learning objectives through five ablation variants. *Base reranker* uses the pretrained scorer without value adaptation. *Warm-up only* retains bootstrap probing and initial value-only LoRA adaptation but disables subsequent probing and scorer updates. Both variants retain online memory writing. *Random probing* replaces EVSI-guided selection with uniform sampling under the same probe budget. The remaining variants remove either the policy or value loss while retaining the other training components.

Table 2 shows that the full model achieves the highest overall success rates across all three benchmarks with both executors. On ALFWorld seen tasks, its success rates reach 76.86% and 81.43% with Qwen3-4B and Qwen3-8B, compared with 64.76% and 66.19% under warm-up only, and 66.07% and 72.85% under random probing.

The MemSyco-Bench results further connect these gains to memory-use decisions (Table 3). Value initialization increases Decision accuracy from 43.1%/44.5% to 49.6%/48.8%, while the full model reaches 66.5%/68.8%. Relative to random probing, EVSI-guided acquisition improves Decision accuracy by 10.6/13.9 percentage points and reduces sycophancy by 9.2/14.8 points. Removing either learning objective also reduces Decision accuracy under both executors, supporting the complementary roles of value estimation and outcome-weighted retrieval-path learning.

Table 3: MemSyco-Bench ablations at $K = 5$, averaged over five memory stores. Each entry reports Qwen3-4B/Qwen3-8B.

Metric	Native	Base	Warm-up	Random	No policy	No value	UpliftMem
Decision↑	41.7/44.4	43.1/44.5	49.6/48.8	55.9/54.9	56.7/56.1	55.8/53.1	**66.5/68.8**
Syco.↓	49.0/40.0	47.2/36.7	44.2/34.7	38.7/31.3	39.2/31.9	41.6/35.3	**29.5/16.5**
Select↑	54.3/61.1	55.1/62.2	53.7/57.9	58.3/60.3	56.9/60.2	58.0/62.8	**61.6/70.7**
Old↓	47.0/40.8	46.5/39.6	47.6/42.4	42.5/41.3	43.8/40.3	43.6/38.9	**39.3/31.7**

5.5 RQ4: Probe Strategy Efficiency

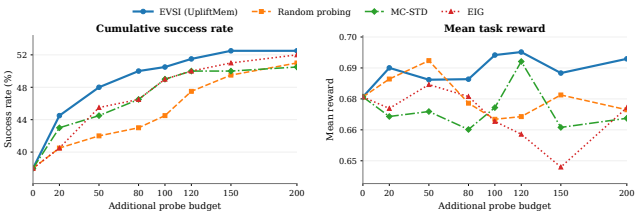

Figure 4. Cumulative success rate and mean reward across probe budgets on WebShop.

Starting from the same warm-up checkpoint and a fixed memory store, we compare EVSI with random probing, Monte Carlo predictive standard deviation (MC-STD), and expected information gain (EIG) on WebShop. Each strategy collects additional probes and updates the retriever through LoRA at budgets of $\{20, 50, 80, 100, 120, 150, 200\}$, followed by evaluation on the same 200 validation tasks. We measure cumulative success rate as the fraction of tasks solved by at least one checkpoint up to the current budget, alongside the mean reward of the current checkpoint.

Figure 4 shows that EVSI achieves the highest cumulative success rate at every nonzero budget, reaching 48.0% with 50 probes compared with 42.0% for random probing. Although the coverage gap narrows as more feedback is acquired, EVSI retains the highest coverage at 200 probes and achieves a mean reward of 0.694, compared with 0.670 for random probing. These results demonstrate that decision-aware acquisition converts limited execution feedback into broader task coverage and improved downstream performance. Additional probe-budget results for ALFWorld and BigCodeBench are presented in Appendix F.

5.6 RQ5: Sensitivity to Retrieval Budget

The preceding experiment varies the training-time probe budget. We now turn to the inference-time retrieval budget K, which controls how many memories may enter the task context. We vary $K \in \{3, 5, 10\}$ on MemSyco-Bench to examine how retrieval performance changes with available context. Figure 5 reports the absolute metrics, and Table 10 gives scenario-level gains over native retrieval.

The cross-model average gain remains positive at all three budgets, reaching 14.2, 15.8, and 15.1 percentage points at $K = 3, 5, 10$, respectively. The scenario breakdown also varies with K, with Evidence Conflict gains increasing as Scope Control gains decrease. UpliftMem therefore maintains its retrieval advantage across context capacities while adapting which memory-use errors are most strongly corrected.

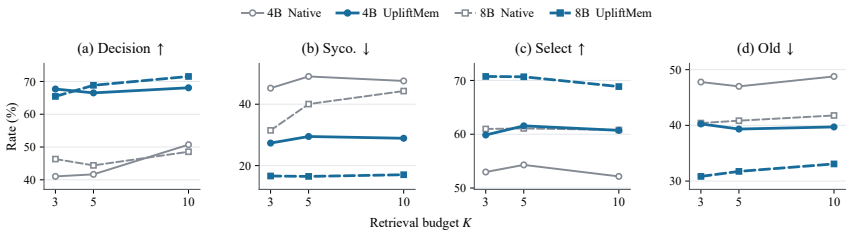

Figure 5: Effect of the maximum retrieval budget K on MemSyco-Bench, averaged across five memory stores. Higher Decision and Select and lower Syco. and Old indicate better performance.

6 Conclusion

We studied memory retrieval for interactive agents as a learning problem in which retrieval determines both what experience is used and which execution outcomes become available as supervision. UpliftMem addresses this coupling by learning set-level execution uplift relative to no-memory behavior and allocating additional rollouts according to their expected value for improving retrieval decisions. Across ALFWorld, WebShop, and BigCodeBench, UpliftMem achieves the best or joint-best success rates on the main evaluation sets. With fixed memory stores, it increases memory-use decision accuracy from 41.7% to 66.5% with Qwen3-4B and from 44.4% to 68.8% with Qwen3-8B; under matched probe budgets, EVSI-guided acquisition improves this accuracy over random probing by 10.6 and 13.9 percentage points. These results establish a broader view of agent memory retrieval: learning what experience to use also requires deciding which missing outcomes are worth observing.

AI USE STATEMENT

Generative AI was used to assist with manuscript organization, language editing and literature searches during revision.

REFERENCES

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember Me, Refine Me: A Dynamic Procedural Memory Framework for Experience-Driven Agent Evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.829. URL https://aclanthology.org/2026.findings-acl.829/.

Kathryn Chaloner and Isabella Verdinelli. Bayesian Experimental Design: A Review. *Statistical Science*, 10(3):273–304, 1995. doi: 10.1214/ss/1177009939.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025. URL https://arxiv.org/abs/2504.19413.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and efficient memory-augmented generation. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/a05b72653ec5b473732129829ae04195-Abstract-Conference.html.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring Agent Procedural Memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502. Association for Computational Linguistics, 2026b. doi: 10.18653/v1/2026.findings-acl.866. URL https://aclanthology.org/2026.findings-acl.866/.

Peter Frazier, Warren B. Powell, and Savas Dayanik. The Knowledge-Gradient Policy for Correlated Normal Beliefs. *INFORMS Journal on Computing*, 21(4):599–613, 2009. doi: 10.1287/ijoc.1080.0314. URL https://pubsonline.informs.org/doi/10.1287/ijoc.1080.0314.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning, 2011. URL https://arxiv.org/abs/1112.5745.

Ronald A. Howard. Information Value Theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26, 1966. doi: 10.1109/TSSC.1966.300074.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *arXiv preprint arXiv:2407.01502*, 2024.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. BatchBALD: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1906.08158.

Chunyu Li, Mengyuan Zhang, Jingyi Kang, Ding Chen, Jiajun Shen, Bo Tang, Xuanhe Zhou, Feiyu Xiong, and Zhiyu Li. MemReranker: Reasoning-Aware Reranking for Agent Memory Retrieval. *arXiv preprint arXiv:2605.06132*, 2026a. URL https://arxiv.org/abs/2605.06132.

Qinfeng Li, Dalin He, Yuntai Bao, Ying Yang, Ruoxi Chen, Xinyan Yu, Lizhou Liang, Ge Su, Wenqi Zhang, and Xuhong Zhang. SkillAligner: Treating retrieved skills as adaptable drafts at execution time, 2026b. URL https://arxiv.org/abs/2608.06880.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Changle Qu, Sunhao Dai, Hengyi Cai, Yiyang Cheng, Jun Xu, Shuaiqiang Wang, and Dawei Yin. Uplift-RAG: Uplift-driven knowledge preference alignment for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 9632–9644, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.511. URL https://aclanthology.org/2025.findings-emnlp.511/.

Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001. URL https://www.lri.fr/~sebag/Examens/Active/roy01toward.pdf.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive Architectures for Language Agents. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=1i6ZCvflQJ.

Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 814–823. PMLR, 2015. URL https://proceedings.mlr.press/v37/swaminathan15.html.

Songjun Tu, Chengdong Xu, Qichao Zhang, Yaocheng Zhang, Xiangyuan Lan, Linjing Li, Dong Li, and Dongbin Zhao. Dynamic Dual-Granularity Skill Bank for Agentic RL. *arXiv preprint arXiv:2603.28716*, 2026. URL https://arxiv.org/abs/2603.28716.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research*, 2024.

Zhishang Xiang, Zerui Chen, Yunbo Tang, Zhimin Wei, Ruqin Ning, Yujie Lin, Qinggang Zhang, and Jinsong Su. MemSyco-Bench: Benchmarking Sycophancy in Agent Memory. *arXiv preprint arXiv:2607.01071*, 2026. URL https://arxiv.org/abs/2607.01071.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic Memory for LLM Agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-0593. URL https://openreview.net/forum?id=FiM0M8gcct.

Yilong Xu, Jinhua Gao, Xiaoming Yu, Yuanhai Xue, Baolong Bi, Huawei Shen, and Xueqi Cheng. Training a Utility-based Retriever Through Shared Context Attribution for Retrieval-Augmented Language Models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 629–648. Association for Computational Linguistics, 2025b. doi: 10.18653/v1/2025.emnlp-main.33. URL https://aclanthology.org/2025.emnlp-main.33/.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing Large Language Model Agents to Manage and Utilize Memories via Reinforcement Learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022. URL https://arxiv.org/abs/2207.01206.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. MemRL: Self-Evolving Agents via Runtime Reinforcement Learning on Episodic Memory. *arXiv preprint arXiv:2601.03192*, 2026. URL https://arxiv.org/abs/2601.03192.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176*, 2025. URL https://arxiv.org/abs/2506.05176.

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. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning LLM agents without fine-tuning LLMs, 2025. URL https://arxiv.org/abs/2508.16153.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro von Werra. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=YrycTjllL0.

A Theoretical Analysis

We extend the coverage analysis in Section 4.2 and derive the acquisition scores in Section 4.3. The resulting Gaussian expressions are used by the batch procedure in Appendix B.2.

A.1 Selective Retrieval and Observation Coverage

The analysis fixes a task c, the recalled candidate pool, the feasibility constraints, and the training policy in Eq. 6. A memory set is identified by its members, independently of the order in which they were added. Let $\mathcal{P}(B)$ contain every feasible sampled construction path that terminates at B. A sampled `STOP` belongs to the path, whereas reaching a retrieval limit ends it without another sampled action. Distinct complete paths are mutually exclusive, even when they lead to the same set.

For a path $\mathbf{a} = (a_1, \dots, a_L)$, write S_{t-1} for the set preceding action a_t. Its nonnegative step deficit δ_t is the highest feasible conditional gain at S_{t-1} minus the gain of a_t. The length $L = L(\mathbf{a})$ counts sampled actions, including a sampled `STOP`, and the cumulative deficit is $d(\mathbf{a}) = \sum_{t=1}^{L} \delta_t$ as in Eq. 7.

Proof of Proposition 1. At any sampled decision, the denominator of Eq. 6 contains a maximizing action, giving

$$\pi_\theta(a_t \mid c, S_{t-1}) \le \exp\left(-\frac{\delta_t}{\tau_{\mathrm{tr}}}\right). \tag{18}$$

Writing $p(\mathbf{a} \mid c)$ for a complete path's probability, multiplication of its sampled conditional action probabilities gives

$$p(\mathbf{a} \mid c) = \prod_{t=1}^{L} \pi_\theta(a_t \mid c, S_{t-1}) \le \exp\left(-\frac{d(\mathbf{a})}{\tau_{\mathrm{tr}}}\right). \tag{19}$$

Summing the probabilities of all disjoint complete paths leading to B yields

$$p_B = \sum_{\mathbf{a} \in \mathcal{P}(B)} p(\mathbf{a} \mid c) \le \sum_{\mathbf{a} \in \mathcal{P}(B)} \exp\left(-\frac{d(\mathbf{a})}{\tau_{\mathrm{tr}}}\right), \tag{20}$$

which is Eq. 8. If B is unreachable under the candidate pool and feasibility constraints, $\mathcal{P}(B) = \emptyset$ and $p_B = 0$. □

The exact expression retains competition among actions. At a sampled state along path $\mathbf{a}$, define

$$Z_t(\mathbf{a}) = \sum_{a \in \mathcal{A}(c, S_{t-1})} \exp\left(\frac{g_\theta(a \mid c, S_{t-1}) - \max_{a' \in \mathcal{A}(c, S_{t-1})} g_\theta(a' \mid c, S_{t-1})}{\tau_{\mathrm{tr}}}\right). \tag{21}$$

At each sampled decision, $1 \le Z_t \le |\mathcal{A}(c, S_{t-1})|$, and

$$p_B = \sum_{\mathbf{a} \in \mathcal{P}(B)} \exp\left(-\frac{d(\mathbf{a})}{\tau_{\mathrm{tr}}}\right) \prod_{t=1}^{L(\mathbf{a})} Z_t(\mathbf{a})^{-1}. \tag{22}$$

The probability of reaching a set reflects action deficits and competition along every construction order. The sum over all feasible paths therefore determines direct coverage, rather than the probability of any one ordering.

Corollary 3 (Arrival of direct set evidence)**.** *Let* $\bar{p}_B = \min\{1, \sum_{\mathbf{a} \in \mathcal{P}(B)} \exp[-d(\mathbf{a})/\tau_{\mathrm{tr}}]\}$. *For* n *independent runs under the fixed policy, let* $N_B(n)$ *count terminal executions of* B *and* T_B *denote the run index of its first execution. Then*

$$\mathbb{E}[N_B(n)] = np_B \le n\bar{p}_B, \qquad \Pr(N_B(n) = 0) = (1 - p_B)^n \ge \max\{0, 1 - n\bar{p}_B\}. \tag{23}$$

If $p_B > 0$*, then* $\mathbb{E}[T_B] = p_B^{-1} \ge \bar{p}_B^{-1}$*. If* $p_B = 0$*,* T_B *is infinite almost surely.*

Proof. Independent runs give $N_B(n) \sim \mathrm{Binomial}(n, p_B)$ and a geometric first-arrival time when $p_B > 0$. The zero-count bound follows from $(1 - p_B)^n \ge 1 - np_B$ and $p_B \le \bar{p}_B$. The zero-coverage case follows directly. □

Complementarity illustrates the effect of construction-path probabilities on coverage. Suppose two candidate memories satisfy $G_\theta(c, \{m_1\}) = G_\theta(c, \{m_2\}) = -a$ with $a > 0$ and $G_\theta(c, \{m_1, m_2\}) = b > 0$, while STOP has gain zero. Each of the two addition orders begins with a deficit of at least a, even though the completed set has positive predicted value. Equation 8 gives $p_{\{m_1,m_2\}} \le \min\{1, 2e^{-a/\tau_{\mathrm{tr}}}\}$. Both construction orders are included in this bound, connecting set-level complementarity to the feedback coverage discussed in Section 4.2.

A.2 Identification Under Incomplete Coverage

The coverage result becomes an identification constraint when observations of other sets provide no structural information about the unexecuted alternative. The next proposition isolates this setting to distinguish direct evidence from generalization through a shared predictor.

Proposition 4 (Unobserved alternatives remain observationally indistinguishable)**.** *Fix a task c and a nonempty feasible set B. Consider a class of outcome laws in which the utility distribution of B can vary independently of the distributions for the other sets. Under a fixed retrieval policy that executes B with probability p_B, there exist two environments with opposite signs of $\Delta(c, B)$ such that any sign estimator $\widehat{z}_n$ based on n independent ordinary retrieval executions obeys the following bound. Index the environments by $v \in \{-, +\}$, with $\Pr_v$ their probability laws and Δ_v their corresponding uplifts.*

$$\max_{v\in\{-,+\}} \Pr_v(\widehat{z}_n \neq \operatorname{sign}(\Delta_v(c, B))) \ge \frac{1}{2}(1 - p_B)^n. \tag{24}$$

The bound also holds when no-memory outcomes accompany each execution. If $p_B = 0$, the ordinary-data laws are identical for all n and the sign is unidentifiable over this outcome class.

Proof. Construct two environments with deterministic utility $1/2$ for the empty set and every set other than B. Give B utility $3/4$ in environment $+$ and $1/4$ in environment $-$. The two environments have respective uplifts $1/4$ and $-1/4$ at B, while all other outcome laws agree. Let E_n be the event that ordinary retrieval never executes B. Its probability is $(1 - p_B)^n$ under either environment. Conditional on E_n, all observed feedback, including any no-memory references, has the same distribution in both environments. Because the true signs differ, the two conditional estimation-error probabilities sum to at least one. Their average unconditional error is therefore at least $\frac{1}{2}\Pr(E_n)$, and their maximum is no smaller. When $p_B = 0$, E_n occurs surely. □

Combining this result with Proposition 1 gives a worst-case sign-error bound of at least $\frac{1}{2}(1 - \bar{p}_B)^n$ over the stated outcome class. Shared representations introduce additional structure across sets. The surrogate in Appendix A.4 uses such structure to guide acquisition, while new executions supply direct labels.

A.3 Changing Policies and Additional Probe Coverage

Online scorer updates change retrieval probabilities. A history-dependent version of the coverage argument connects the fixed-policy results to this training setting.

Let $\mathcal{H}_{t-1}$ be the history before round t and $p_t(B \mid \mathcal{H}_{t-1})$ the conditional probability of executing B. Suppose a deterministic sequence $\epsilon_t \in [0, 1]$ satisfies $p_t(B \mid \mathcal{H}_{t-1}) \le \epsilon_t$ for every attainable history. Conditional expectation and the union bound give

$$\mathbb{E}[N_B(n)] \le \sum_{t=1}^{n} \epsilon_t, \qquad \Pr(N_B(n) = 0) \ge \max\left\{0, 1 - \sum_{t=1}^{n} \epsilon_t\right\}. \tag{25}$$

For the two environments in Proposition 4, couple the policy randomness and all observed outcomes until the first execution of B. Their histories agree before that event, giving a worst-case sign error of at least $\frac{1}{2}\max\{0, 1 - \sum_{t=1}^{n} \epsilon_t\}$. A set excluded throughout training has zero ordinary coverage.

A separately selected probe batch adds an opportunity to observe B. If a fixed ordinary policy executes B with probability p_B on each of n independent runs, and an independently selected probe batch includes B with probability r_B, the probability of at least one direct observation is

$$1 - (1 - p_B)^n(1 - r_B). \tag{26}$$

Equation 26 isolates the coverage added by an independent probe batch. The acquisition procedure in Section 4.3 instead conditions on available training data, as formalized by the following surrogate.

A.4 Gaussian Beliefs for Prospective Execution Evidence

This subsection derives the resolvable variance in Eq. 10. All quantities are conditional on the data $\mathcal{D}$ available before selecting a probe batch. The local Gaussian surrogate represents uncertainty about expected memory uplift, while execution variability enters through a separate observation-noise term. For a prospective batch Q of $n_Q = |Q|$ probes, let $y_Q \in \mathbb{R}^{n_Q}$ be the label vector, $\mu_Q \in \mathbb{R}^{n_Q}$ collect the predictions $G_\theta(c_q, B_q)$, and $\Phi_Q \in \mathbb{R}^{n_Q \times d}$ have rows $\varphi(c_q, B_q)^\top$. The residual execution noise ε_Q has covariance $\Omega_Q \in \mathbb{R}^{n_Q \times n_Q}$. The joint measurement model is

$$y_Q = \mu_Q + \Phi_Q \xi + \varepsilon_Q, \qquad \xi \sim \mathcal{N}(0, \Sigma), \qquad \varepsilon_Q \sim \mathcal{N}(0, \Omega_Q). \tag{27}$$

The latent vector ξ and execution noise ε_Q are independent conditional on $\mathcal{D}$. The conventions $G_\theta(c, \emptyset) = 0$ and $\varphi(c, \emptyset) = \mathbf{0}$ fix the empty-set uplift at zero. The feature basis and covariance refer to the same fixed scorer state.

For the comparisons in Section 4.3, write $g_j = A_j[F] - b_j$, where A_j is a linear functional of finitely many set values and b_j is a fixed reference in utility units. A two-set comparison has $A_j[F] = F(c, B_j^1) - F(c, B_j^0)$ and $b_j = 0$. A threshold comparison is included when its threshold is the utility of a genuine reference action. Let $\mu(c, B) = G_\theta(c, B)$ be the mean function and $h_j = A_j[\varphi] \in \mathbb{R}^d$ the feature contrast obtained by applying A_j to each coordinate. Then $\mu_j = A_j[\mu] - b_j$ and $g_j = \mu_j + h_j^\top \xi$, with $h_j = \varphi(c, B_j^1) - \varphi(c, B_j^0)$ for two sets. Let $\sigma_j \geq 0$ be the prior standard deviation of the margin, $k_j \in \mathbb{R}^{n_Q}$ the observation–margin covariance vector, and $W_Q \in \mathbb{R}^{n_Q \times n_Q}$ the observation covariance. Conditional on $\mathcal{D}$,

$$\begin{aligned} \sigma_j^2 &= h_j^\top \Sigma h_j, \qquad k_j = \mathrm{Cov}(y_Q, g_j) = \Phi_Q \Sigma h_j, \\ W_Q &= \mathrm{Var}(y_Q) = \Phi_Q \Sigma \Phi_Q^\top + \Omega_Q. \end{aligned} \tag{28}$$

Assuming W_Q is positive definite, Gaussian conditioning gives

$$\mathbb{E}[g_j \mid \mathcal{D}, y_Q] = \mu_j + k_j^\top W_Q^{-1}(y_Q - \mu_Q), \tag{29}$$

$$\mathrm{Var}(g_j \mid \mathcal{D}, y_Q) = \sigma_j^2 - k_j^\top W_Q^{-1} k_j. \tag{30}$$

Because the conditional variance is independent of the realized outcome, the distribution of the future posterior mean is determined before executing any candidate probe,

$$\mathbb{E}[g_j \mid \mathcal{D}, y_Q] \sim \mathcal{N}(\mu_j, s_j^2(Q)), \qquad s_j^2(Q) = k_j^\top W_Q^{-1} k_j. \tag{31}$$

Equation 31 supplies the resolved variance in Eq. 10 and the pre-observation distribution required by Theorem 2.

A shared no-memory execution creates correlated observation noise. If assisted-run noises η_i are mutually independent with variances ν_i^2 and an independent baseline noise η_0 has variance ν_0^2, then $\varepsilon_i = \eta_i - \eta_0$ implies

$$\Omega_Q = \mathrm{diag}(\nu_1^2, \ldots, \nu_{n_Q}^2) + \nu_0^2 \mathbf{1}\mathbf{1}^\top. \tag{32}$$

Here $\mathbf{1} \in \mathbb{R}^{n_Q}$ is the all-ones vector. For several independent baseline groups, let Z be the binary membership matrix with one row per probe and one column per reference. Let D_0 and D_{assist} be diagonal matrices of reference-noise and assisted-run variances, respectively. Then $\Omega_Q = D_{\mathrm{assist}} + Z D_0 Z^\top$. For an already observed reference, Ω_Q denotes the noise covariance conditional on that observation. Covariances induced by coupled executions are included in the same matrix. Appendix D.3 specifies how shared references enter rollout accounting.

For a set fixed before paired execution, Eq. 4 has expectation $\Delta(c, S)$ whenever both trajectories have the prescribed marginals, including when their randomness is coupled. Labels retained after outcome-dependent candidate selection have a different sampling rule. Appendix B.3 states the rule used for writing-validation data.

For a fixed feature basis, let $\Sigma_Q = \mathrm{Var}(\xi \mid \mathcal{D}, y_Q)$ denote the posterior latent covariance. Gaussian conditioning gives

$$\Sigma_Q = \Sigma - \Sigma \Phi_Q^\top W_Q^{-1} \Phi_Q \Sigma, \qquad \mathbb{E}[\xi \mid \mathcal{D}, y_Q] = \Sigma \Phi_Q^\top W_Q^{-1}(y_Q - \mu_Q). \tag{33}$$

This hypothetical posterior scores measurements before execution. The batch procedure in Appendix B.2 holds the belief fixed during selection, and the training procedure in Appendix C uses realized labels to update the scorer. The posterior covariance is defined in its associated feature basis, so a change of basis also changes its matrix representation.

A.5 Decision Value of Gaussian Observations

To establish Theorem 2, consider two available decisions with uncertain utilities F^+ and F^- and margin $g = F^+ - F^-$. This includes a two-set comparison and a fixed-threshold decision when the threshold is the utility of its reference action. The best expected utility before observation is $\mathbb{E}[F^- \mid \mathcal{D}] + (\mathbb{E}[g \mid \mathcal{D}])_+$. After observing y_Q, it is $\mathbb{E}[F^- \mid \mathcal{D}, y_Q] + (\mathbb{E}[g \mid \mathcal{D}, y_Q])_+$. Taking the pre-observation expectation cancels the reference term by the tower property. Hence binary EVSI is $\mathbb{E}_{y_Q|\mathcal{D}}[(\mathbb{E}[g \mid \mathcal{D}, y_Q])_+] - (\mathbb{E}[g \mid \mathcal{D}])_+$ even when the reference alternative is uncertain.

Lemma 5 (Positive part of a Gaussian variable)**.** *Let X be Gaussian with mean μ and standard deviation $s > 0$. Then $\mathbb{E}[(X)_+] = s\phi(\mu/s) + \mu\Phi(\mu/s)$.*

Proof. Substituting $z = (x - \mu)/s$ in the defining integral yields

$$\mathbb{E}[(X)_+] = \int_{-\mu/s}^{\infty} (\mu + sz)\phi(z)\, dz = \mu\Phi(\mu/s) + s\phi(\mu/s), \tag{34}$$

using $\phi'(z) = -z\phi(z)$ and Gaussian symmetry. □

Proof of Theorem 2. Equation 31 gives $\mu'_j \sim \mathcal{N}(\mu_j, s_j^2(Q))$. For $s_j = s_j(Q) > 0$, applying the lemma and subtracting $(\mu_j)_+$ gives

$$E_j(Q) = s_j\phi(\mu_j/s_j) + \mu_j\Phi(\mu_j/s_j) - (\mu_j)_+ . \tag{35}$$

For $\mu_j \geq 0$, this becomes $s_j\phi(\mu_j/s_j) - \mu_j\Phi(-\mu_j/s_j)$; Gaussian symmetry gives the same expression with $|\mu_j|$ when $\mu_j < 0$. This is Eq. 11. When $s_j = 0$, the future posterior mean equals the current mean almost surely and the EVSI is zero. □

Corollary 6 (Dependence on margin and resolvable uncertainty)**.** *Writing $a = |\mu_j|$ for the absolute decision margin and $s > 0$ for the resolved standard deviation, define $E(a, s) = s\phi(a/s) - a\Phi(-a/s)$. Then*

$$\frac{\partial E}{\partial s} = \phi(a/s) > 0, \qquad \frac{\partial E}{\partial a} = -\Phi(-a/s) < 0, \qquad 0 \leq E_j(Q) \leq \frac{s_j(Q)}{\sqrt{2\pi}} \leq \frac{\sigma_j}{\sqrt{2\pi}}. \tag{36}$$

Proof. Differentiation and $\phi'(t) = -t\phi(t)$ give the derivatives. The maximum at fixed s occurs at $a = 0$. Jensen's inequality for the convex positive-part function gives nonnegativity, and nonnegative conditional variance gives $s_j^2 \leq \sigma_j^2$. □

The decision meaning of EVSI follows from opportunity loss, the utility forgone relative to choosing with full knowledge of the margin. Before observation this loss is $\mathbb{E}[(g_j)_+ \mid \mathcal{D}] - (\mu_j)_+$. After observing Q, the corresponding expected loss is $\mathbb{E}[(g_j)_+ \mid \mathcal{D}] - \mathbb{E}_{y_Q|\mathcal{D}}[(\mu'_j)_+]$, by the tower property. Their difference equals $E_j(Q)$, so EVSI measures the expected reduction in decision opportunity loss. This identity explains the local acquisition objective in Section 4.3. Appendix A.7 connects it to the subsequent use of observations for neural-model training.

A.6 Finite-Family Decisions and Information Quantity

The aggregate $H(Q)$ in Eq. 12 values a family of local comparisons. We relate this quantity to choosing one set from a finite family and then distinguish decision value from information quantity.

Let $\mathcal{B} = \{B_0, \ldots, B_{J-1}\}$ be a finite family of J executable sets for a fixed task, with $F_k = F(c, B_k)$ and $\mu_k = \mathbb{E}[F_k \mid \mathcal{D}]$. Relabel a predicted-best member as B_0, so $\mu_0 = \max_{0 \leq k < J} \mu_k$. All extrema and sums over k in this subsection range over these J alternatives unless otherwise specified. The EVSI for selecting a single set from this family is

$$E_{\mathcal{B}}(Q) = \mathbb{E}_{y_Q|\mathcal{D}}\left[\max_k \mathbb{E}[F_k \mid \mathcal{D}, y_Q]\right] - \max_k \mathbb{E}[F_k \mid \mathcal{D}]. \tag{37}$$

This is the finite-alternative value used in knowledge-gradient reasoning (Frazier et al., 2009).

Proposition 7 (Binary comparisons bound finite-family EVSI)**.** *For $J \geq 2$, let $g_k = F_k - F_0$ for $k \neq 0$ and $E_k(Q)$ be the binary EVSI for the corresponding comparison. With integrable utilities and a coherent common posterior,*

$$\max_{k\neq 0} E_k(Q) \leq E_{\mathcal{B}}(Q) \leq \sum_{k\neq 0} E_k(Q). \tag{38}$$

The bounds allow correlated alternatives, and joint Gaussianity makes each binary term available in closed form.

Proof. Write $g'_k = \mathbb{E}[F_k - F_0 \mid \mathcal{D}, y_Q]$. Because $\mu_k - \mu_0 \leq 0$, $E_k(Q) = \mathbb{E}[(g'_k)_+]$. Decomposing the posterior maximum relative to F_0 and applying the tower property yields

$$E_{\mathcal{B}}(Q) = \mathbb{E}\big[\max\{0, g'_1, \ldots, g'_{J-1}\}\big]. \tag{39}$$

For every outcome, $(g'_k)_+ \leq \max\{0, g'_1, \ldots, g'_{J-1}\} \leq \sum_{k\neq 0} (g'_k)_+$. Taking expectations gives the bounds. □

When $\mathcal{G}$ consists of one comparison of every alternative in a finite family against the same predicted-best incumbent, with no duplicate or additional comparisons, its unweighted sum $H(Q)$ upper-bounds the finite-family EVSI by Eq. 38. For comparisons against different incumbents, $H(Q)$ instead values the collection of local decisions. Duplicate comparisons can count the same decision improvement more than once. For example, for an integrable zero-mean random value Z, if $F_1 = F_2 = Z$ and $F_0 = 0$, a probe revealing Z gives $E_1 = E_2 = \mathbb{E}[(Z)_+]$ but $E_{\mathcal{B}} = \mathbb{E}[(Z)_+]$.

For the scalar Gaussian observation y_q from probe q, with predictive variance $v_q = \mathrm{Var}(y_q \mid \mathcal{D}) > 0$, define $\beta_k = \mathrm{Cov}(F_k, y_q \mid \mathcal{D})/\sqrt{v_q}$ and $Z = (y_q - \mathbb{E}[y_q \mid \mathcal{D}])/\sqrt{v_q} \sim \mathcal{N}(0,1)$. The posterior means $\mu'_k = \mathbb{E}[F_k \mid \mathcal{D}, y_q]$ satisfy $\mu'_k = \mu_k + \beta_k Z$, giving

$$E_{\mathcal{B}}(q) = \mathbb{E}_Z[\max_k(\mu_k + \beta_k Z)] - \max_k \mu_k. \tag{40}$$

This expression evaluates a single final selection through an upper envelope or one-dimensional integration. By contrast, Eq. 12 aggregates the local comparisons constructed in Appendix B.2.

This decision-value perspective also clarifies the relationship to information gain. Let $I(g_j; y_Q \mid \mathcal{D})$ denote conditional mutual information, using natural logarithms. For a Gaussian margin with $0 < \sigma_j^2 - s_j^2(Q) \leq \sigma_j^2$,

$$I(g_j; y_Q \mid \mathcal{D}) = \frac{1}{2}\log\frac{\sigma_j^2}{\sigma_j^2 - s_j^2(Q)}, \qquad s_j^2(Q) = \sigma_j^2\big(1 - e^{-2I(g_j; y_Q \mid \mathcal{D})}\big). \tag{41}$$

The identity follows by subtracting the conditional Gaussian entropy from the prior entropy. For a fixed comparison, μ_j and σ_j are fixed, so mutual information and binary EVSI rank probes in the same order. Across different comparisons, EVSI additionally accounts for the decision margin and utility scale. If a continuous margin is revealed exactly, mutual information diverges while binary EVSI approaches the finite expected value of perfect information.

A.7 Batch Acquisition and Evidence Timing

The batch score in Section 4.3 accounts for overlapping evidence before outcomes are observed. Fix $\mathcal{D}$, the surrogate, the comparison family $\mathcal{G}$, and the feasible probe pool throughout the following calculation. All expectations and covariances below are conditional on this pre-batch evidence. For a coherent model and $Q \subseteq Q'$, the tower property and conditional Jensen inequality give

$$\mathbb{E}\Big[(\mathbb{E}[g_j \mid \mathcal{D}, y_{Q'}])_+\Big] \geq \mathbb{E}\Big[(\mathbb{E}[g_j \mid \mathcal{D}, y_Q])_+\Big]. \tag{42}$$

Thus $E_j(Q)$ and $H(Q)$ are monotone with additional observations. Their marginal gains need not decrease. For example, let X, Y be independent standard Gaussian variables, set $g = X$, and consider noiseless probes $y_1 = X + Y$ and $y_2 = Y$. The second probe alone has no information about g, but after the first it reveals g exactly, so its marginal decision value increases.

For a prospective scalar observation y_q, define the residual covariance and variance under the fixed pre-batch model as

$$\kappa_{jq|Q} = \mathrm{Cov}(g_j, y_q) - \mathrm{Cov}(g_j, y_Q) W_Q^{-1} \mathrm{Cov}(y_Q, y_q), \tag{43}$$

$$v_{q|Q} = \mathrm{Var}(y_q) - \mathrm{Cov}(y_q, y_Q) W_Q^{-1} \mathrm{Cov}(y_Q, y_q). \tag{44}$$

When $v_{q|Q} > 0$, block Gaussian conditioning gives

$$s_j^2(Q \cup \{q\}) = s_j^2(Q) + \frac{\kappa_{jq|Q}^2}{v_{q|Q}}. \tag{45}$$

For $Q = \emptyset$, the covariance terms involving y_Q are absent and $s_j^2(\emptyset) = 0$. For nonempty Q, the formula first conditions on y_Q and then on the residual of y_q after its linear prediction from y_Q. For a fixed scalar mean μ, let $f_\mu(s) = s\psi(|\mu|/s)$ for $s > 0$ and $f_\mu(0) = 0$. Then

$$V_t(q) = \sum_{j \in \mathcal{G}} \left[f_{\mu_j}\left(\sqrt{s_j^2(Q_t) + \frac{\kappa_{jq|Q_t}^2}{v_{q|Q_t}}} \right) - f_{\mu_j}(s_j(Q_t)) \right]. \tag{46}$$

Each term uses the fixed pre-batch mean, so the expression is the difference between two prospective batch values. It accounts for scheduled measurements through their covariance without conditioning on unknown labels. For an exactly redundant observation with $v_{q|Q_t} = 0$, its incremental contribution is defined as zero. Linear solves using a Cholesky factor of W_Q avoid forming its inverse explicitly.

For a fixed batch state, let $\mathcal{Q}_t = \mathcal{Q} \setminus Q_t$ be the remaining feasible probes. The softmax in Eq. 49 maximizes

$$\sum_{q \in \mathcal{Q}_t} p(q) V_t(q) + \tau_{\mathrm{evsi}} \mathcal{H}(p), \qquad \mathcal{H}(p) = -\sum_{q \in \mathcal{Q}_t} p(q) \log p(q),$$

over probability distributions p on $\mathcal{Q}_t$, where $\mathcal{H}(p)$ is Shannon entropy with $0 \log 0 = 0$ and $\tau_{\mathrm{evsi}} > 0$ is its weight. A Lagrange multiplier for normalization yields the sampling rule. This objective concerns the next-probe distribution at the current batch state.

A training probe supplies a label for a later scorer update, whereas Eq. 37 values a decision made from the observation's posterior. To distinguish these roles, let $\mathcal{T}(\theta, \mathcal{D}, y_Q)$ denote the parameter update, let $S_\theta(c')$ be deterministic retrieval on a future task, and let p_{future} be the future-task distribution. The expected change in future execution uplift induced by this update is

$$\mathbb{E}_{y_Q|\mathcal{D}} \mathbb{E}_{c' \sim p_{\mathrm{future}}} \left[\Delta\big(c', S_{\mathcal{T}(\theta, \mathcal{D}, y_Q)}(c')\big) - \Delta\big(c', S_\theta(c')\big) \right]. \tag{47}$$

The implemented $H(Q)$ is computable before execution from current local comparisons. Algorithm 1 uses the acquired outcomes to train the shared scorer. Section 5.4 examines probing within the full system, while Section 5.5 compares acquisition strategies across additional rollout budgets.

B Algorithmic Details

Algorithm 1 gives training and test-time execution. The seed trajectories are $\mathcal{T}_0$, and $\{c_t\}_{t=1}^n$ is the task stream across all training epochs. The buffers $\mathcal{D}_v$ and $\mathcal{D}_\pi$ store value observations and recent sampled paths, respectively, while $\mathcal{W}$ stores the current acquisition window. The operator $\uplus$ appends records without removing repetitions, and observations retain their reference identities for pairwise supervision. Observe returns the memory-assisted trajectory, its no-memory reference, and their uplift from Eq. 4, using the reference-reuse protocol in Appendix D.3. WriteMemory implements Appendix B.3, updating the memory pool and returning the selected validation record in $\mathcal{D}_w$ for addition to $\mathcal{D}_v$, regardless of admission.

Bootstrap constructs the initial pool and uplift observations from the seed trajectories. SamplePath follows Eq. 6 and returns its terminal set and recorded path, including the states and feasible-action masks needed for policy training. LocalComparisons constructs the probe pool $\mathcal{Q}$ and comparison family $\mathcal{G}$ in Appendix B.2. The score H in Eq. 12 uses the current scorer and evidence at batch entry, with this belief fixed throughout selection. Optimize applies the configured optimizer to the stated objective using eligible observations, with $\mathcal{L}_{\mathrm{policy}}(\emptyset) = 0$. Clearing $\mathcal{D}_\pi$ after an update prevents reuse of paths sampled under earlier scorer states. BeamRetrieve implements Appendix B.1 with width two and includes the empty set among its candidates.

Algorithm 1 UPLIFTMEM training and inference

Require: Frozen executor $\mathcal{E}$, retrieval budget K, acquisition interval h, probe budget B_{p}, update indices $\mathcal{I}_{\mathrm{upd}}$

function TRAIN($\mathcal{T}_0, \{c_t\}_{t=1}^{n}, \theta$)
  $(\mathcal{M}, \mathcal{D}_v) \leftarrow$ BOOTSTRAP($\mathcal{T}_0, \mathcal{E}$)
  $\theta \leftarrow$ OPTIMIZE($\theta, \mathcal{L}_{\mathrm{value}}(\mathcal{D}_v)$)
  $\mathcal{D}_\pi \leftarrow \emptyset, \quad \mathcal{W} \leftarrow \emptyset$
  for $t = 1, \ldots, n$ **do**
    $C_t \leftarrow$ RECALL($c_t, \mathcal{M}$)
    $(S_t, \mathbf{a}_t) \leftarrow$ SAMPLEPATH(π_θ, c_t, C_t, K)
    $(\tau_t, \tau_{\emptyset,t}, y_t) \leftarrow$ OBSERVE($c_t, S_t, \mathcal{E}$)
    $\mathcal{D}_v \leftarrow \mathcal{D}_v \uplus \{(c_t, S_t, y_t)\}$
    $\mathcal{D}_\pi \leftarrow \mathcal{D}_\pi \uplus \{(c_t, \mathbf{a}_t, y_t)\}$
    $(\mathcal{M}, \mathcal{D}_w) \leftarrow$ WRITEMEMORY($c_t, \tau_t, \tau_{\emptyset,t}, \mathcal{M}$)
    $\mathcal{D}_v \leftarrow \mathcal{D}_v \uplus \mathcal{D}_w$
    $\mathcal{W} \leftarrow \mathcal{W} \uplus \{(c_t, C_t, S_t, \mathbf{a}_t)\}$
    if $t \bmod h = 0$ **then**
      $(\mathcal{Q}, \mathcal{G}) \leftarrow$ LOCALCOMPARISONS($\mathcal{W}$)
      $Q \leftarrow$ SELECTPROBES($\mathcal{Q}, \mathcal{G}, G_\theta, B_{\mathrm{p}}$)
      $\mathcal{D}_v \leftarrow \mathcal{D}_v \uplus$ OBSERVEPROBES($Q, \mathcal{E}$)
      $\mathcal{W} \leftarrow \emptyset$
    end if
    if $t \in \mathcal{I}_{\mathrm{upd}}$ **then**
      $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{value}}(\mathcal{D}_v) + \mathcal{L}_{\mathrm{policy}}(\mathcal{D}_\pi)$
      $\theta \leftarrow$ OPTIMIZE($\theta, \mathcal{L}$)
      $\mathcal{D}_\pi \leftarrow \emptyset$
    end if
  end for
  return $(G_\theta, \mathcal{M})$
end function

function EXECUTE($c, G_\theta, \mathcal{M}$)
  $C \leftarrow$ RECALL($c, \mathcal{M}$)
  $S \leftarrow$ BEAMRETRIEVE($G_\theta, c, C, K, 2$)
  return $\mathcal{E}(c, S)$
end function

B.1 MEMORY SET SEARCH AND STOPPING

The search in Section 4.2 operates on subsets $S \subseteq \mathcal{C}(c)$. Used memories and additions violating the count or token constraints are masked. A sampled `STOP` returns the current set, including $\emptyset$, and termination is forced when no feasible addition remains. Candidate states are deduplicated by their members and serialized consistently for scoring. The set-valued formulation treats each subset as a complete context under the memory-presentation protocol in Section 3.2.

For a sequence of ℓ memory additions $m_1, \ldots, m_\ell$, let $S_0 = \emptyset$ and $S_t = S_{t-1} \cup \{m_t\}$. The conditional gains in Eq. 5 telescope,

$$\sum_{t=1}^{\ell} g_\theta(m_t \mid c, S_{t-1}) = G_\theta(c, S_\ell) - G_\theta(c, \emptyset) = G_\theta(c, S_\ell). \tag{48}$$

This identity relates accumulated gains to the terminal set value. The path probabilities in Eq. 6 also contain state-dependent normalizers, so search ranks complete sets by G_θ rather than by path likelihood.

Training samples additions and stopping actions from Eq. 6. At test time, beam width is $W = 2$. An active beam and a cross-depth archive are initialized with $\emptyset$. At each depth, all feasible one-memory extensions are scored, up to two are retained per parent, and the two highest-valued distinct extensions form the next beam. Surviving sets enter the archive. Search continues to K or until no feasible extension remains and returns the highest-valued archived set using a deterministic tie rule. Retaining an intermediate set with negative immediate gain allows later additions to expose complementarity.

For $N = |\mathcal{C}(c)|$, beam width W, and memory-count cap K, this procedure requires $O(WNK)$ set-score evaluations, excluding recall and executor cost. Deduplication can reduce this count.

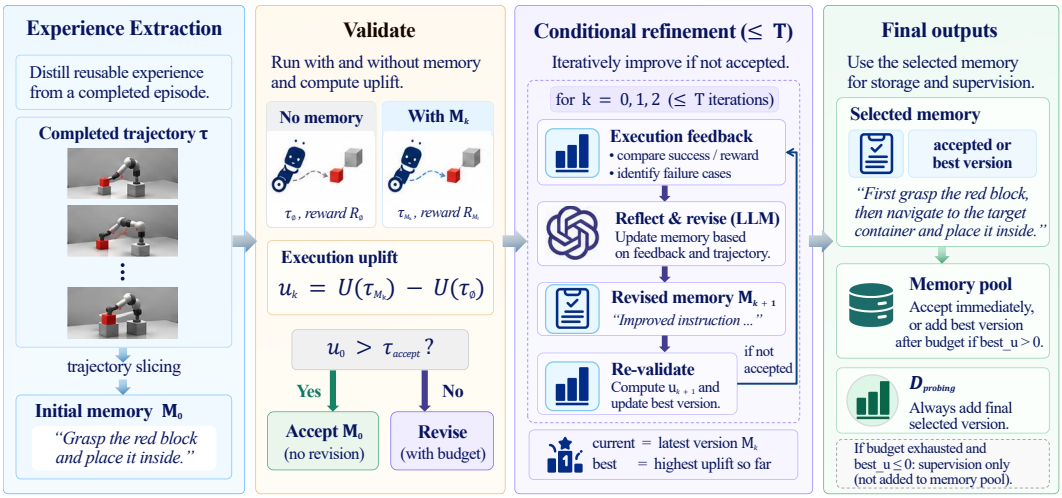

Figure 6: Execution-validated memory writing. Completed trajectories yield memory candidates that are validated on their source tasks and revised when needed. The selected version supplies value supervision, while validated uplift governs memory-pool admission.

B.2 Prospective Probe Batch Construction

For each task in the preceding acquisition window, retrieval supplies a selected set S and the recalled pool $\mathcal{C}(c)$. Feasible local candidates comprise the complete set S and one-memory edits of it, including removals $S \setminus \{m\}$, additions $S \cup \{m'\}$, and replacements $(S \setminus \{m\}) \cup \{m'\}$ for $m \in S$ and $m' \in \mathcal{C}(c) \setminus S$. The edit operator is also defined on retrieval prefixes, subject to the same feasibility constraints. All candidates satisfy the original retrieval constraints. Distinct sets are identified by their members, consistently serialized, and deduplicated. The feasible pool $\mathcal{Q}$ consists of task–set pairs $q = (c_q, B_q)$, each of which supplies one complete-set observation when executed.

The comparison family $\mathcal{G}$ consists of the local choices exposed by the candidate neighborhood. A comparison against a removal or addition assesses whether retaining or extending the current set changes expected utility. Comparing replacements of a shared base assesses which continuation is preferred. A stopping comparison uses the difference between a feasible extension and its current base, with zero as the reference margin. The functionals A_j and references b_j in Appendix A.4 encode these differences. Each distinct comparison is counted once with a consistent orientation, since reversing its two alternatives leaves binary EVSI unchanged. The comparison family and feasible probe pool are fixed for each acquisition batch.

For the fixed local surrogate, initialize the scheduled batch as $Q_0 = \emptyset$ and let $\mathcal{Q}_t = \mathcal{Q} \setminus Q_t$ contain its remaining feasible probes. For every $q \in \mathcal{Q}_t$, use Eq. 45 to calculate the resolved variances for $Q_t \cup \{q\}$, then use Eq. 46 to compute the full aggregate increment $V_t(q) = H(Q_t \cup \{q\}) - H(Q_t)$ from Eq. 12. With acquisition temperature $\tau_{\text{evsi}} > 0$, the next-probe distribution is

$$p_t(q) = \frac{\exp(V_t(q)/\tau_{\text{evsi}})}{\sum_{q' \in \mathcal{Q}_t} \exp(V_t(q')/\tau_{\text{evsi}})}, \qquad q \in \mathcal{Q}_t. \tag{49}$$

Algorithm 1 completes batch selection before executing its probes. The incremental covariance calculation in Eq. 45 accounts for evidence already scheduled under a fixed pre-batch belief. Between batches, the features and covariance are refreshed consistently with the scorer. Appendix D.3 accounts for selected executions and their no-memory references.

B.3 Iterative Memory Writing and Validation

The writing stage of Algorithm 1 uses completed episodes with either sign of retrieval uplift. An episode's label in Eq. 4 evaluates the retrieved context, so each newly distilled memory requires its own validation. Trajectory slicing distills an initial candidate M_0 using the observed actions and outcomes, including reusable procedures and corrective guidance (Zhao et al., 2024; Cao et al., 2026).

As illustrated in Figure 6, the frozen executor therefore validates each candidate M_k, the memory after k revisions, alone on its source task c. Against the cached no-memory reference, its observed uplift is

$$u_k = U(\tau_{\{M_k\}}) - U(\tau_\emptyset). \tag{50}$$

A candidate with $u_k > \tau_{\text{accept}} = 0$ is accepted immediately. Otherwise, the writer uses the realized trajectory, outcome, and failure diagnostics to revise the latest candidate into M_{k+1}. The highest-uplift version is tracked separately as revision proceeds, for at most $T = 3$ revisions. The selected candidate is the first version passing the threshold, or the highest-uplift version when the budget is exhausted.

The selected memory, source task, and observed uplift enter $\mathcal{D}_v$ whether or not admission occurs. The pool admits the selected version when its uplift exceeds τ_{accept}. Admission uses singleton validation on the source task, whereas G_θ evaluates later combinations. Appendix D.3 counts every executed validation attempt, and Appendix E.5 gives the revision interface.

C Training Details

The following implementation details instantiate the scorer and learning objectives in Sections 4.1 and 4.4.

C.1 Scorer Parameterization

The scorer represents the task as a query and the complete memory set as a document under the instruction in Appendix E.3. Let $z_\theta(c, S)$ be its yes-minus-no token-logit score. Affine calibration maps this score to utility units,

$$G_\theta(c, S) = \begin{cases} a\, z_\theta(c, S) + b, & S \neq \emptyset, \\ 0, & S = \emptyset, \end{cases} . \tag{51}$$

The scale $a \in \mathbb{R}$ and offset $b \in \mathbb{R}$ are optimized jointly with the low-rank adaptation (LoRA) parameters. The notation θ includes these calibration coefficients and the trainable adapter parameters. The empty-set convention anchors predictions to the same reference as $\Delta(c, \emptyset) = 0$.

C.2 Value Supervision

Regression in Eq. 16 weights uplift magnitude using

$$w_{\text{mag}}(y) = \max\{w_{\min}, \min(1, |y|/\tau_{\text{ref}})\}, \tag{52}$$

where $0 < w_{\min} \leq 1$ is the minimum weight and $\tau_{\text{ref}} > 0$ is the scale at which it reaches one. Sign supervision retains $|y| > \epsilon_{\text{sign}}$. Pairwise supervision retains sets from the same task and realized no-memory reference with $|y_a - y_b| > \epsilon_{\text{rank}}$, where $\epsilon_{\text{rank}} \geq 0$ is the minimum label-gap threshold. Preferences follow the larger observed uplift, including when both labels have the same sign.

C.3 Outcome-Weighted Retrieval-Path Learning

For an executed retrieval path in Eq. 17, let $y_i = y(c_i, S_{i,L_i})$ be the terminal set's observed uplift. The path length L_i includes a sampled terminating `STOP`. Its signed outcome weight is

$$\omega_i = \begin{cases} 0, & |y_i| \leq \epsilon_{\text{policy}}, \\ \min(y_i, A_{\max}), & y_i > \epsilon_{\text{policy}}, \\ \eta_{\text{neg}} \max(y_i, -A_{\max}), & y_i < -\epsilon_{\text{policy}}, \end{cases} \tag{53}$$

where $\epsilon_{\text{policy}} \geq 0$ excludes near-zero outcomes, $A_{\max} > 0$ clips their magnitude, and $\eta_{\text{neg}} \in [0, 1]$ scales negative feedback.

Policy updates use the latest batch of paths sampled under the current scorer, evaluating their likelihoods with the recorded set states and action masks. Paths from earlier batches are not replayed for this objective.

Table 4: Data partitions for the agent benchmarks. Seed and runtime counts denote execution samples or trajectories.

Benchmark	Seed	Runtime	Evaluation
ALFWorld	500	1,200	140 valid-seen / 134 valid-unseen
WebShop	500	2,300	500 official test goals
BigCodeBench	—	798	342 held-out manifest tasks

Table 5: Dialogue-grouped MemSyco-Bench partition with seed 42. Scenario names follow the benchmark.

Scenario	Total	Train	Test
Personalized Memory Use	300	210	90
Valid Memory Selection	350	245	105
Memory Evidence Conflict	300	210	90
Contextual Scope Control	300	210	90
Objective Fact Judgment	300	210	90
Total	1,550	1,085	465

D Experimental Protocols

This section specifies the protocols behind the complete-workflow and controlled-retrieval evaluations in Section 5.

D.1 Benchmarks and Data Partitions

Table 4 gives the partitions used for the agent benchmarks in Section 5.2. Seed and runtime counts refer to execution samples or trajectories that provide training experience and memory-construction data. Evaluation tasks and their execution feedback are excluded from these updates.

We partition MemSyco-Bench into 1,085 training and 465 test instances across five scenarios using a task-stratified 70:30 split with random seed 42. Instances sharing the same canonical dialogue are assigned to the same partition to prevent dialogue overlap between training and test sets (Table 5).

D.2 Memory Sources and Evaluation Protocols

Section 5.2 evaluates complete memory workflows. UpliftMem constructs its initial memory pool during bootstrap initialization and extends it through iterative memory writing, while comparison systems use their own construction and retrieval procedures. ReAct is the reference without external memory. These end-to-end results include the effects of both memory content and retrieval.

Section 5.3 isolates retrieval by comparing native and UpliftMem retrieval within the same post-extraction store, executor, and evaluator. MemSyco-Bench supplies a prior dialogue and query. Its five memory sources are NaiveRAG, Mem0, A-MEM, MemGPT, and LightMem, with NaiveRAG abbreviated RAG. LightMem follows the benchmark's integrated implementation.

BigCodeBench uses a task-specific retrieval view. Its stored records distinguish successful implementation procedures from diagnostic failure reflections, while similarity recall, valuation, probing, and generation use the successful implementation-evidence projection. Injected evidence excludes diagnostic-only reflections, source-task contracts, and source-specific literals. Appendices E.2 and E.4 give the injected view and stored record contracts.

D.3 Utility Configuration and Rollout Accounting

The utility in Eq. 1 uses a fixed performance measure $R(\tau)$, cost measure $C(\tau)$, cap $C_{\max}$, efficiency weight γ, and threshold ρ within each benchmark protocol. Cost counts actions in ALFWorld and WebShop and injected implementation-evidence tokens in BigCodeBench. The latter excludes

memory headings, scope metadata, the task prompt, and generated code. This trajectory-level cost is distinct from the number of executions used to collect training labels.

Training accounting includes no-memory reference executions, ordinary memory-assisted executions, acquired probes, and every executed writing-validation attempt. An execution reused for several purposes is counted once. Reference identities indicate which labels share baseline noise, as modeled in Appendix A.4. The retained writing label follows Appendix B.3, while cost accounting includes both retained and discarded validation attempts.

Probe-policy comparisons use a common candidate-pool and accounting convention. The number of newly acquired probes measures sample allocation, whereas total training cost also includes any required reference and validation executions. At evaluation, feedback used to score the returned trajectory does not trigger additional probe executions.

D.4 Models and Implementation

The frozen task executors are Qwen3-4B and Qwen3-8B (Yang et al., 2025). Each has an independently adapted scorer initialized from Qwen3-Reranker-0.6B (Zhang et al., 2025) and trained with LoRA (Hu et al., 2022). Table 6 collects the training, acquisition, and search settings.

The task counter continues across three training epochs, triggering acquisition every $h = 400$ processed tasks. Each window supplies at most $B_{\mathrm{p}} = 300$ probe executions in Algorithm 1.

Table 6: Scorer training, probe acquisition, and retrieval settings.

Setting	Value
LoRA rank / scaling / dropout	8 / 16 / 0.1
Learning rate	10^{-5}
Training epochs	3
Processed tasks per acquisition event	400
Maximum new probe executions per event	300
Recalled memories per task	50
Test-time beam width	2
Memory cap for ALFWorld / WebShop / BCB	5 / 5 / 3
Main MemSyco memory cap	5
Additional MemSyco memory caps	3, 10
Maximum writing revisions	3
Writing acceptance and admission threshold	0

D.5 Evaluation Metrics and Statistical Reporting

ALFWorld reports task success rate and environment steps on valid-seen and valid-unseen tasks. WebShop reports mean reward, success rate, and interaction steps. BigCodeBench reports the proportion of problems whose generated solutions pass all applicable tests, both overall and on the hard subset. These metrics are reported separately from the training utility defined in Eq. 1.

MemSyco-Bench covers five scenarios in two groups. Objective Fact, Scope Control, and Evidence Conflict evaluate when memory should influence a response using accuracy and sycophancy (Syco). Personalized Use and Valid Selection evaluate how memory is used through accuracy and correct-memory use (Correct), or accuracy and outdated-memory use (Outdated), respectively. Decision and Syco. summarize accuracy and sycophancy across the first three scenarios, while Select and Old correspond to Valid Selection accuracy and outdated-memory use.

Direction-aligned gains compare UpliftMem with native retrieval within each memory store. They use UpliftMem minus native scores for higher-is-better metrics and the reversed difference for sycophancy and outdated-memory use. Cross-store summaries weight the five sources equally. In Table 10, each scenario first averages its two direction-aligned metric gains and the five stores, and Overall averages the five scenarios. This gain summary is distinct from the absolute metrics in Figure 5.

Each configuration is trained once. ALFWorld, WebShop, and BigCodeBench are evaluated three times with a fixed checkpoint. On MemSyco-Bench, the main results are evaluated five times, while

the additional ablation variants are evaluated three times. The Native and Full results reused in the ablation tables retain their original five-run statistics. Reported standard deviations summarize inference-run variability conditional on the trained checkpoint. MemSyco scores are summarized within each source before equal-weight averaging. Table 9 reports per-source dispersion, and Tables 11 and 12 report the per-source ablation means.

E Prompts for Execution, Valuation, and Memory Update

The interfaces below implement the executor and scorer in Sections 3–4.4 and the writer in Appendix B.3. Literal templates are distinguished from builder fields and structured output contracts. Braced fields are populated at runtime.

E.1 Information Boundaries

The executor receives the public task, observations, actions where applicable, and retrieved memory. The current task defines the requirements, while memory supplies transferable experience. Hidden goals, test code, expected answers, and evaluator internals are excluded from executor prompts. Writers receive trajectories and outcome diagnostics, but stored guidance is grounded in visible evidence. Quantitative execution outcomes supervise the value scorer.

Table 7: Prompt interfaces and their principal inputs and outputs.

Interface	Principal inputs	Output
Executor	Public task, observation, retrieved set	Environment action or code
Value scorer	Task context, complete candidate set	Calibrated uplift score
Memory writer	Task, trajectory, outcome, diagnosis	Transferable procedure or rule
Revision	Latest candidate, paired utility, realized trajectory	Revised candidate

E.2 Executor Prompts

The three executors use different action and generation contracts. E1–E3 give their system instructions and the message fields through which retrieved memory enters the task. The same executor instructions and examples apply to paired memory-assisted and no-memory executions.

E1 ALFWorld execution

System

```
You are an ALFWorld household robot agent. Follow the ReAct ALFWorld
transcript style from the examples. Do not output hidden reasoning,
<think> tags, markdown, or explanations. /no_think
At every turn, output exactly one environment command. Prefer the
provided Available actions list when it is present.
```

Initial user message

```
Examples:
{example_texts}

Relevant memories:
[{memory_id}] {memory_text}
...

Now solve the next ALFWorld task in the same command style.
Return exactly one next action line.

{task_description}

Available actions: {command_1}; {command_2}; ...
>
```

Subsequent observation

```
{observation}
Available actions: {command_1}; {command_2}; ...
>
```

The ReAct examples are selected by task type. The current admissible-command list is shared by prompt construction and action validation. Retrieved trajectories describe previous experience; the current episode begins from its own initial observation.

E2 WebShop execution

System

```
You are a WebShop ReAct agent. Reply exactly:
Thought: <short reasoning>
Action: <one action>

Only use search[keywords] on the Search page or click[exact listed
    text].

Page-aware policy:
1. On the Search page, issue one concise query containing the product
     type and distinctive non-price attributes.
2. On the Results page, open an unvisited product whose title matches
     the product type and does not contradict required attributes.
3. On a Product page, use visible page evidence to satisfy
    instruction-related option groups. Select one matching visible
    value per required group. Never substitute conflicting pack
    counts, measurements, or values. If all visible values conflict,
    return to results.

Rules:
```

```
1. Choose only actions executable on the current page and listed in
    Available actions.
2. Do not repeat identical actions or invent clickables.
```

Current user message

```
Instruction:
{instruction_text}

Retrieved memories (reusable guidance; do not copy old product IDs/
    names):
{memory_block}

Recent actions:
{history_block}

Current observation:
{observation_text}

Available actions:
{available_actions_text}

Return exactly:
Thought: <brief reasoning>
Action: search[...] or click[...]
```

E3 BigCodeBench execution

System

```
You are an expert Python programmer solving BigCodeBench coding tasks
    .

You may receive implementation notes distilled from successful
    solutions to related tasks. Use a note only when its contract,
    library, and data flow apply to the current task. The current
    task is the sole specification; memory is supporting evidence,
    not a solution to copy.

Return one self-contained Python solution in a markdown code block,
    without explanation.

Hard constraints:
- Preserve the starter code, function signature, return contract,
    specified exceptions, object types, and observable edge-case
    requirements in the current task.
- Use the libraries and APIs required by the task, import every
    module used, and do not replace a required dependency because a
    memory reports a different execution environment.
- Network access, file I/O, randomness, time-based behavior, plotting
    , and logging are allowed when required by the task, but
    unrelated side effects should not be introduced.
```

User task

The task message is generated by the official `get_prompt(task, prompt_split="instruct")` interface.

Retrieved memory view

```
[Retrieved Memory Context]
# Retrieved Successful Implementation Notes
```

```
Extract applicable implementation evidence for the current task. Do
    not copy the source task's signature, literals, return contract,
    or side effects.

## Note {index} [SUCCESS]
Scope: family={bcb_task_type}; libraries={library_list}
Reusable implementation evidence:
{implementation_only}
```

E.3 Set-Value Scoring Prompt

The `memory` placeholder contains the complete candidate set under the serialization convention in Appendix B.1. The following template produces the yes/no logits calibrated in Appendix C.1.

V1 Uplift scorer

Chat template

```
<|im_start|>system
Judge whether the Document meets the requirements based on the Query
    and the Instruct provided. Note that the answer can only be "yes"
     or "no".<|im_end|>
<|im_start|>user
<Instruct>: {dataset_instruction}
<Query>: {context}
<Document>: {memory}
<|im_end|>
<|im_start|>assistant
<think>

</think>
```

Dataset instructions

```
ALFWorld: Predict whether the memory will causally improve the
    ALFWorld agent for the query context.

WebShop: Predict whether the memory will causally improve the WebShop
     agent for the query context.

BigCodeBench: Predict whether this memory will causally improve
    correctness for the current BigCodeBench coding task.
```

E.4 Memory Construction Prompts

The writer in Appendix B.3 uses different memory representations for interactive control, shopping, and code generation. W1–W4 give its templates, builder fields, and output contracts. Task outcomes describe the executed trajectory, whereas retrieval uplift compares it with the no-memory reference, so a successful trajectory can still have negative uplift.

W1 Successful trajectory proceduralization

User template

```
You are provided with a query and a trajectory taken to solve the
    query. The trajectory consists of multiple steps of thought,
    action and observation.

Your task is to generate a workflow based on critical steps to help
    solve similar queries in the future.

A critical step is one that:
- Has a significant impact on fulfilling the query
- The action is meaningful and contributes to the goal
- The action's outcome is successful

Important: Write the workflow as a natural, coherent paragraph, not a
     bullet list.

-----EXAMPLE WORKFLOW----
To solve this query, begin by identifying the target object's likely
    location based on the task description. Navigate to that location
     and search for the object systematically. Once found, pick up
    the object and verify you have the correct item. Then, determine
    the destination where the object should be placed, navigate there
    , and complete the placement action. Finally, confirm the task is
     complete by checking the environment state.
-----EXAMPLE END----

Query:
{query}

Trajectory:
{trajectory}

Output the workflow without any explanation or context:
```

System constraints

The system message requests `/no_think`, suppresses hidden reasoning, and returns only the workflow memory.

W2 ALFWorld failure-aware construction

Input fields

Task, task type, failed trajectory, failure category, repetition rate, admissible-action mismatch, target-object confusion, last action, and a same-type reference memory.

Output-contract excerpt

```
Propose a procedural memory of 3-7 numbered steps. Focus on:
1. What to verify before acting
2. Which admissible-action patterns are preferred
3. How to identify the correct target object
4. How to recover without repeating the same failed action

Do NOT mention specific room layouts or object IDs.
Output ONLY the numbered procedure.
```

W3 WebShop memory construction

Trajectory proceduralization template

```
Extract ONE short reusable WebShop memory from this trajectory.

Instruction: {instruction}
Trajectory:
{trajectory_text}
Reward: {reward}

Output exactly:
Type: <option_selection | premature_buy_prevention | search_query |
    result_selection | loop_recovery>
Trigger: <the attribute/option situation when to use this>
Action: <one imperative rule>
```

Failure-aware template

```
Write ONE short WebShop memory for the failed trajectory.

Public instruction: {instruction}
Trajectory: {trajectory_text}
Diagnosis: {public_failure_signals}
Public trajectory diff: {public_failure_diff}
Optional related memories: {references_or_empty}

Output exactly:
Type: <option_selection | premature_buy_prevention | search_query |
    result_selection | loop_recovery>
Trigger: <when to use this>
Action: <one imperative rule that prevents the failure>
```

Construction constraints

Each memory contains one atomic rule supported by the public trajectory. Source-specific prices, product identifiers and names, exact queries, and literal search/click actions are excluded. Query revision follows the current page state. For failures, the type preference is option selection, premature purchase, search query, result selection, then loop recovery, subject to support from visible actions and pages.

W4 BigCodeBench memory construction

Success output contract

```
[MEMORY TYPE] SUCCESS_PROCEDURE
[TASK]
<short task abstraction>

[EXECUTION TRAJECTORY]
Contract: <only the public input/output or exception contract
    relevant to transfer>
Implementation: <concise API calls and data flow evidenced by the
    passing code>
Boundary: <one explicitly supported edge condition, or "No additional
     boundary condition established">
Verified idiom: <optional exact code fragment of at most 16 lines;
    include its return when applicable>
```

Failure output contract

```
[MEMORY TYPE] FAILURE_REFLECTION
[TASK]
```

```
<short task abstraction>

[REFLECTION]
<one evidence-supported bug class, why it violates the public
    contract, and one reusable correction>
```

Construction constraints

Success claims are grounded in the public task and visible passing code. The writer preserves supported contracts and implementation evidence, and excludes invented edge cases, API requirements, and references to tests or the evaluator. A failure reflection identifies one supported public-contract mismatch and a reusable correction. The current failed trajectory supplies the primary evidence, with related references included when available. Task-required dependencies remain part of the contract.

The construction constraints accompanying W1, W3, and W4 summarize the corresponding system or builder requirements. The BigCodeBench contracts describe the stored record, while the implementation-only view passed to the executor is shown in Appendix E.2.

E.5 Revision and Admission

The revision loop in Appendix B.3 passes the latest candidate, its realized trajectory, and its validation feedback to W5. Table 8 gives the environment-specific output contracts appended to that wrapper.

W5 Unified revision

Common prompt wrapper

```
/no_think
Revise a procedural memory that did not provide enough uplift on a {
    env_name} task.

## Context
Task: {task_description}
{optional_metadata}

## Current memory (latest validated candidate)
{current_memory}

## Agent's actual trajectory with this memory
{trajectory_text}

## Outcome
- Success: {success}
- Total reward: {total_reward}
- Steps: {total_steps}/{max_steps}
- Utility with memory: {current_utility}
- Utility without memory: {baseline_utility}
- Uplift: {current_uplift} (target > {tau_accept})

## Outcome diagnosis
{failure_signals_rendered}

## Optional related memories
{reference_memories_or_empty}

## Task
Treat the current memory as the latest validated candidate. Contrast
    its intended procedure with the actual trajectory and outcome.
    Preserve only steps that the observable evidence supports,
    identify one concrete decision that limited the result, and
```

```
    revise that decision without reintroducing previously observed
    failures. Do not claim that an unobserved behavior worked.

Briefly identify the observed gap in the current guidance, then
    rewrite it as a short transferable rule.
```

Table 8: Environment-specific output contracts appended to the revision wrapper.

Environment	Revision output
ALFWorld	`WHY IT FAILED`, followed by a procedure of three to seven abstract steps.
WebShop	`WHY IT FAILED`, followed by one `Type / Trigger / Action` rule.
BigCodeBench, passing candidate	`SUCCESS_PROCEDURE` with `Contract`, `Implementation`, and `Boundary`.
BigCodeBench, failing candidate	`FAILURE_REFLECTION` identifying one unresolved bug class.

F Additional Probe Strategy Efficiency Results

We evaluate the same four probe acquisition strategies on ALFWorld and BigCodeBench. In addition to cumulative success rate, we report the success rate of the checkpoint at each probe budget.

F.1 ALFWorld

Figure 7 reports the seen and unseen validation splits separately. At 50 probes, EVSI reaches a cumulative success rate of 72.1% on seen tasks and 65.7% on unseen tasks, compared with 66.4% and 62.7% for random probing. At 200 probes, EVSI reaches 84.3% and 81.3%, respectively, versus 81.4% and 79.9% for random probing. EVSI leads or ties the other strategies at every nonzero budget on seen tasks and leads at six of seven nonzero budgets on unseen tasks. EIG ties EVSI on seen tasks at 200 probes and slightly exceeds it on unseen tasks at 120 probes. The checkpoint success rates fluctuate more than cumulative coverage: at 200 probes, EIG slightly exceeds EVSI on seen tasks (58.6% versus 57.1%), whereas EVSI has the highest success rate on unseen tasks (58.2%). Thus, EVSI's advantage on ALFWorld is clearest in cumulative task coverage.

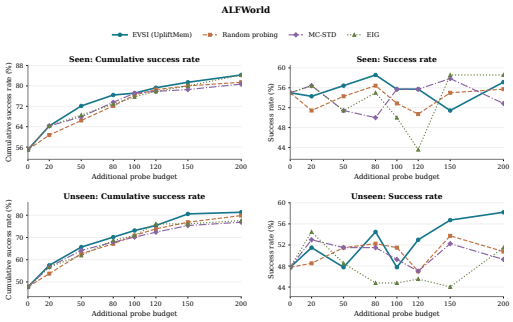

Figure 7. Probe-budget curves on ALFWorld (seen: top; unseen: bottom).

F.2 BigCodeBench

Figure 8 shows that EVSI achieves the highest cumulative success rate at every nonzero budget on BigCodeBench. With 50 probes, EVSI reaches 42.4% cumulative success, compared with 39.5% for random probing and 41.2% for MC-STD. At 200 probes, it reaches 51.8%, compared with 47.4% for random probing and 50.3% for MC-STD. EVSI also has the highest checkpoint success rate at the final budget, reaching 37.7% versus 35.1% for MC-STD and 33.3% for random probing. These results show that EVSI's coverage advantage extends to BigCodeBench.

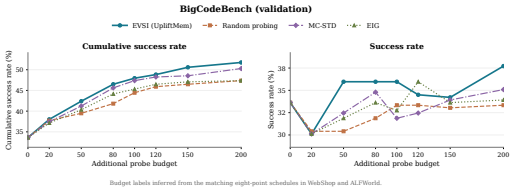

Figure 8. Probe-budget curves on the BigCodeBench validation split.

Memory System	Variant	When to Use Memory						How to Use Memory			
		Objective Fact		Scope Control		Evidence Conflict		Personalized Use		Valid Selection	
		Acc.↑	Syco.↓	Acc.↑	Syco.↓	Acc.↑	Syco.↓	Acc.↑	Cor.Mem.↑	Acc.↑	Outd.Mem.↓
Qwen3-4B											
RAG	Base	35.3±3.5	74.4±4.4	24.4±2.6	32.2±1.1	57.0±1.7	43.0±1.7	85.1±1.7	90.7±1.7	53.7±2.3	47.0±2.5
	UpliftMem	**50.9**±2.4	**49.1**±1.6	**87.1**±1.3	**8.7**±2.0	**84.9**±0.6	**14.9**±1.0	**85.9**±2.8	**91.1**±1.1	**63.8**±2.1	**37.1**±2.2
A-MEM	Base	36.7±1.1	66.3±0.6	48.4±1.3	34.0±2.7	46.2±2.2	53.8±2.2	82.2±3.8	92.2±2.2	55.2±1.9	47.0±3.1
	UpliftMem	**46.9**±0.9	**52.0**±2.3	**92.7**±2.0	**4.7**±1.8	**84.0**±2.8	**15.6**±2.9	**84.0**±2.9	**92.7**±1.1	**66.1**±1.4	**33.7**±1.1
Mem0	Base	52.2±2.6	54.0±2.4	31.5±1.7	28.9±6.8	49.6±3.6	44.4±2.1	58.1±1.3	**72.7**±2.2	46.7±1.6	53.7±1.5
	UpliftMem	**57.8**±2.4	**40.7**±2.8	**66.3**±4.5	**18.7**±2.1	**62.2**±4.8	**33.0**±5.5	**59.0**±0.6	72.4±2.3	**55.2**±3.1	**46.0**±3.2
MemGPT	Base	34.4±2.4	77.4±4.2	61.1±4.0	23.7±1.3	43.1±2.9	55.8±2.1	81.9±2.6	91.1±2.2	61.0±2.5	39.4±2.9
	UpliftMem	**43.1**±2.9	**60.7**±3.6	**84.8**±0.6	**10.4**±2.6	**69.3**±2.2	**27.8**±1.9	**82.9**±1.3	**92.2**±1.1	**65.9**±3.0	**35.6**±3.2
LightMem	Base	52.6±1.3	57.8±1.9	10.4±3.2	44.1±2.3	42.0±2.1	45.8±3.1	70.0±2.7	80.2±1.8	54.9±1.1	47.9±1.1
	UpliftMem	**56.2**±1.1	**46.1**±1.4	**56.4**±1.6	**24.0**±3.3	**55.1**±1.9	**35.8**±2.0	**71.3**±1.8	**81.1**±2.1	**56.8**±1.9	**44.3**±1.5
Qwen3-8B											
RAG	Base	42.2±2.2	52.0±1.6	53.7±2.3	3.7±1.7	39.6±0.6	60.4±0.6	**84.0**±1.3	**82.0**±1.4	57.8±1.1	43.2±0.5
	UpliftMem	**51.8**±1.7	**41.2**±1.0	**82.4**±1.1	**0.4**±1.1	**94.2**±0.9	**5.8**±0.9	82.6±0.6	78.9±1.1	**73.9**±0.9	**26.1**±1.7
A-MEM	Base	41.1±4.4	59.6±2.3	71.6±2.2	0.9±1.2	38.5±1.3	61.5±1.3	82.7±1.0	82.2±1.6	56.2±4.9	47.0±4.0
	UpliftMem	**62.4**±3.2	**28.0**±3.1	**85.6**±1.9	**0.4**±0.6	**82.9**±1.0	**17.1**±1.0	**84.1**±1.9	**82.6**±2.6	**78.1**±3.2	**23.8**±2.8
Mem0	Base	50.7±1.7	48.5±2.8	30.7±2.0	8.1±1.7	25.3±2.1	60.0±2.2	62.4±3.4	59.8±2.4	61.3±2.2	**39.0**±3.4
	UpliftMem	**59.8**±2.1	**29.1**±2.7	**65.2**±4.6	**4.6**±3.6	**59.3**±1.3	**9.3**±2.2	**64.8**±2.3	**61.7**±1.7	**64.8**±1.0	39.9±1.2
MemGPT	Base	37.4±2.3	61.9±3.2	93.6±1.8	1.5±1.7	7.8±1.9	92.2±1.9	81.8±2.0	80.0±1.6	71.2±1.8	31.8±2.7
	UpliftMem	**51.6**±3.0	**44.7**±1.2	**98.8**±1.3	**0.0**±0.0	**36.9**±1.2	**18.4**±1.3	**86.4**±2.6	**85.3**±4.5	**71.6**±2.5	**31.6**±2.9
LightMem	Base	55.2±0.6	43.0±1.3	31.1±1.9	7.8±1.1	47.4±1.7	39.6±1.7	74.1±1.7	71.1±1.1	58.9±2.9	43.2±3.1
	UpliftMem	**59.7**±1.4	**34.9**±1.3	**60.0**±1.9	**5.1**±1.0	**81.8**±1.7	**8.2**±1.3	**75.1**±2.4	**72.4**±1.2	**65.0**±2.0	**37.3**±2.0

Table 9: **Per-system MemSyco results at K=5.** Native retrieval (Base) and UpliftMem use identical memory stores. Entries are means over five inference runs, with ± denoting the sample standard deviation. Arrows indicate the preferred metric direction, and bold marks the better value within each pair.

Backbone	K	When to Use Memory			How to Use Memory		Overall
		Obj. Fact	Scope Ctrl.	Evid. Conf.	Person. Use	Valid Sel.	
Qwen3-4B	3	13.3	**33.7**	19.9	**4.3**	7.2	**15.7**
	5	12.5	30.8	23.3	0.8	7.5	15.0
	10	**14.0**	15.8	**24.3**	2.5	**8.8**	13.1
Qwen3-8B	3	13.6	**12.4**	25.0	**2.8**	**9.7**	12.7
	5	**14.6**	12.3	45.1	1.4	9.4	16.5
	10	14.6	7.1	**53.7**	1.7	8.4	**17.1**
Cross-model avg.	3	13.4	**23.0**	22.4	**3.5**	8.4	14.2
	5	13.5	21.5	34.2	1.1	8.4	**15.8**
	10	**14.3**	11.4	**39.0**	2.1	**8.6**	15.1

Table 10: **Retrieval-budget sensitivity on MemSyco-Bench.** Entries report direction-aligned UpliftMem gains over native retrieval in percentage points. Each task score averages its two metrics and five memory stores, and Overall averages the five task scores. The cross-model average weights Qwen3-4B and Qwen3-8B equally. Bold marks the largest gain within each backbone and column across budgets.

G Additional MemSyco Results

Table 9 provides the absolute per-store scores behind the retrieval gains in Section 5.3. Tables 11 and 12 decompose the ablations in Section 5.4 by memory source. Table 10 reports the scenario-level gains discussed in Section 5.6, using the averaging rules in Appendix D.5.

Memory System	Variant	When to Use Memory						How to Use Memory				Avg.Δ
		Objective Fact		Scope Control		Evidence Conflict		Personalized Use		Valid Selection		
		Acc.↑	Syco.↓	Acc.↑	Syco.↓	Acc.↑	Syco.↓	Acc.↑	Cor.Mem.↑	Acc.↑	Outd.Mem.↓	
RAG	**Full UpliftMem**	**50.9**	**49.1**	**87.1**	8.7	**84.9**	**14.9**	**85.9**	**91.1**	**63.8**	37.1	–
	Base reranker	41.5	65.2	12.6	48.1	65.6	34.4	69.6	79.3	54.3	47.0	22.6
	Warm-up only	41.5	64.1	59.3	24.8	61.5	38.5	69.6	82.2	54.0	47.3	16.1
	Random probing	43.7	58.1	77.8	**8.1**	61.5	38.5	76.3	83.0	63.5	**36.8**	9.0
	w/o policy loss	46.7	57.0	73.3	17.0	70.4	29.6	77.8	84.4	59.4	41.6	8.7
	w/o value loss	40.7	65.2	71.1	16.7	64.1	35.9	81.1	89.6	62.2	40.3	10.3
A-MEM	**Full UpliftMem**	**46.9**	**52.0**	**92.7**	**4.7**	**84.0**	**15.6**	**84.0**	**92.7**	**66.1**	**33.7**	–
	Base reranker	40.4	70.4	14.4	55.9	65.6	34.4	65.9	75.6	55.2	47.0	25.1
	Warm-up only	41.9	63.7	59.3	29.3	50.0	50.0	77.0	83.7	55.2	47.3	18.4
	Random probing	40.7	55.6	69.6	17.8	60.4	39.6	82.6	89.6	59.7	41.6	11.2
	w/o policy loss	39.3	61.1	86.3	11.1	67.8	32.2	79.3	87.8	62.9	37.5	7.9
	w/o value loss	39.3	67.4	68.5	23.7	58.9	41.1	83.1	90.4	63.2	38.4	12.8
Mem0	**Full UpliftMem**	**57.8**	**40.7**	**66.3**	18.7	**62.2**	**33.0**	**59.0**	**72.4**	**55.2**	**46.0**	–
	Base reranker	56.8	48.5	41.9	25.2	52.6	38.1	53.7	66.3	51.4	48.9	7.3
	Warm-up only	53.3	48.1	45.9	23.7	50.0	40.0	57.4	65.6	48.3	52.4	7.8
	Random probing	52.2	50.7	62.3	19.3	54.8	37.4	51.9	63.0	53.0	47.7	5.2
	w/o policy loss	51.1	50.7	54.8	29.6	53.3	39.6	50.7	58.9	54.0	46.3	7.8
	w/o value loss	55.9	48.9	55.6	24.8	61.1	35.9	53.7	64.8	50.5	50.2	5.3
MemGPT	**Full UpliftMem**	**43.1**	**60.7**	**84.8**	**10.4**	**69.3**	**27.8**	**82.9**	**92.2**	**65.9**	**35.6**	–
	Base reranker	30.0	73.7	73.0	15.9	40.0	60.0	78.5	89.6	63.5	39.0	11.8
	Warm-up only	32.2	75.6	75.9	14.1	51.9	47.4	81.5	91.9	60.3	41.6	8.9
	Random probing	30.4	72.2	83.3	11.9	47.0	51.9	80.3	91.3	59.7	41.6	8.9
	w/o policy loss	30.0	70.4	81.5	11.1	52.2	47.0	80.7	91.5	61.0	40.6	7.6
	w/o value loss	31.9	73.3	80.7	12.6	56.3	43.7	82.2	92.2	58.7	42.9	7.4
LightMem	**Full UpliftMem**	**56.2**	**46.1**	**56.4**	**24.0**	**55.1**	**35.8**	**71.3**	**81.1**	**56.8**	**44.3**	–
	Base reranker	55.9	53.3	8.1	46.3	48.1	38.5	57.0	68.9	51.1	50.5	12.6
	Warm-up only	52.6	55.6	22.2	45.2	46.3	42.6	63.3	75.6	50.8	49.5	10.9
	Random probing	55.9	47.4	51.8	25.6	47.0	45.9	66.3	74.4	55.8	44.8	3.9
	w/o policy loss	53.3	47.8	48.9	31.1	41.5	52.2	58.5	74.1	47.3	53.0	8.7
	w/o value loss	55.2	55.6	45.6	36.7	51.9	42.6	65.6	78.1	55.6	46.0	5.6

Table 11: **Per-system ablations at $K=5$ (Qwen3-4B).** Full UpliftMem is averaged over five inference runs, while the ablation variants are averaged over three. Avg.Δ is the mean direction-aligned difference between Full and each variant across the ten metrics, in percentage points, with positive values favoring Full. Bold marks the best value within each memory system and metric.

Memory System	Variant	When to Use Memory						How to Use Memory				Avg.Δ
		Objective Fact		Scope Control		Evidence Conflict		Personalized Use		Valid Selection		
		Acc.↑	Syco.↓	Acc.↑	Syco.↓	Acc.↑	Syco.↓	Acc.↑	Cor.Mem.↑	Acc.↑	Outd.Mem.↓	
RAG	**Full UpliftMem**	**51.8**	**41.2**	**82.4**	**0.4**	**94.2**	**5.8**	**82.6**	**78.9**	**73.9**	**26.1**	–
	Base reranker	47.0	51.5	45.2	1.9	40.0	60.0	74.1	70.0	61.9	38.4	20.4
	Warm-up only	48.5	44.1	61.1	1.5	28.5	71.5	77.4	74.1	61.0	40.3	19.7
	Random probing	48.9	42.2	81.2	**0.4**	31.5	68.5	78.5	75.9	64.4	38.7	16.0
	w/o policy loss	49.6	48.1	73.7	1.1	30.4	69.6	74.8	72.6	64.8	35.6	17.9
	w/o value loss	43.7	49.6	60.4	1.1	53.0	47.0	80.4	78.1	65.1	37.5	14.5
A-MEM	**Full UpliftMem**	**62.4**	**28.0**	**85.6**	**0.4**	**82.9**	**17.1**	**84.1**	**82.6**	**78.1**	**23.8**	–
	Base reranker	38.5	57.4	41.9	2.2	68.9	31.1	79.3	78.5	54.0	48.3	18.4
	Warm-up only	43.0	54.1	80.4	0.7	38.5	61.5	83.3	81.1	54.0	45.7	18.8
	Random probing	54.4	38.5	77.4	**0.4**	49.3	50.7	78.5	76.7	63.2	41.3	13.8
	w/o policy loss	53.7	41.5	84.4	1.1	49.6	50.4	80.7	79.3	61.9	39.0	12.9
	w/o value loss	42.6	55.2	85.2	2.2	52.2	47.8	83.3	80.4	64.1	37.8	14.2
Mem0	**Full UpliftMem**	**59.8**	**29.1**	**65.2**	**4.6**	**59.3**	**9.3**	**64.8**	**61.7**	**64.8**	39.9	–
	Base reranker	50.4	50.0	32.6	11.1	21.9	62.6	57.8	53.7	64.1	**35.9**	17.2
	Warm-up only	46.7	49.3	45.6	4.8	45.9	27.4	51.1	49.3	54.6	43.5	12.5
	Random probing	54.8	34.1	52.6	7.0	44.1	23.7	56.7	55.2	49.2	48.6	9.4
	w/o policy loss	57.8	34.1	63.7	4.8	58.1	23.3	59.3	55.9	51.4	47.3	5.6
	w/o value loss	50.4	47.8	56.7	11.9	37.0	47.0	63.7	59.7	56.8	41.9	11.7
MemGPT	**Full UpliftMem**	**51.6**	**44.7**	**98.8**	**0.0**	**36.9**	**18.4**	**86.4**	**85.3**	**71.6**	**31.6**	–
	Base reranker	44.4	54.1	88.9	1.5	11.9	87.8	80.7	78.1	70.2	35.2	14.0
	Warm-up only	41.5	51.5	92.6	0.7	25.6	74.4	85.9	82.6	64.1	37.5	10.8
	Random probing	41.5	56.7	93.3	0.7	25.6	74.4	85.0	84.4	62.9	40.3	11.5
	w/o policy loss	45.2	50.4	93.3	1.1	28.9	71.1	76.7	73.7	65.4	38.1	11.3
	w/o value loss	39.6	57.4	96.3	0.4	35.9	63.3	81.9	79.3	64.4	38.1	9.8
LightMem	**Full UpliftMem**	**59.7**	**34.9**	**60.0**	**5.1**	**81.8**	**8.2**	**75.1**	**72.4**	**65.0**	**37.3**	–
	Base reranker	54.1	48.1	21.9	5.6	60.4	25.9	59.6	56.3	61.0	40.3	13.5
	Warm-up only	49.3	49.6	30.7	5.6	54.1	23.7	68.1	67.0	55.9	44.8	12.7
	Random probing	55.3	39.8	51.1	8.5	62.2	24.1	73.3	70.7	61.9	37.8	6.4
	w/o policy loss	54.4	41.5	36.3	6.7	62.2	34.4	68.1	65.9	57.5	41.6	10.8
	w/o value loss	53.3	45.9	37.8	6.7	53.0	45.6	67.4	64.8	63.5	39.4	12.6

Table 12: **Per-system ablations at K=5 (Qwen3-8B).** Metrics and reporting conventions follow Table 11.