FROM RELEVANCE TO DECISION VALUE: LOW-OVERHEAD TRAJECTORY-GROUNDED MEMORY VALUATION AND SELECTION FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory selection for large language model (LLM) agents hinges on a fundamental question: *How should the value of a memory unit be defined?* Existing methods often approximate memory value through relevance or novelty, while generative selectors and learned memory policies can require additional training and interaction. A practical observation is that many real-world agents operate repeatedly within bounded workflow domains, naturally accumulating execution histories over time. Motivated by this setting, we take a decision-centric view and define memory value by its query-conditioned contribution to downstream task utility. Based on this definition, we propose a practical trajectory-grounded Value-aware Memory (VAM) estimator, which directly induces a low-overhead Top-K memory selection method without additional learned selectors and rollouts. We further analyze two key questions about the proposed estimator. First, when does the estimated value meaningfully reflect downstream memory contribution? Second, how much trajectory evidence is enough to make the valuation reliable? We answer the first through an average marginal-contribution interpretation, and the second by deriving a Valuation Reliability Assessment (VRA) method to quantify how strongly the available trajectory evidence supports the reliability of the resulting valuation. Experiments on HotpotQA, ALFWorld, and ToolBench demonstrate a favorable performance–cost trade-off across tool use, multi-hop question answering, and interactive decision making. Cross-task evaluation on ALFWorld further shows that values estimated solely from training-task trajectories support effective memory selection for structurally matched, unseen test tasks, without using test-task trajectories for valuation. Additional analyses show that VRA characterizes trajectory-evidence sufficiency for reliable valuation and that VAM progressively improves downstream utility as execution history accumulates.

1 INTRODUCTION

LLM agents increasingly rely on memory for multi-step reasoning and decision making. As agents operate over time and accumulate interactions across tasks and environments, their memory pools can quickly grow (Maharana et al., 2024; Wu et al., 2025b; Liu et al., 2024). The agent must therefore select a compact subset of memories for the current task, making memory selection a central component of agent design (Zhang et al., 2025; Salama et al., 2025). Underlying this selection problem, however, is a more fundamental question: ***How should the value of a memory unit be defined for an LLM agent?*** Without an appropriate notion of value, it remains unclear what criterion memory selection should ultimately optimize.

Existing approaches provide different implicit answers. **1) Relevance-based retrieval** ranks memories by lexical or semantic relevance, recency, novelty, or diversity, which do not directly measure downstream utility (Robertson & Zaragoza, 2009; Karpukhin et al., 2020; Carbonell & Goldstein, 1998; Wu et al., 2025b). **2) Generative selection** uses LLMs to judge memory usefulness, but leaves value implicit and incurs additional generation costs (Shinn et al., 2023; Ouyang et al., 2025; Cao et al., 2025; Cheng et al., 2025). **3) Learned utility and memory policies** align memory operations with downstream feedback, but require dedicated training and exploration (Zhang et al., 2026b; Yan

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.

et al., 2026; Yue et al., 2026; Wu et al., 2026). Thus, existing approaches typically trade off low selection overhead against more direct alignment with downstream utility.

In contrast, this work takes a *decision-centric* view of memory value. Specifically, we define the value of a memory as its query-conditioned contribution to downstream tasks, rather than its semantic relevance or informativeness. This view is related to the broader problem of assigning downstream credit to individual components, as studied in data valuation and reinforcement-learning credit assignment (Ghorbani & Zou, 2019; Ilyas et al., 2022; Arjona-Medina et al., 2019); our focus is instead on query-conditioned valuation of reusable memory units for inference-time selection. We use *decision value* to broadly refer to such downstream utility gains, including improvements in answer correctness, tool-use success, and environment-level task completion. This outcome-grounded definition directly reflects the practical objective of memory selection: identifying memories that improve what the agent ultimately achieves (Xiong et al., 2025).

Meanwhile, a practical observation is that many real-world production agents are deployed within bounded operational domains and are specialized around a relatively stable set of recurring task types or workflow families (Anthropic, 2024; 2026). For example, Klarna's customer-service AI assistant handled 2.3 million conversations in its first month of deployment, while Topstep's Fin AI Agent handles over 150,000 support conversations per month, with AI involved in 97% of them (Klarna, 2024; OpenAI, 2024; Intercom, 2025). These deployments illustrate a common operating regime: a bounded family of task types can generate a much larger number of repeated executions over time, naturally accumulating substantial execution history. When execution trajectories are logged together with their realized outcomes, they provide not only interaction history but also direct evidence about downstream task utility. We therefore ask whether this already accumulated evidence can be reused to estimate memory decision value, rather than generating dedicated trajectories solely for valuation. Importantly, recurring workflows need not involve repeated execution of the identical task. This motivates a further empirical question: can values estimated from prior tasks support memory selection for previously unseen tasks with matching structure, without requiring execution history from the target task itself?

Combining this decision-centric objective with trajectory-supported estimation raises multiple challenges. First, the ideal measurement of decision value is fundamentally *counterfactual*: it requires comparing downstream outcomes with and without each candidate memory, making direct estimation dependent on candidate-specific rollouts. Second, even when historical trajectories are already available, their observational variation must provide a meaningful basis for estimating downstream memory contribution without additional generative selection, dedicated training, or candidate-level evaluation. Third, because only finitely many trajectories may support each memory, we must determine how strongly the observed memory–outcome relationship is supported by the available evidence. The central challenge is therefore to turn accumulated agent trajectories into reliable downstream-grounded memory values while preserving low incremental selection overhead.

Our core contribution is a low-overhead, trajectory-grounded framework for memory valuation and selection. 1) We define memory value by its contribution to downstream task utility and propose Value-aware Memory (VAM), which estimates this value from accumulated trajectories and directly induces Top-K selection without learned or generative selectors or candidate-specific rollouts. 2) We analyze two questions about this estimator: **when does the observational value reflect downstream memory contribution, and how much trajectory evidence is enough to trust it?** Under comparable trajectory contexts, the value admits an average marginal-contribution interpretation. For binary outcomes, we further introduce Valuation Reliability Assessment (VRA) to quantify the statistical support for the observed memory–outcome association. 3) Finally, we evaluate the method on ToolBench (Qin et al., 2024), HotpotQA (Yang et al., 2018), and ALFWorld (Shridhar et al., 2021), covering tool use, multi-hop question answering, and interactive decision making. The method achieves a favorable performance–cost trade-off without additional LLM generation for memory selection. Beyond repeated execution of the same task, we demonstrate cross-task transfer on 183 held-out ALFWorld games, where values estimated exclusively from matched training-task trajectories achieve performance comparable to LLM-based selectors without additional generative selection or test-game trajectories for valuation. Furthermore, our experiments characterize the evidence supporting positive memory–outcome associations and show that VAM progressively improves downstream utility as execution history accumulates.

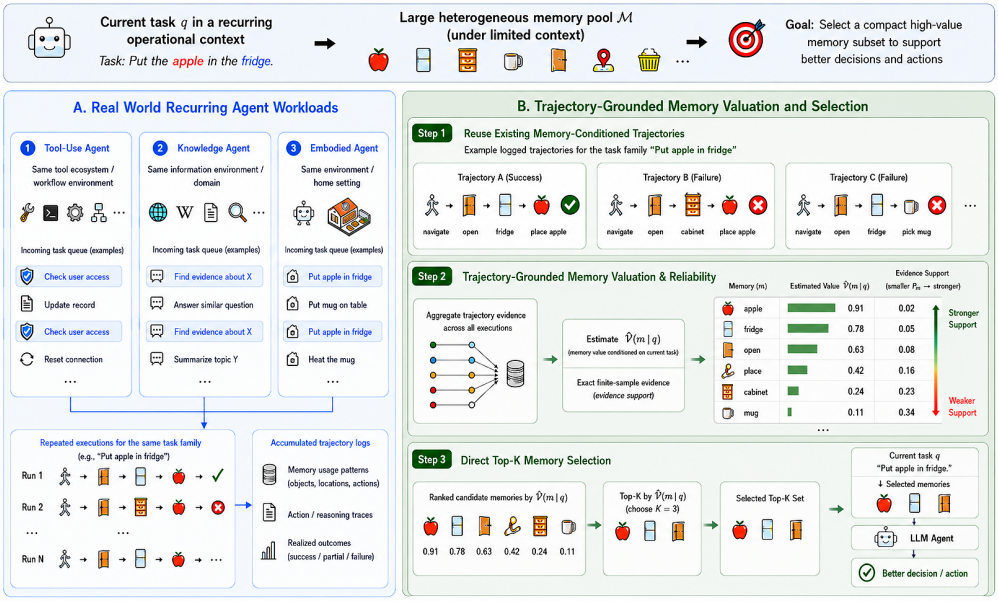

Figure 1: Overview of the proposed trajectory-grounded and decision-centric framework.

2 Related Work

Relevance-Based Retrieval. A common approach assigns value to candidate memories through fixed query-dependent signals, including lexical matching such as BM25, dense embedding similarity, recency, novelty, and diversity-aware criteria such as maximal marginal relevance (MMR) (Robertson & Zaragoza, 2009; Karpukhin et al., 2020; Carbonell & Goldstein, 1998). These methods are inexpensive and scalable, making them a natural default for large memory pools. Such signals measure whether a memory is related, recent, distinctive, or non-redundant, but not whether its inclusion changes the downstream reasoning process or improves the eventual task outcome.

Generative Selection. A second line of work uses LLM generation to interpret, transform, or evaluate past experience before reuse, including Reflexion, Meta-Policy Reflexion, Agent Workflow Memory, and WebATLAS (Shinn et al., 2023; Wu et al., 2025a; Wang et al., 2025b; Cheng et al., 2025), as well as SE-Agent (Lin et al., 2025). These methods can move beyond surface relevance by considering procedural knowledge, causal consequences, and task-specific usefulness. Nevertheless, they generally leave memory value implicit in the selector's generative judgment, rather than defining an explicit, outcome-grounded, and reusable value for each memory unit. Memory selection therefore becomes an additional reasoning process, incurring extra token and latency costs while making selection quality dependent on the capability and reliability of the selector model.

Learned Utility and Memory Policies. Recent work moves more explicitly toward outcome-aware valuation by learning memory scores, memory operations, or retrieval policies from downstream feedback, including Memory-R1, Mem-α, and Mem-T (Yan et al., 2026; Wang et al., 2025a; Yue et al., 2026), U-Mem and MemGen (Wu et al., 2026; Zhang et al., 2026a). This direction is conceptually closest to downstream-utility-grounded selection because memory behavior is optimized against task outcomes rather than query–memory correspondence alone. However, obtaining such utility signals commonly requires policy training, repeated online feedback, rollout-based exploration, model-internal access, or long-horizon credit assignment.

Positioning of Our Work. Different from the above studies, our framework focuses on estimating explicit, downstream-utility-grounded memory values from execution histories accumulated through prior operation. More broadly, our decision-centric formulation connects to downstream contribution estimation and credit assignment in data valuation and reinforcement learning (Ghorbani & Zou, 2019; Ilyas et al., 2022; Wang & Jia, 2023; Arjona-Medina et al., 2019). We bring these ideas to reusable memories conditioned on the current query, estimating their values from logged agent trajectories for direct inference-time selection. The resulting VAM estimates directly

support Top-K selection without additional generative selectors, parameter training, or candidate-specific rollouts. We establish conditions under which the observational value admits an average marginal-contribution interpretation and introduce VRA to assess the trajectory evidence supporting the observed memory–outcome association. Beyond same-task history reuse, we evaluate cross-task transfer on structurally matched, held-out ALFWorld tasks, using exclusively training-task trajectories for valuation.

3 Proposed Method

Problem formulation. We consider a memory-augmented reasoning process that produces a trajectory $\tau = (s_1, a_1, \ldots, s_T, a_T)$, where T denotes the trajectory horizon, s_t denotes the reasoning state at step t, and a_t denotes the corresponding memory-aware reasoning action. Let $R(\tau)$ denote the final task reward, such as QA correctness, verifier score, or task-completion success. Let $\mathcal{M} = \{m_1, m_2, \ldots, m_N\}$ denote a candidate memory pool, where N is the number of candidate memory units. Given a current query or task q, our goal is to select a subset of K memories that maximizes the expected downstream reward:

$$S_{\text{opt}}(q) = \arg\max_{\{S \subseteq \mathcal{M}, |S|=K\}} \mathbb{E}_{\tau \sim p(\tau|q,S)} \left[R(\tau) \right], \tag{1}$$

where $p(\tau \mid q, S)$ denotes the trajectory distribution induced by the reasoning agent given query q and memory subset S. Eq. (1) is a combinatorial optimization problem over $\binom{N}{K}$ possible subsets. Direct evaluation is expensive because each candidate subset requires a complete downstream reasoning rollout or verifier-based assessment. We therefore seek a tractable surrogate based on memory-level value estimation.

Decision-Centric Memory Value and Trajectory-Grounded Estimation We first define the ideal quantity that memory selection should capture. For each memory unit $m \in \mathcal{M}$, we define its ideal counterfactual value as

$$V^{\star}(m \mid q) = \mathbb{E}_{\tau \sim p(\tau|q,m)} \left[R(\tau) - R(\tau^{-m}) \right], \tag{2}$$

where τ^{-m} denotes the corresponding counterfactual trajectory in which m is removed, masked, or replaced while preserving the remaining reasoning and memory context as much as possible. Thus, $V^{\star}(m \mid q)$ measures the expected downstream contribution of memory m, rather than its semantic relevance or standalone informativeness.

Directly estimating Eq. (2) would require dedicated counterfactual rollouts for individual memories. Instead, we reuse trajectories that have already been produced during prior agent interactions. Let $\tilde{R}(\tau)$ denote the task-aligned utility used for practical valuation. It may coincide with the terminal reward $R(\tau)$ or provide a denser task-aligned surrogate when terminal feedback is sparse. For query or task context q, let $\mathcal{D}_q = \left\{ (\tau_i, A_i, \tilde{R}_i) \right\}_{i=1}^{N_\tau}$, $\tilde{R}_i = \tilde{R}(\tau_i)$, where $A_i \subseteq \mathcal{M}$ is the set of memory units recorded as present in trajectory τ_i. $\mathcal{D}_q$ may contain trajectories from the same task instance or from related tasks within a matched task family, depending on the deployment setting. Here we assume these historical trajectories have been generated before and are reused as logged evidence rather than produced by an additional rollout. Let $c_i(m) = \mathbf{1}[m \in A_i]$ indicate whether memory m occurs in trajectory i. We define an amortized estimator[1]

$$\widehat{V}_{\text{am}}(m \mid q) = \frac{\sum_{i=1}^{N_\tau} c_i(m) \tilde{R}_i}{\sum_{i=1}^{N_\tau} c_i(m)} - \frac{\sum_{i=1}^{N_\tau} (1 - c_i(m)) \tilde{R}_i}{\sum_{i=1}^{N_\tau} (1 - c_i(m))}, \tag{3}$$

Eq. (3) estimates memory value from logged trajectories through a presence–absence difference in mean utility, the treatment–control contrast of causal inference (Imbens & Rubin, 2015) and the form of the maximum-sample-reuse estimator in Data Banzhaf (Wang & Jia, 2023). This empirical difference converges to the population-level value in the infinite-trajectory limit:

$$V_{\text{inf}}(m \mid q) = \mathbb{E}[\tilde{R} \mid c(m) = 1, q] - \mathbb{E}[\tilde{R} \mid c(m) = 0, q]. \tag{4}$$

Thus, $\widehat{V}_{\text{am}}$ is the finite-trajectory estimator of V_{inf}. The following analysis characterizes when this observational quantity admits an average downstream marginal-contribution interpretation. When

[1] We consider memories that are both present and absent across the logged trajectories, so that the presence–absence contrast is defined. Memories lacking such variation are excluded from value-based ranking.

$\tilde{R} = R$, this interpretation directly applies to the terminal task utility in Eq. (2); otherwise, it applies to surrogate utility $\tilde{R}$, whose usefulness depends on its alignment with the terminal objective.

Why Does Trajectory-Grounded Valuation Work? The estimator above uses an observational presence–absence contrast. We next show why this contrast can reflect the downstream contribution of a memory itself. For a fixed query q, define $F_q(S) = \mathbb{E}_{\tau \sim p(\tau|q,S)}\left[\tilde{R}(\tau)\right]$ as the expected downstream utility obtained with memory subset S. For a candidate memory m, let $B \subseteq \mathcal{M} \setminus \{m\}$ denote the remaining memory context of a trajectory. The context-specific marginal contribution of m is $\Delta_m(B \mid q) = F_q(B \cup \{m\}) - F_q(B)$. We assume that, conditional on the remaining memory context B, logged outcomes with and without m reflect the corresponding downstream utilities. We further assume that the memory presence and absence groups provide comparable coverage of B.

Proposition 3.1 (Trajectory-context interpretation)**.** Suppose that logged outcomes are contextually representative and that trajectories containing and not containing m induce the same distribution over the remaining memory context B. Then

$$V_{\text{inf}}(m \mid q) = \mathbb{E}_B\left[\Delta_m(B \mid q)\right]. \tag{5}$$

The proof is provided in Appendix A.1 [2]. Proposition 3.1 gives the key trajectory-level interpretation: under comparable logged contexts, V_{inf} measures how much memory m changes downstream utility on average across the trajectory contexts in which it may be used. It is therefore not merely a measure of how often m appears in successful trajectories.

This interpretation becomes especially intuitive in deterministic binary-outcome tasks. Suppose $F_q(S) \in \{0, 1\}$ and memory inclusion is monotone, $F_q(B \cup \{m\}) \geq F_q(B)$. Then $\Delta_m(B \mid q) \in \{0, 1\}$, equaling one exactly when m changes the outcome from failure to success in context B. Calling such a memory *pivotal* in B, Proposition 3.1 yields

$$V_{\text{inf}}(m \mid q) = \Pr_B\left[m \text{ is pivotal for the downstream outcome}\right]. \tag{6}$$

Thus, V_{inf} measures how frequently m is outcome-critical across trajectory contexts; further interpretation is provided in Appendix A.1.

When Is Trajectory-Grounded Valuation Reliable? We have established a population-level interpretation of $V_{\text{inf}}(m \mid q)$. We now ask a complementary practical question: *how much trajectory evidence is enough to trust an estimated memory value?* To address this question, we introduce *Valuation Reliability Assessment (VRA)*, which quantifies how strongly the available trajectory evidence supports an observed memory valuation. We focus on binary trajectory outcomes, $\tilde{R} \in \{0, 1\}$, as in tasks evaluated by answer correctness, execution success, or task completion. For each candidate memory m, the trajectories have four possible observations:

	$\tilde{R} = 1$	$\tilde{R} = 0$
$c(m) = 1$	n_{11}	n_{10}
$c(m) = 0$	n_{01}	n_{00}

, (7)

where n_{11} counts successful trajectories containing m, n_{10} counts failed trajectories containing m, n_{01} counts successful trajectories without m, and n_{00} counts failed trajectories without m.

Assume that, conditional on q, the logged trajectories are independent samples from a fixed logging distribution. We test the null hypothesis $c(m) \perp \tilde{R} \mid q$. Conditioned on the row and column margins of Eq. (7), the number of successful trajectories containing m follows a hypergeometric distribution. Thus, for a positive observed association, the corresponding one-sided exact probability is given by Fisher's exact test (Fisher, 1935):

$$p_{\text{ex}}(m) = \sum_{x=n_{11}}^{\min(n_{1\cdot}, n_{\cdot 1})} \frac{\binom{n_{1\cdot}}{x}\binom{N_\tau - n_{1\cdot}}{n_{\cdot 1} - x}}{\binom{N_\tau}{n_{\cdot 1}}}, \tag{8}$$

where $N_\tau = n_{11} + n_{10} + n_{01} + n_{00}$, $n_{1\cdot} = n_{11} + n_{10}$, and $n_{\cdot 1} = n_{11} + n_{01}$.

[2] The appendix further shows that, when the two context distributions are not exactly matched, the observational value decomposes into the average marginal contribution plus a context-shift term; for bounded utility, the magnitude of this additional term can be controlled by the discrepancy between the two context distributions

Proposition 3.2 (Valuation Reliability from Trajectory Evidence)**.** Based on the exact association evidence in Eq. (8), we define the *Valuation Reliability Assessment* as

$$\mathrm{VRA}(m \mid q; \mathcal{D}_q) \triangleq 1 - p_{\mathrm{ex}}(m). \tag{9}$$

When no successful trajectory is observed without m, i.e., $n_{01} = 0$, the exact probability simplifies to $p_{\mathrm{ex}}(m) = \binom{n_{11}+n_{10}}{n_{11}} / \binom{N_\tau}{n_{11}}$. Furthermore, letting $q_m = (n_{11} + n_{10})/N_\tau$ denote the observed occurrence rate of memory m, we have $\mathrm{VRA}(m \mid q; \mathcal{D}_q) \geq 1 - q_m^{n_{11}}$. The derivation and further properties are given in Appendix A.2.

VRA quantifies how strongly the current trajectory evidence supports the observed positive memory–outcome association underlying the valuation. Importantly, it is an evidence-support quantity rather than a posterior probability that the estimated memory value is correct. Thus, $\widehat{V}_{\mathrm{am}}(m \mid q)$ determines the value ranking, whereas VRA assesses the finite-sample evidence supporting that valuation. More generally, reliability depends on the full occurrence–outcome composition of the logged trajectories rather than on a universal trajectory-count threshold.

From Memory Value to Direct Top-K Selection The downstream reasoning system ultimately receives a set of selected memories. To lift individual memory values to set-level selection, we use the first-order surrogate $U^{(1)}(S \mid q) = \sum_{m \in S} V_{\mathrm{inf}}(m \mid q)$, $|S| = K$. Under the cardinality constraint $|S| = K$, maximizing $U^{(1)}(S \mid q)$ is equivalent to selecting the K memories with the largest population observational values[3]. Since these values are unknown, practical selection directly uses their trajectory-grounded estimates:

$$\widehat{S}(q) = \mathrm{TopK}_{m \in \mathcal{M}} \widehat{V}_{\mathrm{am}}(m \mid q). \tag{10}$$

In summary, Proposition 3.1 gives conditions for an average marginal-contribution interpretation, while Proposition 3.2 provides the statistical basis for VRA. Thus, $\widehat{V}_{\mathrm{am}}$ determines the Top-K ranking, and VRA assesses evidence supporting the memory–outcome association. Selection requires no parameter training, candidate-specific rollouts, or additional LLM generation at inference time[4].

4 Experiments

4.1 Experimental Setup

Benchmarks and Implementation. We evaluate our method on three benchmarks covering complementary forms of memory-augmented reasoning: 1) ToolBench for multi-step tool use, 2) HotpotQA for multi-hop question answering, and 3) ALFWorld for interactive embodied decision-making. Following the cardinality-constrained formulation, we retain $K = 2$, $K = 3$, and $K = 2$ memory units for HotpotQA, ALFWorld, and ToolBench respectively (A detailed memory-budget sensitivity analysis can be found in Appendix B.1). We report success rate for ToolBench and ALFWorld, and Joint F1 as the primary metric for HotpotQA. For each benchmark, our proposed method operates on a collection of memory-subset trajectories produced before the final memory-selection and evaluation stage. We describe the benchmark-specific trajectory construction protocol and data separation in Appendix C.1. We use MiniMax-M2.7 for ToolBench and HotpotQA, and DeepSeek-V4-Flash for ALFWorld. Using two different model families allows us to evaluate whether the proposed method remains effective across distinct backbone models. Within each benchmark, the same model is used for both downstream reasoning and all LLM-based selection baselines.

Baselines. We compare the proposed Value-Aware Memory (VAM) against eight baselines, grouped into four categories: 1) **No Memory** evaluates the backbone model without memory augmentation. 2) **Retrieval-based methods** include BM25, embedding similarity, and maximal marginal relevance (MMR), which select memory units based on lexical or semantic relevance without using trajectory outcomes. 3) **Trajectory-statistics baseline**: Presence-only Mean Reward (PMR) ranks memories by the average reward of logged trajectories in which they appear, using the same trajectory evidence as VAM but omitting the memory-absence term. 4) **LLM-based methods** include an LLM

[3]This first-order approximation does not require cross-memory interactions to vanish exactly. Appendix A.3 analyzes the effect of residual interactions on value-based set selection.

[4]Diversity-aware reranking is optional for overlapping memories; see Appendix B.2.

Selector, which selects memories from the query and candidate pool; Procedural Selection, which additionally uses successful trajectories; and Reflection Selection, which contrasts successful and failed trajectories. All memory-selection methods select from the same candidate memory pool and return the same number of memory units.

4.2 Experiment Results and Analyses

Table 1: Overall performance in the same-instance repeated-execution setting, where memory values are estimated from prior trajectories of the same task instance. The evaluation covers 43 ToolBench tasks, 100 HotpotQA questions, and 100 ALFWorld games, using 3,620 logged trajectories in total (Detailed in Appendix C.1). Values in parentheses denote the percentage of total input tokens consumed by the memory-selection stage. Performance is reported as mean ± standard deviation over 10 repeated evaluation runs.

Method	HotpotQA		ALFWorld		ToolBench	
	Joint F1 ↑	Avg. Total Input Tokens ↓ (Sel. Share)	Success Rate ↑	Avg. Total Input Tokens ↓ (Sel. Share)	Success Rate ↑	Avg. Total Input Tokens ↓ (Sel. Share)
No Memory	0.000 ± 0.000	535 (0%)	53.0 ± 1.4%	40,554 (0%)	48.8 ± 1.0%	**9,793** (0%)
BM25	0.216 ± 0.013	518 (0%)	85.0 ± 1.2%	23,171 (0%)	53.5 ± 1.8%	11,363 (0%)
Embedding Similarity	0.222 ± 0.017	552 (0%)	84.0 ± 1.6%	23,392 (0%)	55.8 ± 1.5%	12,243 (0%)
MMR	0.210 ± 0.011	571 (0%)	74.0 ± 0.9%	23,668 (0%)	51.2 ± 1.8%	12,788 (0%)
PMR	0.415 ± 0.011	511 (0%)	93.0 ± 0.8%	8,704 (0%)	62.8 ± 2.1%	10,158 (0%)
LLM Selector	0.497 ± 0.015	2,021 (75.6%)	93.0 ± 1.1%	8,730 (4.77%)	69.8 ± 2.3%	14,597 (23.77%)
Reflection	**0.501 ± 0.012**	6,901 (92.9%)	**98.0 ± 0.8%**	9,240 (11.43%)	62.8 ± 1.8%	22,958 (48.48%)
Procedural	0.469 ± 0.018	4,589 (89.3%)	97.0 ± 1.3%	9,815 (7.75%)	**72.1 ± 1.0%**	15,498 (30.65%)
Proposed VAM	0.485 ± 0.010	**493** (0%)	97.0 ± 1.0%	**8,667** (0%)	69.8 ± 1.5%	9,797 (0%)

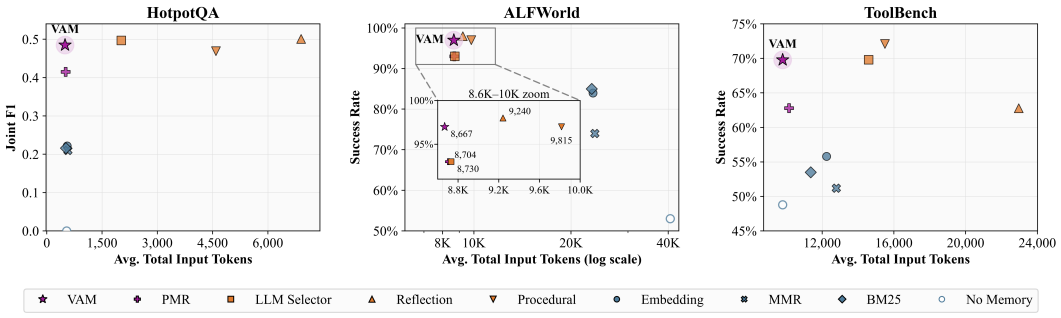

Figure 2: Performance–cost trade-off across datasets, comparing downstream performance against total input tokens across selection and execution.

Overall Performance Table 1 and Figure 2 show that VAM achieves a consistently favorable performance–cost trade-off across all three benchmarks, lying on or near the upper-left Pareto frontier. On HotpotQA, it achieves a Joint F1 of 0.485 with the fewest input tokens among all methods; LLM Selector and Reflection obtain slightly higher scores but require approximately 4.1–14.0× as many input tokens. On ALFWorld, VAM achieves 97.0% success, comparable to Reflection at 98.0%, while avoiding a generative selection stage. On ToolBench, it matches LLM Selector at 69.8% success with only 9.8K input tokens per instance, while Procedural achieves 72.1% at 15.5K. Compared with PMR, VAM improves HotpotQA Joint F1 by 0.070 and ALFWorld and ToolBench success rates by 4.0 and 7.0 percentage points, respectively, while using fewer input tokens on all three benchmarks. These results support the value of incorporating memory-absence outcomes beyond presence-only reward statistics. On HotpotQA and ToolBench, the token savings relative to LLM-based selectors primarily come from eliminating generative selection overhead, whereas on ALFWorld effective memories also shorten downstream interaction trajectories and reduce cost.

Cross-Task Transfer Evaluation We evaluate cross-task transfer on 183 held-out ALFWorld test games, with both VAM and PMR values estimated exclusively from training-set trajectories and no test-game trajectory used for valuation. Table 2 reports performance across six distinct task types. VAM achieves the highest overall success rate of 95.3%, outperforming No Memory (43.5%), Embedding Similarity (82.1%), and PMR (65.6%). The 29.7-percentage-point advantage over PMR

Table 2: Cross-task transfer from training trajectories to 183 playable held-out ALFWorld test games, with valuation evidence drawn from 686 training families. VAM values are estimated exclusively from the training split; no test-game trajectory is used for valuation. For each test game, the source pool is constructed using ALFWorld-provided PDDL attributes, matching the target object and goal/receptacle. *Avg. Total Input Tokens* reports end-to-end input tokens, including both memory selection and downstream execution. Results are mean ± SD over 10 evaluation runs.

Method	Overall	Held-out Task Type: Success Rate (%)						Avg. Total Input Tokens/Run ↓
		Pick-Place	Two-Obj.	Heat	Cool	Clean	Look-at	
No Memory	43.5 ± 1.1	54.8 ± 1.5	35.5 ± 1.6	45.8 ± 0.0	15.2 ± 1.5	50.8 ± 1.2	59.5 ± 2.4	43.03K
Embedding Similarity	82.1 ± 0.8	93.8 ± 1.2	80.3 ± 1.6	79.2 ± 1.9	65.5 ± 1.5	86.8 ± 1.2	80.5 ± 2.4	20.74K
PMR	65.6 ± 0.3	97.6 ± 1.1	55.2 ± 1.5	54.2 ± 0.0	35.5 ± 0.0	57.9 ± 1.2	89.5 ± 0.0	31.89K
LLM Selector	91.0 ± 0.7	97.6 ± 0.0	70.0 ± 1.6	95.8 ± 0.0	91.3 ± 1.5	94.7 ± 1.2	**94.7 ± 0.0**	12.46K
Procedural	92.9 ± 0.6	**100.0 ± 0.0**	78.3 ± 1.6	**100.0 ± 0.0**	93.5 ± 0.0	94.7 ± 1.2	85.8 ± 2.4	12.56K
Reflection	94.9 ± 0.5	**100.0 ± 0.0**	**87.2 ± 1.6**	**100.0 ± 0.0**	90.3 ± 1.4	97.4 ± 0.0	91.1 ± 2.4	11.64K
VAM	**95.3 ± 0.6**	**100.0 ± 0.0**	85.2 ± 1.6	98.8 ± 1.9	**95.8 ± 1.5**	**100.0 ± 0.0**	85.8 ± 2.4	**10.81K**

supports the benefit of incorporating memory-absence outcomes in this transfer setting. VAM performs comparably to Reflection at 94.9%, while requiring no generative selection stage and using the fewest end-to-end input tokens per execution. These results show that decision values estimated entirely from training-task trajectories can transfer effectively to structurally matched, previously unseen test tasks, extending VAM beyond repeated execution of the same target task.

Empirical Validation of Decision-Value Ranking. The remaining analyses return to the same-instance setting. We exhaustively evaluate singleton and pairwise memory configurations on HotpotQA to construct an empirical ranking based on average marginal contributions, and assess how well each selector recovers its Top-2 memories (Appendix B.3). This reference is obtained through controlled downstream interventions, independently of the logged trajectories used by VAM.

Table 3 shows that semantic relevance alone does not reliably identify memories with high empirical decision value. LLM-based selectors achieve stronger alignment, reaching 60–64% Exact@2 and 77.0–79.5% Recall@2. VAM achieves 66.0% and 80.5%, respectively, matching or slightly exceeding these selectors without generative selection at inference time. Mean Value Regret@2 complements selection agreement by measuring the loss in summed empirical memory values relative to the reference Top-2 set. VAM yields the lowest regret of 0.0101, versus 0.0220–0.0504 for LLM-based selectors and 0.3861–0.4321 for relevance-based retrieval. Thus, VAM's selections both closely match the empirical reference and incur smaller value losses when they differ. These results validate selection quality rather than numerical score calibration.

Table 3: Agreement with the empirical decision-value ranking on 100 HotpotQA tasks. **Exact@2** reports the percentage of tasks for which a method exactly recovers the empirical Top-2 memory set. **One-Match** and **Zero-Match** report the number of tasks recovering one or none of the empirical Top-2 memories, respectively. **Recall@2** measures the average fraction of empirical Top-2 memories recovered. **Mean Value Regret@2** measures the average empirical decision-value loss relative to the optimal empirical Top-2 set.

Method	Exact@2 ↑	One-Match	Zero-Match	Recall@2 ↑	Mean Value Regret@2 ↓
BM25	10.0%	55	35	37.5%	0.4199
Embedding Similarity	11.0%	54	35	38.0%	0.4321
MMR	11.0%	61	28	41.5%	0.3861
LLM Selector	62.0%	33	5	78.5%	0.0265
Reflection	60.0%	34	6	77.0%	0.0504
Procedural	64.0%	31	5	79.5%	0.0220
VAM	**66.0%**	**29**	**5**	**80.5%**	**0.0101**

Valuation Reliability and Trajectory Sufficiency. Figure 3 uses a synthetic simulation to examine how trajectory quantity and outcome composition affect memory valuation and reliability. A key observation is that a large value estimate does not necessarily imply strong support: under sparse trajectory evidence, ground-truth critical memories may already receive high $\widehat{V}_{\mathrm{am}}$ values while their VRA remains low. As more informative trajectory evidence accumulates, the critical memories become both consistently high-valued and strongly supported by VRA, while non-critical values stabilize near zero. The comparison across success–failure compositions further shows that trajectory sufficiency depends not only on the number of logged trajectories, but also on how informative their

outcome coverage is. Together, these results highlight the complementary roles of $\widehat{V}_{\mathrm{am}}$ and VRA: the former determines the Top-K ranking, whereas the latter quantifies how strongly the available trajectory evidence supports the resulting valuation.

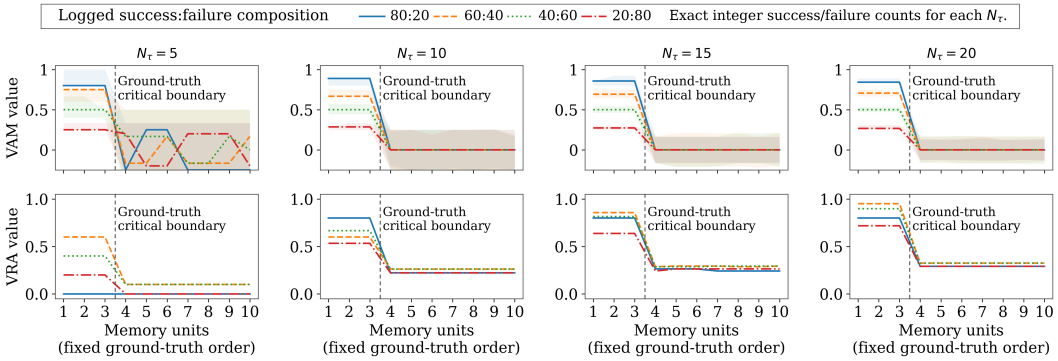

Figure 3: Synthetic experiment under different trajectory budgets and logged success–failure compositions. The three ground-truth critical memories (m_1–m_3) are shown first, followed by seven non-critical memories in a fixed order. The top row reports median VAM values over 1,000 repeated simulations, with shaded regions denoting the 25th–75th percentile range. The bottom row reports median VRA over the same 1,000 simulations, where larger values indicate stronger evidence.

Progressive Improvement with Accumulated Execution History. Figure 4 shows that VAM progressively improves as execution history accumulates. With only two prior executions, only 29.8% of candidate memories are identifiable and the average VRA among the selected Top-4 memories remains low; by eight executions, coverage reaches 79.1% and average VRA exceeds 80%, with both approaching their full-history levels thereafter. This stronger evidence is also reflected in the selection itself, which increasingly recovers the full-history reference set, while downstream success for $K = 4$ rises from 84.0% at $N_\tau = 2$ to 97.5% at $N_\tau = 8$, before stabilizing at 98.5%. Together, these results show that VAM is not a static procedure that requires a fully populated trajectory pool before becoming useful; rather, it can continually update from naturally accumulated execution evidence and enter a practically useful regime after only a modest number of prior executions.

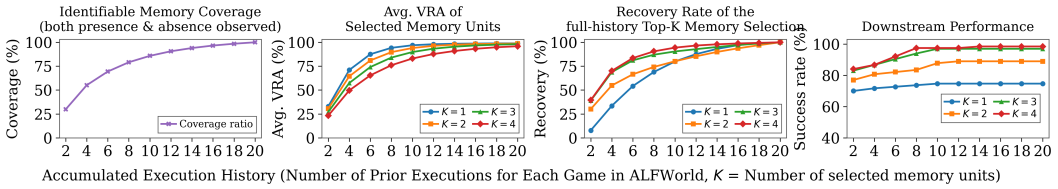

Figure 4: Progressive VAM improvement with accumulated execution history on ALFWorld. $K = 1$ to 4 indicates the number of selected memory units, and each history budget is evaluated over all 100 ALFWorld games.

5 Conclusion

This work introduced VAM, a low-overhead, trajectory-grounded approach that estimates memory value from its query-conditioned contribution to downstream utility. VAM directly supports Top-K selection without training, candidate-specific rollouts, or online generative selection. Across ToolBench, HotpotQA, and ALFWorld, VAM achieves a favorable performance–cost trade-off, while cross-task ALFWorld results further show that values estimated from training trajectories can transfer to structurally matched test games. VRA complements VAM by quantifying trajectory-evidence support, and progressive-history experiments show that valuation improves as execution evidence accumulates. The current study focuses on settings with logged outcome supervision and structurally related task histories; broader transfer and deployment settings remain for future investigation.

REFERENCES

Anthropic. Building effective agents. https://www.anthropic.com/engineering/building-effective-agents, December 2024. Accessed: 2026-08-20.

Anthropic. Building effective ai agents: Architecture patterns and implementation frameworks. https://resources.anthropic.com/building-effective-ai-agents, 2026. Accessed: 2026-08-20.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, volume 32, pp. 13544–13555, 2019.

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. *arXiv preprint arXiv:2512.10696*, 2025.

Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 335–336, 1998.

Jiali Cheng, Anjishnu Kumar, Roshan Lal, Rishi Rajasekaran, Hani Ramezani, Omar Zia Khan, Oleg Rokhlenko, Sunny Chiu-Webster, Gang Hua, and Hadi Amiri. WebATLAS: An LLM agent with experience-driven memory and action simulation. *arXiv preprint arXiv:2510.22732*, 2025.

Ronald A. Fisher. *The Design of Experiments*. Oliver and Boyd, Edinburgh, 1935.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2242–2251. PMLR, 2019.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9525–9587. PMLR, 2022.

Guido W. Imbens and Donald B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, 2015.

Intercom. From chat to voice: Ai at topstep's frontline. https://community.intercom.com/ai-transformation-stories-104/from-chat-to-voice-ai-at-topstep-s-frontline-13226, November 2025. Accessed: 2026-08-20.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proc. Conf. Empir. Methods Nat. Lang. Process.*, pp. 6769–6781, 2020.

Klarna. Klarna ai assistant handles two-thirds of customer service chats in its first month. https://www.klarna.com/international/press/klarna-ai-assistant-handles-two-thirds-of-customer-service-chats-in-its-first-month/, February 2024. Accessed: 2026-08-20.

Jiaye Lin, Yifu Guo, Yuzhen Han, Sen Hu, Ziyi Ni, Licheng Wang, Mingguang Chen, Hongzhang Liu, Ronghao Chen, Yangfan He, Daxin Jiang, Binxing Jiao, Chen Hu, and Huacan Wang. SE-Agent: Self-evolution trajectory optimization in multi-step reasoning with LLM-based agents. *arXiv preprint arXiv:2508.02085*, 2025.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguist.*, 12:157–173, 2024.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proc. Annu. Meet. Assoc. Comput. Linguist.*, pp. 13851–13870, 2024.

OpenAI. Klarna's ai assistant does the work of 700 full-time agents. https://openai.com/index/klarna/, 2024. Accessed: 2026-08-20.

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

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *Proc. Int. Conf. Learn. Represent.*, 2024.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, 2009.

Rana Salama, Jason Cai, Michelle Yuan, Anna Currey, Monica Sunkara, Yi Zhang, and Yassine Benajiba. MemInsight: Autonomous memory augmentation for LLM agents. In *Proc. Conf. Empir. Methods Nat. Lang. Process.*, pp. 33136–33152, 2025.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik R. Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Adv. Neural Inf. Process. Syst.*, volume 36, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *Proc. Int. Conf. Learn. Represent.*, 2021.

Jiachen T. Wang and Ruoxi Jia. Data banzhaf: A robust data valuation framework for machine learning. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 6388–6421. PMLR, 2023.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proc. Int. Conf. Mach. Learn.*, pp. 63897–63911. PMLR, 2025b.

Chunlong Wu, Ye Luo, Zhibo Qu, and Min Wang. Meta-policy reflexion: Reusable reflective memory and rule admissibility for resource-efficient LLM agent. *arXiv preprint arXiv:2509.03990*, 2025a.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *Proc. Int. Conf. Learn. Represent.*, 2025b.

Xinle Wu, Rui Zhang, Mustafa Anis Hussain, and Yao Lu. Towards autonomous memory agents. *arXiv preprint arXiv:2602.22406*, 2026.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts LLM agents: An empirical study of experience-following behavior. *arXiv preprint arXiv:2505.16067*, 2025.

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 *Proc. Annu. Meet. Assoc. Comput. Linguist.*, pp. 12805–12825, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proc. Conf. Empir. Methods Nat. Lang. Process.*, pp. 2369–2380, 2018.

Yanwei Yue, Boci Peng, Xuanbo Fan, Jiaxin Guo, Qiankun Li, and Yan Zhang. Mem-T: Densifying rewards for long-horizon memory agents. *arXiv preprint arXiv:2601.23014*, 2026.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. MemGen: Weaving generative latent memory for self-evolving agents. In *Proc. Int. Conf. Learn. Represent.*, 2026a.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, 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*, 2026b.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Trans. Inf. Syst.*, 43(6):1–47, 2025.

AI Use Statement

1. **Research design and scientific decisions.** The research idea, overall framework, methodology, experimental design, interpretation of results, and final conclusions were developed and determined by the authors.

2. **Implementation and manuscript preparation.** Generative AI tools were used for code implementation, to assist in preparing an initial draft of the manuscript, and to generate explanatory figures illustrating the proposed framework. The manuscript was subsequently reviewed and substantially revised through multiple iterations by the authors. Experimental result figures were produced by the authors from the experimental data using Python-based plotting scripts.

3. **Verification and responsibility.** All AI-assisted outputs, including code, text, and explanatory figures, were reviewed and validated by the authors. The authors verified the final implementation, experimental results, claims, and manuscript, and take full responsibility for the final content of this work.

A Theoretical Analysis of Trajectory-Grounded Memory Valuation

This section develops the theoretical analysis underlying trajectory-grounded memory valuation and value-based selection. We organize the analysis around three related questions. First, we characterize when the population observational value admits an interpretation as an average downstream marginal contribution. Second, we characterize the finite-sample evidence supporting the observed association between memory occurrence and downstream outcome. Third, we study the approximation introduced when individual memory values are used for set-level Top-K selection. Together, these analyses connect logged trajectory evidence to downstream memory selection at the levels of population interpretation, finite-sample evidence, and set-level approximation.

A.1 Population-Level Interpretation of Observational Memory Value

This subsection provides the proof of Proposition 3.1 and characterizes how differences between the trajectory contexts in which a memory is present and absent affect the interpretation of its observational value.

Setup. For a fixed query q, consider a candidate memory m. Let $B \subseteq \mathcal{M} \setminus \{m\}$ denote the set of remaining memory units associated with a trajectory. Recall that

$$F_q(S) = \mathbb{E}_{\tau \sim p(\tau|q,S)} \left[\tilde{R}(\tau) \right] \tag{A.1}$$

denotes the expected downstream utility under memory configuration S, and

$$\Delta_m(B \mid q) = F_q(B \cup \{m\}) - F_q(B) \tag{A.2}$$

denotes the context-specific marginal contribution of m.

Let $P_1^m(B \mid q)$ denote the distribution of the remaining memory context among logged trajectories in which m is present, and let $P_0^m(B \mid q)$ denote the corresponding distribution when m is absent.

To connect logged outcomes to the intervention-style quantities in Eq. (A.1), Proposition 3.1 assumes *contextual representativeness*: conditional on the remaining context B, the expected logged utilities with and without m agree with those induced by the corresponding memory configurations,

$$\begin{aligned} \mathbb{E}[\tilde{R} \mid B, c(m) = 1, q] &= F_q(B \cup \{m\}), \\ \mathbb{E}[\tilde{R} \mid B, c(m) = 0, q] &= F_q(B). \end{aligned} \tag{A.3}$$

Under this condition, the law of total expectation gives

$$\begin{aligned} V_{\inf}(m \mid q) &= \mathbb{E}[\tilde{R} \mid c(m) = 1, q] - \mathbb{E}[\tilde{R} \mid c(m) = 0, q] \\ &= \mathbb{E}_{B \sim P_1^m} \left[F_q(B \cup \{m\}) \right] - \mathbb{E}_{B \sim P_0^m} \left[F_q(B) \right]. \end{aligned} \tag{A.4}$$

Proof of Proposition 3.1. Suppose in addition that the presence and absence groups induce the same distribution over the remaining trajectory context:

$$P_1^m(B \mid q) = P_0^m(B \mid q) = P^m(B \mid q). \tag{A.5}$$

Substituting Eq. (A.5) into Eq. (A.4) gives

$$\begin{aligned} V_{\inf}(m \mid q) &= \mathbb{E}_{B \sim P^m} \left[F_q(B \cup \{m\}) - F_q(B) \right] \\ &= \mathbb{E}_{B \sim P^m} \left[\Delta_m(B \mid q) \right]. \end{aligned} \tag{A.6}$$

Therefore, under contextual representativeness and matched trajectory contexts, the observational presence–absence contrast equals the average context-specific marginal downstream contribution of m.

General case with context mismatch. Exact equality between P_1^m and P_0^m is not required to characterize the observational value. Adding and subtracting $\mathbb{E}_{B \sim P_1^m}[F_q(B)]$ in Eq. (A.4) yields

$$\begin{aligned} V_{\inf}(m \mid q) &= \mathbb{E}_{B \sim P_1^m} \left[F_q(B \cup \{m\}) - F_q(B) \right] \\ &\quad + \mathbb{E}_{B \sim P_1^m} \left[F_q(B) \right] - \mathbb{E}_{B \sim P_0^m} \left[F_q(B) \right] \\ &= \mathbb{E}_{B \sim P_1^m} \left[\Delta_m(B \mid q) \right] + \mathcal{B}_m(q), \end{aligned} \tag{A.7}$$

where

$$\mathcal{B}_m(q) = \mathbb{E}_{B\sim P_1^m}\left[F_q(B)\right] - \mathbb{E}_{B\sim P_0^m}\left[F_q(B)\right] \tag{A.8}$$

captures the shift between the surrounding trajectory contexts represented in the two groups.

Equation (A.7) therefore separates two effects contained in the observational contrast. The first is the average marginal contribution of m over the contexts in which it is observed. The second arises when trajectories containing m systematically differ in their surrounding memory configurations from trajectories in which m is absent.

If $F_q(S) \in [0,1]$, the context-shift term satisfies

$$|\mathcal{B}_m(q)| \le \mathrm{TV}\left(P_1^m(\cdot \mid q), P_0^m(\cdot \mid q)\right), \tag{A.9}$$

where TV denotes total-variation distance. Hence, increasingly comparable presence and absence contexts imply an increasingly accurate average-marginal interpretation of V_{inf}. Importantly, this discrepancy is an identification issue induced by the logging distribution rather than a finite-sample evidence issue. Increasing the number of logged trajectories does not by itself remove systematic context mismatch.

Binary-outcome interpretation. Consider the special case of deterministic binary downstream outcomes, $F_q(S) \in \{0,1\}$. Suppose further that memory inclusion is monotone in the considered context,

$$F_q(B \cup \{m\}) \ge F_q(B). \tag{A.10}$$

Then $\Delta_m(B \mid q) \in \{0,1\}$. In particular,

$$\Delta_m(B \mid q) = 1 \quad \Longleftrightarrow \quad F_q(B \cup \{m\}) = 1 \ \text{ and } \ F_q(B) = 0. \tag{A.11}$$

That is, m is *pivotal* in context B: its inclusion changes the downstream outcome from failure to success.

Under the matched-context condition of Proposition 3.1,

$$V_{\mathrm{inf}}(m \mid q) = \Pr_B\left[m \text{ is pivotal in context } B\right]. \tag{A.12}$$

Thus, in this special case, the population observational value directly measures how frequently a memory changes the downstream outcome across trajectory contexts.

For stochastic binary tasks, $F_q(S)$ instead represents a success probability and $\Delta_m(B \mid q)$ represents the corresponding change in success probability. The pivotality interpretation should therefore be understood as a deterministic special case of the more general average marginal-contribution result.

Necessary, replaceable, and non-pivotal memories. The preceding result also clarifies how different memory roles are reflected by the observational value.

A memory may be necessary for a successful execution without being pivotal in every logged context. For example, if other requirements for success are absent from B, adding m alone may still leave the task unsuccessful. A necessary memory therefore becomes pivotal specifically in contexts where the remaining configuration is otherwise sufficient for success, and its population value depends on how frequently such contexts occur in the logging distribution. Thus, necessity does not by itself imply that V_{inf} must be close to one.

A memory may instead be useful but replaceable. If alternative memories or execution patterns already make a context successful without m, then

$$F_q(B \cup \{m\}) = F_q(B),$$

and the marginal contribution of m in that context is zero. As such alternative contexts become more common, the average marginal value of m naturally decreases.

At the other extreme, if the downstream utility never changes when m is added, $\Delta_m(B \mid q) = 0$ for every relevant context, then its observational value is zero under matched contexts. Outside the monotone special case, a memory may also reduce downstream utility in some contexts, producing negative context-specific contributions and potentially a negative observational value.

This interpretation does not require a task to admit a unique reasoning or execution path. Multiple successful trajectory configurations are naturally represented through the distribution over B. What matters for the value of m is not whether it belongs to a particular successful path, but how often its inclusion changes downstream utility across the trajectory contexts represented by the logging process.

Relation to the ideal counterfactual value. The analysis above characterizes the population observational quantity $V_{\text{inf}}(m \mid q)$ as an average context-specific marginal contribution under explicit logging conditions. When $\tilde{R} = R$ and the intervention represented by comparing $F_q(B \cup \{m\})$ with $F_q(B)$ matches the remove-m counterfactual used in Eq. (2), this marginal interpretation directly aligns with the ideal decision-centric target. Exact equality with $V^\star(m \mid q)$ additionally depends on whether the distribution of trajectory contexts used by the observational analysis matches the context distribution induced by the counterfactual definition.

The remaining theoretical questions concern different stages of the selection pipeline. Finite logged data determine how strongly the observed memory–outcome association is supported by the available trajectory evidence; this is analyzed in Appendix A.2. Separately, using individual memory values to approximate the utility of an entire selected set introduces a set-level approximation, which is analyzed in Appendix A.3.

A.2 Valuation Reliability Assessment under Binary Trajectory Outcomes

The preceding analysis characterizes when the population observational value $V_{\text{inf}}(m \mid q)$ admits a downstream marginal-contribution interpretation. We now consider a complementary practical question: *how much trajectory evidence is enough to trust an estimated memory value?* To address this question, we introduce *Valuation Reliability Assessment (VRA)*, which quantifies how strongly the available trajectory evidence supports the observed memory valuation.

We focus on binary trajectory outcomes, $\tilde{R} \in \{0, 1\}$, as in tasks whose executions are ultimately evaluated by answer correctness, execution success, or task completion. For the analysis below, we assume that the logged trajectories are conditionally independent samples from a fixed logging distribution given q. Each trajectory contributes one of four possible memory–outcome observations for a candidate memory m:

$$\begin{array}{c|cc} & \tilde{R}=1 & \tilde{R}=0 \\ \hline c(m)=1 & n_{11}(m) & n_{10}(m) \\ c(m)=0 & n_{01}(m) & n_{00}(m) \end{array}. \tag{A.13}$$

Here, $n_{11}(m)$ counts successful trajectories containing m, $n_{10}(m)$ counts failed trajectories containing m, $n_{01}(m)$ counts successful trajectories without m, and $n_{00}(m)$ counts failed trajectories without m. Let $N_\tau = n_{11} + n_{10} + n_{01} + n_{00}$, where the dependence on m is omitted when unambiguous.

For binary outcomes, Eq. (3) can be written directly from these four counts as

$$\widehat{V}_{\text{am}}(m \mid q) = \frac{n_{11}}{n_{11} + n_{10}} - \frac{n_{01}}{n_{01} + n_{00}}, \tag{A.14}$$

assuming that both occurrence groups are observed. Thus, the same trajectory table determines both the estimated memory value and the evidence used to assess the reliability of that valuation.

Fisher exact-test basis for VRA. Consider the null hypothesis

$$H_0 : \qquad c(m) \;\perp\; \tilde{R} \mid q, \tag{A.15}$$

under which memory occurrence and trajectory success are independent for the current query or task context. For binary outcomes and positive occurrence–absence coverage, Eq. (A.15) is equivalent to

$$V_{\text{inf}}(m \mid q) = 0. \tag{A.16}$$

Hence, testing Eq. (A.15) evaluates whether the available trajectory evidence supports a positive population occurrence–outcome association.

Conditioned on the observed row and column margins in Eq. (A.13), the number of successful trajectories containing m follows a hypergeometric distribution under H_0. This corresponds to

Fisher's exact test (Fisher, 1935). Let $n_{1\cdot} = n_{11} + n_{10}$, $n_{0\cdot} = n_{01} + n_{00}$, and $n_{\cdot 1} = n_{11} + n_{01}$. For a positive observed association, the corresponding one-sided exact probability can be written in our notation as

$$p_{\mathrm{ex}}(m) = \sum_{x=n_{11}}^{\min(n_{1\cdot},n_{\cdot 1})} \frac{\binom{n_{1\cdot}}{x}\binom{n_{0\cdot}}{n_{\cdot 1}-x}}{\binom{N_\tau}{n_{\cdot 1}}}. \tag{A.17}$$

This is the exact conditional probability, under Eq. (A.15), of observing an association at least as strong as the current one. Smaller $p_{\mathrm{ex}}(m)$ therefore indicates stronger statistical evidence against the hypothesis that memory occurrence and trajectory success are unrelated.

Based on this exact probability, we define the *Valuation Reliability Assessment (VRA)* as

$$\mathrm{VRA}(m \mid q; \mathcal{D}_q) \triangleq 1 - p_{\mathrm{ex}}(m). \tag{A.18}$$

A larger $\mathrm{VRA}(m \mid q; \mathcal{D}_q)$ indicates stronger trajectory evidence supporting the observed positive memory–outcome association and hence stronger support for the resulting valuation. Importantly, VRA is an evidence-support quantity rather than a posterior probability that the estimated memory value is correct. The estimated value $\widehat{V}_{\mathrm{am}}$ and VRA therefore serve complementary roles: $\widehat{V}_{\mathrm{am}}$ determines the magnitude and ranking of memory value, whereas VRA assesses how strongly the available trajectory evidence supports that valuation.

Zero-violation case. An especially informative trajectory pattern occurs when

$$n_{01} = 0, \tag{A.19}$$

that is, no successful trajectory in the logged collection is observed without memory m. This condition does not require the presence of m to be sufficient for success: failed trajectories may still contain the memory, so n_{10} may be positive. It only states that the observed successful trajectories consistently contain m.

Under Eq. (A.19), we have $n_{\cdot 1} = n_{11}$. The observed count n_{11} is then the largest attainable number of successful trajectories containing m under the fixed margins, so the tail in Eq. (A.17) contains a single term:

$$p_{\mathrm{ex}}(m) = \frac{\binom{n_{11}+n_{10}}{n_{11}}}{\binom{N_\tau}{n_{11}}}. \tag{A.20}$$

Accordingly,

$$\mathrm{VRA}(m \mid q; \mathcal{D}_q) = 1 - \frac{\binom{n_{11}+n_{10}}{n_{11}}}{\binom{N_\tau}{n_{11}}}. \tag{A.21}$$

This gives the zero-violation expression stated in Proposition 3.2.

How trajectory evidence affects VRA. The zero-violation expression provides a direct combinatorial interpretation of valuation reliability. Conditional on the observed number of successful trajectories, the denominator $\binom{N_\tau}{n_{11}}$ counts the possible placements of these successes among all logged trajectories under the independence null, whereas the numerator $\binom{n_{11}+n_{10}}{n_{11}}$ counts the placements in which every success falls inside the subset of trajectories containing m. Repeated observations of the same outcome-consistent occurrence pattern therefore make such an association increasingly difficult to explain under the independence null and consequently increase VRA.

Several monotonicity properties make this behavior explicit. First, holding n_{10} and n_{00} fixed, adding an additional successful trajectory containing m strictly decreases the exact probability:

$$\frac{p_{\mathrm{ex}}(n_{11}+1, n_{10}, n_{00})}{p_{\mathrm{ex}}(n_{11}, n_{10}, n_{00})} = \frac{n_{11}+n_{10}+1}{n_{11}+n_{10}+n_{00}+1} < 1. \tag{A.22}$$

Hence, repeated successful observations containing m increase VRA, provided that informative memory-absence evidence remains available.

Second, holding n_{11} and n_{10} fixed, adding a failed trajectory in which the memory is absent also decreases the exact probability:

$$\frac{p_{\mathrm{ex}}(n_{11}, n_{10}, n_{00}+1)}{p_{\mathrm{ex}}(n_{11}, n_{10}, n_{00})} = \frac{n_{10}+n_{00}+1}{n_{11}+n_{10}+n_{00}+1} < 1. \tag{A.23}$$

Thus, VRA is strengthened not only by successful trajectories containing m, but also by failed trajectories in which m is absent.

Third, holding n_{11} and n_{00} fixed, increasing the number of failed trajectories that nevertheless contain m weakens the observed association. For $n_{11}, n_{00} > 0$,

$$\frac{p_{\text{ex}}(n_{11}, n_{10}+1, n_{00})}{p_{\text{ex}}(n_{11}, n_{10}, n_{00})} = \frac{(n_{11}+n_{10}+1)(n_{10}+n_{00}+1)}{(n_{10}+1)(n_{11}+n_{10}+n_{00}+1)} > 1. \tag{A.24}$$

Such observations reduce VRA because they weaken the evidence that the occurrence of m distinguishes successful from unsuccessful executions.

A useful limiting case follows immediately. If $n_{00} = 0$, then under the zero-violation condition every logged trajectory contains m, yielding $p_{\text{ex}}(m) = 1$ and therefore

$$\text{VRA}(m \mid q; \mathcal{D}_q) = 0. \tag{A.25}$$

Although $n_{01} = 0$ still holds trivially, the logged data contain no memory-absence evidence. VRA therefore provides no statistical support based solely on the fact that all successful trajectories contain m. This degenerate case illustrates why occurrence–absence variation, rather than trajectory count alone, is essential for reliable valuation.

Trajectory sufficiency and evidence accumulation. The preceding results connect trajectory coverage directly to valuation reliability. Let

$$q_m = \frac{n_{11}+n_{10}}{N_\tau} \tag{A.26}$$

denote the empirical occurrence rate of memory m. Under the zero-violation condition, Eq. (A.20) can be written as

$$p_{\text{ex}}(m) = \prod_{i=0}^{n_{11}-1} \frac{n_{11}+n_{10}-i}{N_\tau - i}. \tag{A.27}$$

Since $n_{11} + n_{10} \le N_\tau$, each factor satisfies

$$\frac{n_{11}+n_{10}-i}{N_\tau - i} \le \frac{n_{11}+n_{10}}{N_\tau} = q_m. \tag{A.28}$$

Therefore,

$$\text{VRA}(m \mid q; \mathcal{D}_q) \ge 1 - q_m^{n_{11}}. \tag{A.29}$$

This is the geometric lower bound stated in Proposition 3.2.

The bound makes the connection between trajectory quantity and valuation reliability explicit. For a fixed observed occurrence rate $q_m \in (0,1)$ and a target VRA level $\rho \in (0,1)$,

$$n_{11} \ge \frac{\log(1-\rho)}{\log q_m} \quad \Longrightarrow \quad \text{VRA}(m \mid q; \mathcal{D}_q) \ge \rho. \tag{A.30}$$

Thus, in the zero-violation regime, Eq. (A.30) provides a direct quantitative relation between outcome-consistent trajectory evidence and a desired VRA level. This relation should not be interpreted as a universal prospective sample-size requirement, because the empirical occurrence rate q_m and the full occurrence–outcome composition may themselves change as additional trajectories are observed.

A stronger asymptotic statement follows when occurrence coverage remains nondegenerate along an expanding trajectory collection. If there exists some constant $\bar{q}_m < 1$ such that $q_m \le \bar{q}_m$ throughout the considered sequence of trajectory collections, then

$$\text{VRA}(m \mid q; \mathcal{D}_q) \ge 1 - \bar{q}_m^{n_{11}}. \tag{A.31}$$

Hence, under persistent occurrence–absence coverage, the lower bound on VRA approaches one geometrically as successful trajectories consistently containing m accumulate.

These results do not imply a universal trajectory-count threshold. The reliability provided by a given trajectory budget depends jointly on memory occurrence coverage and outcome composition. The same number of trajectories may therefore yield high VRA when the occurrence–outcome structure is well resolved, but low VRA when the memory appears almost universally or when successful trajectories frequently occur without it.

Complementary roles of memory value and VRA. The trajectory-grounded value estimate and VRA answer two different questions. For binary outcomes, Eq. (A.14) measures the magnitude and direction of the observed difference in success probability:

$$\widehat{V}_{\mathrm{am}}(m \mid q) = \widehat{P}(\tilde{R} = 1 \mid c(m) = 1, q) - \widehat{P}(\tilde{R} = 1 \mid c(m) = 0, q). \tag{A.32}$$

In contrast, VRA quantifies how strongly the available trajectory evidence supports the observed positive memory–outcome association underlying that valuation.

A memory may therefore receive a large point-estimated value while having low VRA when only a small number of informative trajectories are available. Conversely, additional trajectories that preserve the same occurrence–outcome association can substantially increase VRA without changing the underlying definition of memory value or the direct Top-K selection rule. Thus, $\widehat{V}_{\mathrm{am}}$ determines which memories are ranked highly, whereas VRA quantifies the evidence supporting the observed positive memory–outcome association.

This distinction is especially important when interpreting trajectory budget. Increasing the number of logged trajectories is useful not merely because it increases the raw sample count, but because it can provide additional outcome-consistent evidence while preserving informative variation between memory-present and memory-absent trajectories. VRA therefore provides a quantitative way to assess whether the currently available trajectory evidence already offers strong support for the observed valuation or whether additional evidence may still be informative.

General observations beyond the zero-violation case. The simplified expressions above apply only when $n_{01} = 0$. When successful trajectories without m are observed, the general exact probability in Eq. (A.17) remains applicable, and VRA is computed directly as $1 - p_{\mathrm{ex}}(m)$. In this case, VRA reflects the extent to which successful trajectories without m weaken the observed positive association. The reliability analysis therefore does not require every useful memory to follow a strict necessary-condition pattern; the zero-violation case is instead a useful regime in which the dependence of VRA on trajectory quantity and occurrence coverage becomes analytically transparent.

Scope of the analysis. VRA quantifies the statistical support provided by the available logged trajectories for an observed memory–outcome association. It does not replace the population-level interpretation of V_{inf}, which depends on the contextual representativeness and trajectory-context comparability conditions analyzed in Appendix A.1. Likewise, strong VRA for individual memories does not eliminate the approximation introduced when individual values are combined for direct Top-K selection; residual cross-memory interactions are analyzed separately in Appendix A.3. These analyses therefore address three distinct aspects of the framework: population-level interpretation, trajectory-evidence support for valuation, and set-level approximation.

A.3 Set-Level Approximation of Value-Based Top-K Selection

The main-text selector uses individual population observational values as a first-order surrogate for set-level downstream utility. This subsection characterizes the approximation induced by that step. The analysis does not assume that memory units are strictly independent; instead, it makes the residual between the additive value-based surrogate and the true set-level utility explicit.

Throughout this subsection, we use the same task-aligned utility $\tilde{R}$ as in the preceding analysis:

$$F_q(S) = \mathbb{E}_{\tau \sim p(\tau \mid q, S)} \left[\tilde{R}(\tau) \right]. \tag{A.33}$$

When $\tilde{R} = R$, this is directly the original downstream task objective. When a denser surrogate is used, the analysis applies to the corresponding surrogate objective, while alignment between $\tilde{R}$ and the terminal reward R remains a separate modeling consideration.

General set-level residual. The population first-order surrogate used in the main text is

$$U^{(1)}_{\mathrm{occ}}(S \mid q) = \sum_{m \in S} V_{\mathrm{inf}}(m \mid q). \tag{A.34}$$

We define the residual relative to the true set-level utility as

$$\Xi_q(S) = F_q(S) - F_q(\emptyset) - \sum_{m \in S} V_{\inf}(m \mid q). \tag{A.35}$$

Equivalently,

$$F_q(S) = F_q(\emptyset) + \sum_{m \in S} V_{\inf}(m \mid q) + \Xi_q(S). \tag{A.36}$$

This decomposition is exact by construction. The term $\Xi_q(S)$ collects whatever set-level structure is not represented by simply summing the individual population observational values. It therefore provides the relevant residual for analyzing the actual value-based Top-K selector used in the main text.

Let

$$S_{\mathrm{occ}} = \arg \max_{\substack{S \subseteq \mathcal{M} \\ |S|=K}} \sum_{m \in S} V_{\inf}(m \mid q) \tag{A.37}$$

denote the population Top-K set induced by the individual observational values, and let

$$S^\star = \arg \max_{\substack{S \subseteq \mathcal{M} \\ |S|=K}} F_q(S) \tag{A.38}$$

denote the optimal subset under the true set-level utility.

Proposition A.1 (Bounded regret of value-based Top-K selection)**.** Suppose that, for all feasible subsets S with $|S| = K$,

$$|\Xi_q(S)| \le \epsilon_K. \tag{A.39}$$

Then

$$F_q(S^\star) - F_q(S_{\mathrm{occ}}) \le 2\epsilon_K. \tag{A.40}$$

Proof. Using Eq. (A.36),

$$\begin{aligned} F_q(S^\star) - F_q(S_{\mathrm{occ}}) = &\sum_{m \in S^\star} V_{\inf}(m \mid q) - \sum_{m \in S_{\mathrm{occ}}} V_{\inf}(m \mid q) \\ &+ \Xi_q(S^\star) - \Xi_q(S_{\mathrm{occ}}). \end{aligned} \tag{A.41}$$

By definition, S_{occ} maximizes the additive population-value objective, and therefore

$$\sum_{m \in S^\star} V_{\inf}(m \mid q) - \sum_{m \in S_{\mathrm{occ}}} V_{\inf}(m \mid q) \le 0.$$

Together with Eq. (A.39),

$$\Xi_q(S^\star) - \Xi_q(S_{\mathrm{occ}}) \le 2\epsilon_K,$$

which proves Eq. (A.40).

Proposition A.1 directly characterizes the population selector used in the main text. The first-order surrogate need not equal the true set-level utility exactly; rather, the regret is controlled whenever the residual $\Xi_q(S)$ remains bounded over feasible subsets.

Relation to conventional cross-memory interactions. To clarify what contributes to $\Xi_q(S)$, define the singleton contribution

$$v_q(m) = F_q(\{m\}) - F_q(\emptyset) \tag{A.42}$$

and the conventional non-additive interaction residual

$$\Gamma_q(S) = F_q(S) - F_q(\emptyset) - \sum_{m \in S} v_q(m). \tag{A.43}$$

Also define

$$d_q(m) = v_q(m) - V_{\inf}(m \mid q). \tag{A.44}$$

Then

$$\Xi_q(S) = \sum_{m \in S} d_q(m) + \Gamma_q(S). \tag{A.45}$$

Equation (A.45) makes two distinct effects explicit. The term $d_q(m)$ captures the difference between the context-averaged observational value used by VAM and the marginal contribution measured specifically relative to the empty-memory context. This difference is not a finite-sample evidence issue: even with arbitrarily strong finite-sample support for the observational association, V_{inf} may differ from $v_q(m)$ because the former averages marginal effects over the trajectory contexts represented by the logging process.

The term $\Gamma_q(S)$ captures genuine cross-memory non-additivity beyond singleton effects. It may be positive when memories exhibit complementarity, or negative when they exhibit redundancy, diminishing returns, or interference.

Thus, the residual relevant to practical value-based selection may reflect both the context over which individual marginal values are summarized and interactions among the memories selected together.

Pairwise interaction special case. For two memories m_i and m_j, define the conventional pairwise interaction

$$\Delta_q(m_i, m_j) = F_q(\{m_i, m_j\}) - F_q(\{m_i\}) - F_q(\{m_j\}) + F_q(\emptyset). \tag{A.46}$$

Here, $\Delta_q(m_i, m_j) > 0$ indicates super-additive interaction, while $\Delta_q(m_i, m_j) < 0$ indicates sub-additive interaction such as redundancy, diminishing returns, or interference.

Suppose that

$$\Gamma_q(S) = \sum_{\substack{m_i, m_j \in S \\ i<j}} \Delta_q(m_i, m_j), \tag{A.47}$$

and that

$$|d_q(m)| \le \rho, \qquad |\Delta_q(m_i, m_j)| \le \eta. \tag{A.48}$$

Then, for every subset S with $|S| = K$,

$$\begin{aligned} |\Xi_q(S)| &\le \sum_{m \in S} |d_q(m)| + \sum_{\substack{m_i, m_j \in S \\ i<j}} |\Delta_q(m_i, m_j)| \\ &\le K\rho + \binom{K}{2}\eta. \end{aligned} \tag{A.49}$$

Applying Proposition A.1 yields

$$F_q(S^\star) - F_q(S_{\mathrm{occ}}) \le 2K\rho + K(K-1)\eta. \tag{A.50}$$

This special case separates the effect of context-averaged individual valuation from the effect of pairwise non-additivity. When the observational value coincides with the singleton contribution, $\rho = 0$, and Eq. (A.50) reduces to

$$F_q(S^\star) - F_q(S_{\mathrm{occ}}) \le K(K-1)\eta.$$

The result also illustrates why small memory budgets help limit the impact of bounded pairwise interactions. Explicitly modeling pairwise or higher-order effects may improve set-level optimality, but doing so would require substantially more interaction estimates and potentially combinatorial selection, conflicting with the low-overhead objective of the framework.

Memory granularity and composite units. Strong complementarity may partly arise from the granularity at which memory units are defined. For example, two memory fragments may individually provide incomplete parts of a procedural or multi-hop reasoning pattern, while their combination forms a coherent and reusable reasoning unit. In such cases, naturally coupled fragments can instead be represented as a composite memory unit.

Formally, suppose a collection of strongly interacting atomic memories is grouped into

$$G_j = \{m_{j,1}, \ldots, m_{j,r_j}\}, \tag{A.51}$$

and represented as a composite memory

$$\widetilde{m}_j := G_j. \tag{A.52}$$

If the dominant interactions occur within each group and cross-group interactions are small, then representing each G_j as one unit can absorb part of the interaction structure that would otherwise appear in $\Gamma_q(S)$.

In the limiting case where the composite units are mutually additive under the relevant task utility, the interaction component across those units vanishes. This observation highlights that complementarity is partly relative to the chosen memory representation: a memory unit need not correspond to a raw text fragment or a single transition, but may instead represent a higher-level fact, trajectory segment, procedure, or reusable skill.

We emphasize that this analysis does not assume that arbitrary interacting memories can be discovered and merged without cost. Composite-unit construction is an upstream memory-representation choice. The valuation framework operates on the resulting candidate units, while residual context dependence and cross-unit interactions determine the remaining set-level approximation.

Relation to the preceding analysis. The theoretical results above concern three different stages of trajectory-grounded value-based selection. Appendix A.1 characterizes the population interpretation of the observational value and the role of trajectory-context comparability. Appendix A.2 characterizes the finite-sample evidence supporting the observed memory–outcome association. The present subsection characterizes the additional approximation induced when individual population values are used to construct a set-level Top-K decision.

These questions are conceptually distinct. Additional logged trajectories can strengthen finite-sample evidence when they consistently reinforce the observed association, but they do not by themselves remove systematic context mismatch in the population interpretation or residual set-level non-additivity. Conversely, small cross-memory interaction does not resolve weak finite-sample support for individual memory–outcome associations. Together, the three analyses specify the distinct conditions under which trajectory-grounded memory values can support reliable low-overhead selection.

B Supplementary Experiment and Empirical Analyses

This section provides supplementary analyses of three aspects that can affect direct value-based Top-K selection: the memory budget, candidate-memory redundancy, and cross-memory interaction.

B.1 Sensitivity to the Memory Budget

The memory budget K is chosen to provide sufficient task-relevant context while keeping the selected memory set compact, and therefore need not be identical across benchmarks with different task structures. For example, each candidate pool in our HotpotQA evaluation contains two gold supporting paragraphs, making $K = 2$ a natural starting point; using only a single memory may under-provide the evidence required for multi-hop reasoning. In ALFWorld, tasks generally involve longer multi-step interaction sequences, motivating a slightly larger memory budget such as $K = 3$. These choices are intended as practical defaults rather than assumptions that a particular value of K is universally optimal. We therefore additionally evaluate VAM over multiple memory budgets to examine how sensitive downstream performance and pipeline cost are to the number of selected memories. For the main comparison, however, we keep a fixed memory budget for all selection methods within each benchmark rather than tuning K separately for VAM.

Figure 5 shows that the effect of the memory budget is task dependent. On ToolBench, performance improves from $K = 1$ to $K = 4$, while total input token consumption decreases from 10.3K to 9.07K. The main comparison uses the same fixed memory budget across selectors within each benchmark, whereas this analysis examines VAM's sensitivity to K. On HotpotQA, the largest performance gain occurs from $K = 1$ to $K = 2$, but additional memories continue to improve Joint F1 with smaller increments, while total input tokens increase from 380 to 719. ALFWorld also shows diminishing performance gains, with success increasing from 97.0% at $K = 3$ to 98.5% at $K = 4$. Its total input token consumption decreases from 11.4K at $K = 1$ to 8.64K at $K = 4$, with little change beyond $K = 3$. Overall, larger memory budgets improve performance across the tested range, with diminishing gains. Their effect on end-to-end input token consumption varies by task: additional memories increase cost on HotpotQA but reduce it on ToolBench and ALFWorld.

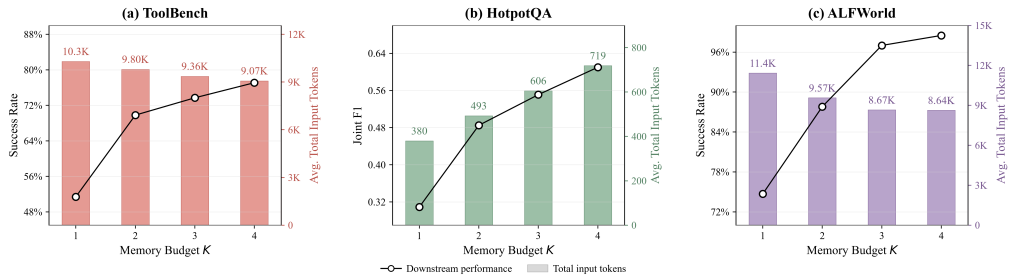

Figure 5: Effect of memory budget K on downstream performance (lines) and total input token consumption (bars) across the three benchmarks using VAM.

B.2 Effect of Candidate-Memory Overlap

To study the effect of candidate redundancy, we construct controlled overlap settings on ALFWorld by injecting functionally equivalent action memories with different surface action names. Specifically, an injected memory preserves the same or highly similar task context, pre-action observation, and resulting observation as an existing memory, while using an alternative action expression that produces the same functional transition. Since an ALFWorld memory unit is represented by its state–action–outcome context rather than the action string alone, these variants create transition-level redundancy without being exact textual duplicates.

We consider the original candidate pool and two increasingly redundant settings with 40% and 80% injected overlap, and compare direct value-based Top-K selection with diversity-aware reranking.

Table 4: Effect of candidate-memory overlap on ALFWorld. We compare direct Top-K selection with optional diversity-aware reranking under the original candidate pool and two controlled overlap settings. For each setting, we report the downstream success rate and the average pairwise semantic similarity among selected memories.

Selection Method	**Original (0% Injected Overlap)**		**Moderate (40%)**		**High (80%)**	
	Success ↑	**Avg. Sim. ↓**	**Success ↑**	**Avg. Sim. ↓**	**Success ↑**	**Avg. Sim. ↓**
Direct Top-K	**97.0%**	0.18	84.0%	0.43	76.0%	0.67
Top-K + Diversity Reranking	**97.0%**	**0.16**	**88.0%**	**0.25**	**85.0%**	**0.31**

Table 4 provides a supplementary analysis of candidate-memory overlap. Under the original candidate pool, direct Top-K and diversity-aware reranking achieve the same downstream success rate, while both select memories with relatively low semantic similarity. As overlap increases, direct Top-K increasingly selects similar memories and its downstream performance degrades. A lightweight similarity-based reranking substantially reduces the redundancy and recovers part of the lost performance, particularly under the high-overlap setting.

These results suggest that direct value-based Top-K is sufficient when the candidate memory pool is reasonably diverse, while diversity-aware reranking can be optionally introduced when substantial candidate redundancy is present.

B.3 Empirical Analysis of Pairwise Memory Interactions

We next empirically examine the cross-memory interaction component discussed in Appendix A.3. For this analysis, we construct a separate exhaustive evaluation set on HotpotQA. Unlike the logged trajectory collection used to estimate VAM scores, this evaluation set is generated solely for empirical validation by enumerating all singleton and pairwise subsets of the 10 candidate paragraphs for each question.

For each configuration, the selected paragraphs are provided to the same downstream QA model used in the main experiments, and the resulting downstream reward is recorded. Let $R_q(S)$ denote the official HotpotQA supporting-fact EM score for query q under memory set S.

For each question q, the candidate pool contains $N_q = 10$ paragraphs: two gold supporting paragraphs and eight distractor paragraphs. We exhaustively evaluate all N_q singleton configurations and $\binom{N_q}{2}$ pairwise configurations. Each question therefore contributes 10 singleton and 45 pairwise configurations, giving 1,000 singleton and 4,500 pairwise configurations across the 100 questions.

Empirical decision-value reference. For a memory m_i, we estimate its empirical decision value by averaging its marginal contribution over all other single-memory contexts:

$$V_{\text{emp}}(m_i \mid q) = \frac{1}{N-1} \sum_{j \neq i} \left[R_q(\{m_i, m_j\}) - R_q(\{m_j\}) \right], \tag{B.1}$$

where N is the number of candidate memories for the task. The candidate memories are ranked according to V_{emp}, and the resulting Top-2 set is used as the empirical decision-value reference in Sec. 4.2. Ties in empirical memory values are broken by a fixed candidate order.

Mean Value Regret@2 is computed as the difference between the summed empirical values of the empirical Top-2 set and those of the Top-2 set selected by each method. Thus, the exhaustive evaluation provides a direct downstream-utility-based reference for assessing both selection agreement and the severity of selection errors.

Empirical pairwise interaction. For a query q and a memory pair (m_i, m_j), define

$$\Delta_q^{\text{emp}}(m_i, m_j) = R_q(\{m_i, m_j\}) - R_q(\{m_i\}) - R_q(\{m_j\}) + R_q(\emptyset). \tag{B.2}$$

The condition $\Delta_q^{\text{emp}}(m_i, m_j) = 0$ indicates additive behavior with respect to the singleton utilities. A non-zero value indicates that the joint utility cannot be fully explained by the two singleton contributions. $R_q(\varnothing)$ is taken from the corresponding no-memory evaluation using the same reward metric. Positive and negative values correspond to super-additive and sub-additive effects, respectively. Because the sign may arise from different mechanisms, including complementarity, redundancy, diminishing returns, or interference, we primarily examine the prevalence and magnitude of non-additivity through $|\Delta^{\text{emp}}|$.

Table 5: Overall empirical pairwise interaction statistics on HotpotQA.

Dataset	Median Δ	Median $\lvert\Delta\rvert$	Non-zero (%)
HotpotQA	0.0000	0.0000	14.0

As shown in Table 5, pairwise interactions are relatively sparse across the full candidate space. In particular, 86.0% of all memory pairs exhibit zero measured interaction, resulting in both a zero median interaction and a zero median interaction magnitude. Thus, pairwise interaction is not pervasive across arbitrary candidate combinations, although it may be important for particular memory pairs.

To examine whether these interactions are structured, we further divide the memory pairs according to the relevance of their constituent memories.

Table 6: Pairwise interaction statistics by memory-pair type on HotpotQA.

Pair Type	# Pairs	Non-zero (%)
Gold–Gold	100	**70.0**
Gold–Distractor	1,600	24.4
Distractor–Distractor	2,800	6.0

Table 6 shows that pairwise non-additivity is concentrated among pairs containing gold supporting paragraphs. Non-zero interactions occur in 70.0% of Gold–Gold pairs, 24.4% of Gold–Distractor pairs, and only 6.0% of Distractor–Distractor pairs.

Overall, the empirical results suggest that the cross-memory interaction component is sparse globally but concentrated among tightly coupled, task-relevant memories. This observation is consistent with the set-level analysis in Appendix A.3: individual-value selection provides a useful first-order approximation across much of the candidate space, while localized non-additive effects can remain important for particular memory combinations.

Importantly, the pairwise statistic above diagnoses the conventional interaction component associated with singleton utilities. It does not by itself measure the entire residual $\Xi_q(S)$ in Eq. (A.35), which may also reflect the difference between context-averaged observational values and empty-context singleton contributions.

C Detailed Experimental Settings

C.1 Benchmark and Memory-Pool Construction

ToolBench trajectory and memory construction. For ToolBench, we curate 43 tasks from the StableToolBench solvable-query pool, retaining tasks with at least five archived trajectories, both successful and failed executions required for VAM estimation, at least two deduplicated candidate memory units, and valid offline API/query coverage. A ToolBench memory unit is constructed from one tool-call transition and contains the task goal, the preceding tool-interaction context, the current API call and argument pattern, and the observed outcome. Two simplified examples are shown below:

Example 1:

```
Goal: Find the contact information of a specified person.
Previous action: none
Previous observation: none
Current action: search_contact(name)
Argument pattern: [name]
Observed outcome: A non-empty structured response
containing the matched contact.
```

Example 2:

```
Goal: Retrieve available travel options and complete a
booking.
Previous action: search_options
Previous observation: Several candidate options were
returned.
Current action: create_booking(option_id, date)
Argument pattern: [date, option_id]
Observed outcome: The booking request was successfully
confirmed.
```

Each trajectory is represented as an ordered sequence of such tool-call transitions, and the set of memory units appearing in trajectory τ_i defines its memory subset A_i. Terminal `Finish` actions and units containing explicit final-answer information are excluded from the candidate pool to prevent answer leakage. For trajectory-grounded valuation, a trajectory is labeled successful if and only if all task-specific sub-goals are satisfied according to the deterministic evaluator; otherwise, it is labeled failed. The same success criterion is used for downstream evaluation. Tables 7 and 8 summarize the source composition and the resulting trajectory statistics.

HotpotQA trajectory and memory construction. We construct the HotpotQA trajectory evidence from the development distractor split and retain 100 bridge-type questions that require multi-hop evidence chaining. Each candidate memory unit consists of the title and textual content of one context paragraph. Each of the 100 selected questions contains exactly ten candidate memories: two gold paragraphs containing the annotated supporting facts and eight distractor paragraphs. The memory-selection budget is fixed to $K = 2$ in the main comparison. Across these 100 questions, the candidate pools contain 1,000 memory units in total, comprising 200 gold paragraphs and 800 distractor paragraphs. For each question, we collect trajectories by interacting with the answering agent under different retrieved paragraph subsets. We collect 8–12 trajectories per question, including both successful and failed attempts, and record the paragraphs used together with the resulting task outcome. Logged trajectories are labeled successful if their official HotpotQA supporting-fact EM score equals 1, and failed otherwise. These trajectories provide the presence–absence variation used by the trajectory-grounded value estimator. During final evaluation, the answering model receives only the selected paragraph contents and is evaluated using the standard HotpotQA metrics reported in the main text. Two simplified memory examples are shown below:

Table 7: ToolBench trajectory sources.

Agent configuration	Trajectories
claude_dfs	77
davinci_dfs	74
toolllama_dfs_retriever	65
toolllama_dfs	56
gpt-4-0613_dfs	54
chatgpt_dfs	53
claude_cot	32
davinci_cot	33
chatgpt_cot	31
gpt-4-0613_cot	31
toolllama_cot	31
GLM-5.2_cot	96
Total	**633**

Table 8: Aggregate statistics of the ToolBench trajectory collection.

Statistic	Value
Total trajectories	633
Unique tasks	43
Success trajectories	346
Failure trajectories	287
Trajectories per task	5 / 14.7 / 20
Steps per trajectory	1 / 3.3 / 6

For the final two rows, values denote minimum / mean / maximum.

Example 1:

```
Title: Person A
Paragraph: Person A was a Canadian scientist who studied at
University B and later became known for work in wireless
communications.
```

Example 2:

```
Title: University B
Paragraph: University B is a public research university
located in City C. It was established in the nineteenth
century and offers programs in engineering and science.
```

Table 9 summarizes the resulting memory and trajectory collections.

Table 9: Statistics of the HotpotQA memory and trajectory collections.

Statistic	Value
Questions (tasks)	100
Total memory units	1,000
Gold memory units	200
Distractor memory units	800
Candidate-pool size	10
Memory-selection budget K	2
Total trajectories	987
Success trajectories	495
Failure trajectories	492
Trajectories per task	8–12 (mean: 9.9)
Trajectory length (min/mean/max)	4 / 6.6 / 10

ALFWorld same-instance trajectory and memory construction. We construct the ALFWorld trajectory evidence from 100 solvable games covering five task types, with approximately 20 trajectories collected for each game. All trajectories are executed in the actual text-based environment, and their binary outcomes are determined by the final environment `won` state. A trajectory consists of a sequence of $(\text{observation}, \text{action}, \text{next observation})$ steps. A memory unit is defined as an experience transition consisting of the task goal, the pre-action observation, the executed action, and the resulting environment feedback. For each trajectory, the memory units corresponding to its executed experience transitions constitute the memory subset A_i used by the trajectory-grounded value estimator. Two simplified examples consistent with this representation are shown below:

Example 1:

```
Task goal: Hold a laptop and turn on the lamp.
```

```
Observation:  You are in the bedroom and can see several
receptacles.
Action:  go to bed 1
Next observation:  You arrive at bed 1 and observe a laptop.
```

Example 2:

```
Task goal:  Clean an apple and place it in the refrigerator.
Observation:  You are at sinkbasin 1 while holding apple 1.
Action:  clean apple 1 with sinkbasin 1
Next observation:  Apple 1 is now clean.
```

Table 10 summarizes the resulting trajectory and memory collections.

Table 10: Statistics of the ALFWorld trajectory and memory collections.

Statistic	**Value**
Games (tasks)	100
Total trajectories	2,000
Trajectories per game	20
Success trajectories	1,073
Failure trajectories	927
Total unique memory units	1,311
Memory units per game	13.1
`pick_and_place_simple`	800 trajectories
`pick_heat_then_place`	500 trajectories
`look_at_obj_in_light`	300 trajectories
`pick_clean_then_place`	200 trajectories
`pick_two_obj_and_place`	200 trajectories

ALFWorld cross-task transfer construction. We construct a separate ALFWorld setting for the cross-task transfer evaluation in Table 2. The source and target pools follow the native ALFWorld `train`, `valid_seen`, and `valid_unseen` splits and cover all six task types considered in the transfer study. The final evaluation contains 183 playable held-out games, while the valuation source pool contains 686 training families drawn exclusively from the training split. Table 11 summarizes the resulting source and target pools by task type. Memory values are estimated only from trajectories associated with the matched training families, and no trajectory from the held-out game is used for valuation. The resulting training-derived values are then used directly for memory selection on the held-out game.

Cross-task mapping rules: Cross-task transfer uses two mappings: training source games are first matched to each test game using its PDDL task type, target object, and goal/receptacle attributes; action-level memory units are then canonicalized across games by removing scene-specific identifiers and normalizing equivalent action forms before being instantiated in the test game.

Table 11: ALFWorld cross-task transfer pools. Held-out games are drawn from the native validation splits, while valuation evidence is constructed exclusively from matched families in the training split.

Task Type	**Held-out Games**	**Training Families**
`pick_two_obj_and_place`	29	74
`pick_heat_then_place`	24	67
`pick_cool_then_place`	31	80
`pick_clean_then_place`	38	89
`look_at_obj_in_light`	19	54
`pick_and_place_simple`	42	322
Total	**183**	**686**

For each held-out game, relevant source families are identified using ALFWorld-provided PDDL attributes, including the target object and goal/receptacle structure. This procedure matches *structurally related* training tasks rather than identical instances of the held-out game: the source and target games remain in disjoint native data splits.

C.2 Baseline Details

C.2.1 No Memory

Ablates the memory-augmentation channel: the downstream agent receives only the original task/query, with no memory units appended to its prompt. Serves as a memory-free reference for end-to-end performance and token consumption.

C.2.2 BM25

Lexical retrieval baseline using `rank_bm25.BM25Okapi` with library defaults ($k_1 = 1.5$, $b = 0.75$). Both the query and each candidate memory are lowercased and whitespace-tokenized without stopword removal or stemming.

C.2.3 Embedding Similarity

Dense retrieval using `sentence-transformers/all-MiniLM-L6-v2` with ℓ_2-normalized outputs. Query and document embeddings are pre-computed once and cached to disk. Ranking uses cosine similarity, which is equivalent to a dot product under normalization:

$$\text{score}(m, q) = \frac{\langle e_q, e_m \rangle}{\|e_q\| \, \|e_m\|}. \tag{C.1}$$

Documents are rendered identically to the BM25 baseline. The top-K units by cosine similarity are returned.

C.2.4 MMR

Maximal Marginal Relevance over the same embedding space as method (3), with $\lambda_{\text{MMR}} = 0.7$. Greedy selection maximizes:

$$\arg \max_{m \in \mathcal{M} \setminus S} \left[\lambda_{\text{MMR}} \, \text{sim}(m, q) - (1 - \lambda_{\text{MMR}}) \max_{m' \in S} \text{sim}(m, m') \right], \tag{C.2}$$

where S is the set of already-selected units. This introduces an explicit redundancy penalty and serves as a diversity-aware retrieval control.

C.2.5 Presence-only Mean Reward (PMR)

PMR ranks candidate memories by the mean reward of logged trajectories in which they appear, omitting VAM's memory-absence term. It uses the same trajectory pool and reward definition as VAM and selects the top-K memories.

C.2.6 LLM Selector

Delegates memory selection to a generative model given the task and the full candidate list, without additional trajectory evidence. For cross-task transfer, candidate memories come from the same structurally matched training pool used by VAM.

Prompt template:

> *"You are selecting the most useful memory units to help an agent solve a task.*
> *Current task: {query}. Below are ALL candidate experience units extracted from prior attempts at this task. Each unit shows a tool-call transition and its result.*
> *Candidate memory units: {candidates_text}. Select the top-{k} most useful ones for solving the task. Consider which tool calls and results are most likely to guide the agent to the correct solution. Return ONLY the numbers of the top-{k} most useful units, separated by commas.*
> *For example: 3, 7."*

> *"You are selecting the most useful paragraphs to help answer a multi-hop question.*
> *Question: {query}.*
> *Below are ALL candidate paragraphs that could be provided as context. Each has a title and content.*
> *Candidate paragraphs: {candidates_text}.*
> *Select the top-{k} most useful ones for answering the question. A multi-hop question requires information from multiple paragraphs to reason through to the answer.*
> *Return ONLY the numbers of the top-{k} most useful paragraphs, separated by commas.*
> *For example: 3, 7."*
>
> *"You are selecting the most useful action hints to help an agent complete a household task.*
> *Task: {game['task_goal']}.*
> *Below are all candidate actions from prior attempts at this task.*
> *Candidate actions: {candidates_text}.*
> *Select the top-{k} most useful actions that would help an agent succeed at this task.*
> *After your reasoning, start a new line with ANSWER: followed by ONLY the numbers separated by commas (e.g., "ANSWER: 2, 5, 7")."*

Selected indices are parsed from the output using the regular expression, deduplicated in order of appearance, and mapped back to candidate units. LLM calls use `MiniMax-M2.7` for ToolBench and HotpotQA, and `DeepSeek-V4-Flash` for ALFWorld, consistent with the downstream backbone used for each benchmark. We set `max_tokens=4096` and `temperature=0` where supported; MiniMax uses its server-default temperature setting. Up to four retries are performed with exponential backoff and a one-second inter-task delay.

C.2.7 REFLECTION

Extends the LLM Selector with contrastive trajectory evidence. In addition to the candidate list, the prompt exposes up to five successful and five failed trajectories for the same task, each rendered in full via `format_trajectory_full`, without observation truncation.

Prompt template:

> *"You are selecting the most useful memory units to help an agent solve a task.*
> *Current task: {query}.*
> *Below are ALL candidate experience units extracted from prior attempts at this task. Each unit shows a tool-call transition and its result.*
> *Candidate memory units: {candidates_text}.*
> *Below are trajectories from previous attempts at this SAME task, labelled by outcome.*
> *Successful trajectories (the agent solved the task): {success_text}.*
> *Failed trajectories (the agent did not solve the task): {failure_text}.*
> *Compare the successful and failed trajectories. Based on the contrast, select the top-{k} candidate units that are most critical for success—the ones whose presence in a trajectory is associated with success and whose absence is associated with failure.*
> *Return ONLY the numbers of the top-{k} most useful units, separated by commas.*
> *For example: 3, 7."*
>
> *"You are selecting the most useful paragraphs to help answer a multi-hop question.*
> *Below are retrieval trajectories from previous attempts on the SAME question. They show which paragraphs were retrieved and whether the attempt succeeded.*
> *Question: {query}.*

CANDIDATE PARAGRAPHS (all available): {candidates_text}.
SUCCESSFUL retrieval trajectories: {success_text}.
FAILED retrieval trajectories: {failure_text}.
Based on the contrast between successes and failures, select the top-{k} paragraphs that are most critical for answering the question. Focus on paragraphs that appear in successful trajectories but are missing in failed ones.
Return ONLY the numbers of the top-{k} most useful paragraphs, separated by commas.
For example: 3, 7."

"You are selecting the most useful action hints to help an agent succeed.
Task: {game['task_goal']}.
CANDIDATE ACTIONS: {candidates_text}.
{traj_context}.
Compare the successful and failed trajectories.
Select the top-{k} actions that are most critical for success.
After your reasoning, start a new line with ANSWER: followed by ONLY the numbers separated by commas (e.g., "ANSWER: 1, 3, 5").

Reflection uses the same trajectory pool and success/failure labeling rule as VAM, but its prompt includes at most five successful and five failed trajectories. Parsing, model configuration, retries, and caching are identical to those of the LLM Selector.

C.2.8 PROCEDURAL

Uses the same interface as Reflection but is conditioned only on successful trajectories, with up to five full-text trajectories. Failed trajectories are not exposed.

Prompt template:

"You are selecting the most useful memory units to help an agent solve a task.
Current task: {query}.
Below are ALL candidate experience units extracted from prior attempts at this task. Each unit shows a tool-call transition and its result.
Candidate memory units: {candidates_text}.
Below are trajectories from previous SUCCESSFUL attempts at this SAME task.
Successful trajectories (the agent solved the task): {success_text}.
Based on the patterns in the successful trajectories, select the top-{k} candidate units that are most useful for solving the task—the ones whose presence in a successful trajectory indicates a critical tool-call transition.
Return ONLY the numbers of the top-{k} most useful units, separated by commas.
For example: 3, 7."

"You are selecting the most useful paragraphs to help answer a multi-hop question.
Below are SUCCESSFUL retrieval trajectories from previous attempts on the SAME question.
Question: {query}.
CANDIDATE PARAGRAPHS (all available): {candidates_text}.
SUCCESSFUL retrieval trajectories: {success_text}.
Based on these successful attempts, select the top-{k} paragraphs that are most critical for answering the question. Focus on paragraphs that appear consistently across successful trajectories.
Return ONLY the numbers of the top-{k} most useful paragraphs, separated by commas.
For example: 3, 7."

"You are selecting the most useful action hints to help an agent succeed.
Task: {game['task_goal']}.
CANDIDATE ACTIONS: {candidates_text}.

{*traj_context*}.
Based on what consistently worked in successful trajectories, select the top-{k} *actions that are most useful for success.*
After your reasoning, start a new line with ANSWER: followed by ONLY the numbers separated by commas (e.g., "ANSWER: 1, 3, 5").

Tasks with zero successful trajectories receive an empty selection, since procedural memory is defined from successful trajectories.

Parsing, model configuration, retries, and caching are identical to those of the LLM Selector.