ATTENTION-DERIVED MEMORY-USE TRACES FOR AGENT MEMORY REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving agent memory is usually updated from task outcomes, execution trajectories, and post-hoc textual reflections. These signals can guide memory writing, but they provide limited evidence about how retrieved memory was actually used in the decisions made during execution. Without such read-side evidence, it is difficult to distinguish among different causes of memory-related errors. We introduce *execution-time memory utilization* as a complementary signal for persistent memory refinement. To estimate this utilization, we turn to attention as an indicator of memory access. Because raw attention is distributed across layers and heads, remains at the token level, and is not aligned with semantic memory segments or agent decisions, we identify model-specific retrieval heads and aggregate their attention into a segment-by-decision memory-use trace. Intervention experiments provide temporal and segment-level evidence that this trace reflects memory utilization. We then apply this signal in Attention-Guided Memory Refinement (AGMR), a simple refinement procedure that compares the utilization expected from reflection with the utilization observed in the trace to propose targeted edits. Across three interactive benchmarks and two models, AGMR improves task performance while generally reducing memory length.

1 INTRODUCTION

Large language model (LLM) agents increasingly store trajectories, reflections, workflows, and reusable experience to support interactive and long-horizon decision making Yao et al. (2022b); Park et al. (2023); Wang et al. (2023); Zheng et al. (2023); Wang et al. (2024b). Because accumulated experience is inevitably incomplete, recent systems treat memory as a self-evolving component that is revised through subsequent experience and reflection Shinn et al. (2023); Zhao et al. (2024); Suzgun et al. (2026); Zhang et al. (2025a); Cai et al. (2025); Fang et al. (2026); Yan et al. (2025).

Most self-evolving memory systems decide how to update memory from task outcomes, execution trajectories, and post-hoc textual analyses. Abstractly, they revise a retrieved memory M according to the observed trajectory τ, outcome y, and reflection r. These signals describe what happened or provide retrospective interpretations, but they do not explicitly record how retrieved memory dynamically participated in the decisions made during execution. In other words, current systems learn how to rewrite memory from what happened, without directly examining how that memory was read and utilized during execution. This omission matters: without read-side evidence, memory refinement remains a free-form reflection process over text, with no external anchor for deciding which memory segments to update and how, even though generated explanations can be plausible yet unfaithful accounts of model behavior Turpin et al. (2023); Agarwal et al. (2024); Chuang et al. (2026); Arcuschin et al. (2025).

Execution-time utilization provides a complementary view. For the same failed decision, there may be different underlying causes (Section 2.1). These cases may be difficult to distinguish from execution trajectories and textual reflections alone, yet call for different memory interventions. Reflection identifies memory segments that could support the correct decision, while an execution-time trace estimates how strongly different segments were accessed. Comparing these two sources of evidence can help distinguish the underlying causes and determine which memory segments to revise and how. This motivates three research questions:

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.

RQ1: Necessity. Why can purely textual post-hoc evidence be insufficient to determine the appropriate memory intervention, and what complementary information is provided by an execution-time read-side trace?

RQ2: Extraction and validity. How can we extract a segment-by-decision trace of memory access, and what evidence supports interpreting it as a proxy for memory utilization?

RQ3: Actionability and utility. Does comparing the utilization expected from reflection with the utilization observed in the trace yield targeted updates that really provide value beyond semantic reflection alone and improve agent performance?

To obtain this read-side evidence, we turn to attention, but raw attention weights are not directly suitable for this purpose. Attention is functionally heterogeneous across layers and heads and remains token-level; it must therefore be aggregated over semantic memory segments and aligned with individual agent decisions. Building on evidence that a sparse subset of attention heads supports contextual retrieval and information routing Wu et al. (2024); Kahardipraja et al. (2025); Bick et al. (2025); Zhang et al. (2025b), we identify retrieval-sensitive heads for each backbone and aggregate their attention over memory segments and decision steps to construct an actionable *segment-by-decision memory-use trace*. The resulting utilization matrix is treated as an internal proxy for memory access.

We demonstrate the utility of this signal in **Attention-Guided Memory Refinement (AGMR)**, a deliberately simple refinement procedure for trajectory memory. For a failed execution, semantic reflection localizes the first erroneous decision and the memory expected to support it; the utilization trace then helps distinguish different causes of failure to guide targeted memory edits. For a successful execution, semantic redundancy judgments and utilization evidence jointly guide selective memory simplification. Our contributions are threefold:

- **Read-side perspective.** We identify execution-time memory utilization as an overlooked source of evidence for self-evolving memory and show why post-hoc evidence can leave the appropriate intervention ambiguous.
- **Utilization extraction and diagnosis.** We identify memory retrieval sensitive attention heads, transform token-level attention into a segment-by-decision memory-use trace, and evaluate the trace with functional, localized causal, confound, and stability analyses.
- **Verified memory refinement.** We combine semantic reflection with utilization-grounded diagnosis in a simple memory updater and show gains in agentic task performance while generally reducing memory length.

2 Execution-Time Memory-Use Traces

2.1 Why Read-Side Evidence Is Needed

We first address **RQ1** by asking what is missing when a memory update is based only on the retrieved text, the resulting trajectory, the outcome, and post-hoc semantic reflection. Consider a persistent memory base $\mathcal{B}$ and a task x. The agent retrieves a memory $M = [m_1, \ldots, m_K]$ from $\mathcal{B}$, where each m_i is a semantically coherent segment, and produces an execution trajectory $\tau = (a_1, \ldots, a_T)$. After observing outcome y, a conventional self-evolving updater may produce textual feedback r and revise the retrieved memory as $M' = g(M, \tau, y, r)$. This abstraction captures the dominant purely textual evidence, but contains no explicit trace of how M was accessed while τ was generated.

Consider a wrong decision at step t, and suppose semantic reflection identifies a memory segment m_s that should support the correct action. Under a fixed high-attention criterion, attention allocation falls into three mutually exclusive and exhaustive cases: (i) m_s receives high attention, and no other segment does; (ii) neither m_s nor any other segment receives high attention; or (iii) at least one other segment receives high attention, whether or not m_s does. These cases correspond to accessed-but-ineffective support, missed support, and potential competition from other memory, respectively. All three patterns occur repeatedly in our experiments. In our observations, case (i) arises mainly when m_s supports the correct action but also contains misleading expressions, which may obscure its useful guidance during execution. Figure 1 shows representative examples at the first erroneous decision t, with each heat strip giving the model's utilization over the retrieved memory segments at that decision. We defer the construction of this utilization signal to Section 2.2; the colors in each strip retain the scale of its corresponding full segment-by-decision trace, shown in Figure 11.

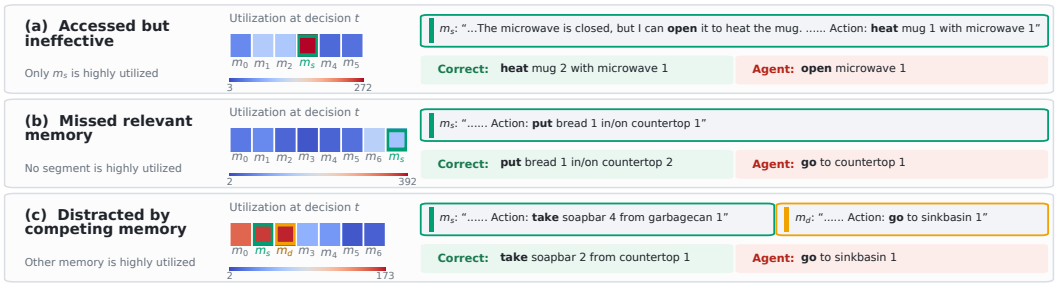


Figure 1: **Representative read-side patterns at the first erroneous decision.** Green outlines mark the segment m_s that semantic reflection identifies as support for the correct action; in (c), the amber outline marks an action-conflict segment m_d whose utilization exceeds that of m_s. *Correct* gives the action that should be taken at t, whereas *Agent* gives the action actually produced. In (a), the highly utilized m_s contains a misleading *open* cue that is reflected in the wrong action; in (b), m_s is insufficiently utilized; and in (c), the agent follows the competing *go* action from m_d.

Textual analysis of (M, τ, y, r) alone does not directly reveal which attention-allocation pattern occurred. Consequently, an updater based only on these text inputs lacks explicit evidence for choosing among these access-dependent edits. We therefore augment the update with a memory-use trace $\mathbf{U}$:

$$M' = g(M, \tau, y, r, \mathbf{U}).$$

This trace allows the updater to condition its edits on real memory access patterns.

2.2 Extracting Decision-Aligned Memory-Use Traces

To address **RQ2**, we first define what a segment-by-decision memory-use trace is and then describe how to compute it from attention.

Definition 2.1 (Execution-time memory-use trace)**.** *Given a retrieved memory $M = [m_1, \ldots, m_K]$ and an execution trajectory $\tau = (a_1, \ldots, a_T)$, an execution-time memory-use trace is a matrix*

$$\mathbf{U} = [u_{i,t}] \in \mathbb{R}_{\geq 0}^{K \times T}, \tag{1}$$

where $u_{i,t}$ estimates how strongly memory segment m_i is accessed immediately before the agent produces decision a_t. Row $\mathbf{U}_{i,:}$ tracks one segment throughout execution, while column $\mathbf{U}_{:,t}$ shows how strongly each segment is accessed at decision a_t.

We estimate $\mathbf{U}$ from attention, using it as a proxy for segment-level memory access. Let $A_{q,k}^{(\ell,h)}$ denote attention from query token q to key token k at layer ℓ and head h. Raw attention is not directly suitable for this purpose: heads perform heterogeneous functions, while token-level scores must be aggregated into memory segments and aligned with individual decisions. We therefore first identify retrieval-sensitive heads using a supervision set.

To construct this set, we collect transition cases in which an execution fails without memory but succeeds with it. We then use an LLM to identify the first corrected decision and its supporting memory tokens, yielding $\mathcal{D}_{\text{ret}} = \{(P_j, S_j)\}_{j=1}^{N}$, where P_j is the prompt immediately before the decision and S_j is the supporting token set. Here, $q(P)$ denotes the final token of the prompt P. We score each layer–head pair by

$$s_{\ell,h} = \sum_{(P,S)\in\mathcal{D}_{\text{ret}}} \sum_{k\in S} A_{q(P),k}^{(\ell,h)}, \tag{2}$$

and select the top $K_{\text{head}} = 5$ heads as $\mathcal{H}_b^\star = \text{TopK}_{(\ell,h)}\, s_{\ell,h}$ for backbone b. Retrieval-head identification is a one-time preprocessing step per backbone: once $\mathcal{H}_b^\star$ is obtained, it remains fixed throughout any refinement. We suppress the backbone subscript on $\mathcal{H}^\star$ when it is unambiguous.

To assess the stability of head selection, we resample the supervision examples and recompute the top-five heads. Across 10,000 bootstrap resamples, the mean Jaccard overlap with the heads selected from the full dataset is 0.867 for Qwen and 0.811 for Llama. Selections from disjoint halves

of the supervision set share an average of 4.22 and 4.01 out of five heads, respectively, further supporting selection stability (Appendix D.2). Dataset construction and selected-head details appear in Appendix C.1–D.1. Figure 6 visualizes the $s_{\ell,h}$ distributions for Qwen and Llama.

At execution step t, we extract attention from the final prompt token q_t after applying the backbone's chat template. Because the hidden state at this position predicts the first response token, its attention provides a view of memory access immediately before the decision begins. Let $\bar{T}_i$ denote the non-observation tokens of memory segment m_i. We estimate its utilization before decision a_t as

$$u_{i,t} = \frac{n_t}{|\mathcal{H}^\star|} \sum_{(\ell,h)\in\mathcal{H}^\star} \sum_{k\in\bar{T}_i} A^{(\ell,h)}_{q_t,k}, \tag{3}$$

where n_t is the prompt token count. Excluding observation tokens limits the influence of verbose observations, while multiplying by n_t partially compensates for attention dilution as the prompt grows. Stacking these scores over memory segments and decisions produces $\mathbf{U}$ in Equation (1).

To assess whether the final prompt token provides a representative view of memory access, we compare the deployed $u_{i,t}$ with a variant that averages attention over the full Thought/Action generation. Across all tasks, the resulting segment rankings show strong agreement, with a median within-task Spearman correlation of 0.829 (Appendix E.5). We next test whether the resulting trace reflects memory utilization through group-level ablation, decision-level interventions, and random-head controls.

2.3 Grounding the Trace as a Utilization Proxy

Attention is observational, so a high value cannot by itself establish that a segment affected a decision. We therefore ground $\mathbf{U}$ at three levels. First, we test whether the selected retrieval heads contribute to memory-assisted execution at all. We then use localized interventions to test two decision-level properties: whether the trace identifies *when* the same segment matters and whether it identifies *which* segments matter at a fixed decision.

The selected retrieval heads contribute to memory-assisted execution. Figure 2 evaluates the heads as a group. Here, ICL denotes an in-context learning baseline with a fixed demonstration, while trajectory memory provides a retrieved memory. Across all four model–environment settings, adding trajectory memory improves agent performance, masking the selected retrieval heads removes much of that gain, and masking the same number of random heads has a substantially smaller effect. Thus, the selected heads are functionally relevant to memory-assisted execution.

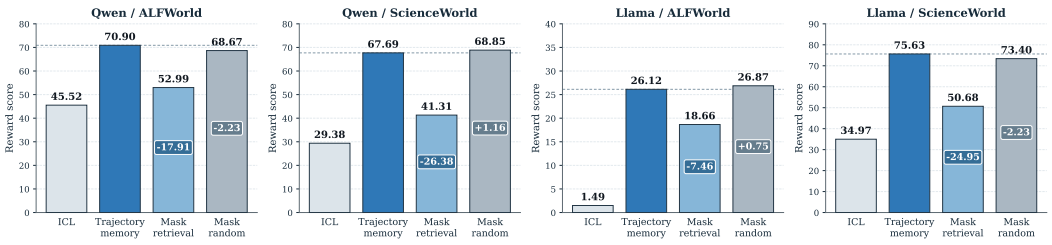


Figure 2: **Selected retrieval heads contribute to memory-assisted execution.** Masking $\mathcal{H}^\star$ removes much of the trajectory-memory benefit, whereas masking the same number of random heads has a substantially smaller effect. Dashed lines mark the unmasked trajectory-memory score; labels inside the masked-condition bars report reward changes relative to this baseline.

Decision-level intervention. To test $\mathbf{U}$ at the segment–decision level, we intervene on attention from the tested heads to a target memory segment. For a target segment m_i at decision a_t, we prevent the tested heads from attending to any token in m_i by setting the corresponding attention logits to $-\infty$ before softmax. The prompt, token positions, and all other attention logits remain unchanged, and attention is renormalized over the remaining context.

Let P_t denote the prompt before decision a_t, and let $\mathrm{tok}(a_t) = (z_{t,1}, \ldots, z_{t,J_t})$ be the token sequence of its recorded Thought/Action response. Under condition $c \in \{\mathrm{base}, \mathrm{masked}\}$, we teacher-

force this response and compute its mean token-level negative log-likelihood (NLL):

$$L_c(a_t) = -\frac{1}{J_t}\sum_{j=1}^{J_t} \log p_\theta^{(c)}\left(z_{t,j} \mid P_t, z_{t,<j}\right), \qquad \Delta L = L_{\text{masked}}(a_t) - L_{\text{base}}(a_t). \tag{4}$$

A larger $L_c(a_t)$ means that the model assigns lower probability to the recorded decision. Thus, $\Delta L > 0$ indicates that the segment supported the decision, whereas $\Delta L < 0$ indicates an opposing effect. We use $|\Delta L|$ to measure how strongly blocking access to the segment changes the decision, regardless of direction. If $u_{i,t}$ reflects memory utilization, larger values should correspond to larger $|\Delta L|$. Implementation details are provided in Appendix E.7.

Utilization identifies when the same segment matters. For each memory segment m_i, we use its row in the unperturbed trace $\mathbf{U}$ to identify the decisions at which its utilization is highest and lowest, $t_i^{\max} = \arg\max_t u_{i,t}$ and $t_i^{\min} = \arg\min_t u_{i,t}$. We then mask m_i in the selected retrieval heads separately at decisions $a_{t_i^{\max}}$ and $a_{t_i^{\min}}$, and compare the resulting causal sensitivities. If the trace reflects utilization, the first intervention should produce a larger $|\Delta L|$, and this difference should grow with the utilization gap. Figure 3 shows the comparison and the random-head controls.

Table 1: **Same-segment temporal intervention.** *Ret.-selected* uses retrieval-head trace $\mathbf{U}$ to select each segment's maximum- and minimum-utilization decisions, then masks random heads. *Own-selected* instead uses the random heads' own trace to select the two decisions and masks those same heads. Brackets show 95% confidence intervals.

Statistic	Retrieval heads	Random, ret.-selected	Random, own-selected
Mean $\|\Delta L\|$ at maximum	0.00697	0.00190	0.00206
Mean $\|\Delta L\|$ at minimum	0.00215	0.00191	0.00178
Maximum/minimum ratio	3.25 [2.81, 3.77]	1.00 [0.92, 1.08]	1.16 [1.03, 1.29]
Maximum–minimum gap	0.00482 [0.00391, 0.00589]	−0.000003 [−0.00016, 0.00015]	0.00028 [0.00006, 0.00049]
Positive-gap tasks	94.4%	51.7%	61.8%
Attention-gap/effect-gap raw ρ	0.406	−0.019	0.042

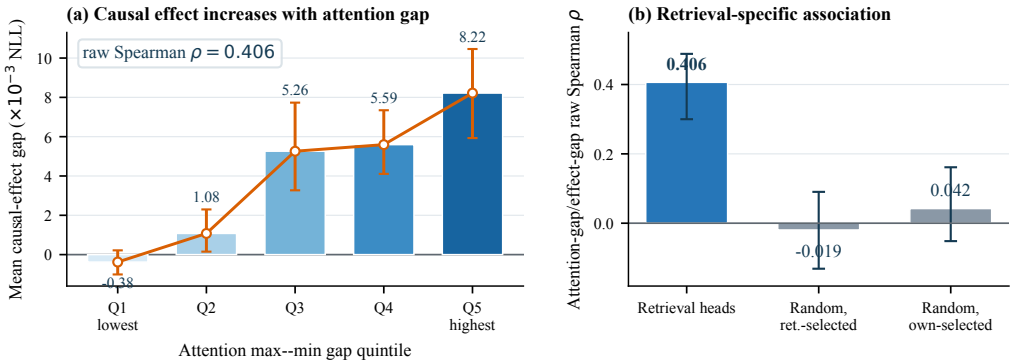


Figure 3: **Utilization identifies when the same segment matters.** For each segment, we compare interventions at its maximum- and minimum-utilization decisions. (a) The x-axis groups segments by quintiles of their utilization gap, $u_{i,t_i^{\max}} - u_{i,t_i^{\min}}$; the y-axis shows the mean causal-sensitivity gap, $|\Delta L|_{t_i^{\max}} - |\Delta L|_{t_i^{\min}}$. (b) Spearman correlations between the two gaps for retrieval heads and two random-head controls.

As reported in Table 1, at a segment's maximum-utilization decision, masking it produces mean $|\Delta L| = 0.00697$, versus 0.00215 at its minimum-utilization decision: a 3.25× ratio (95% CI $[2.81, 3.77]$). The gap is positive for 94.4% of tasks, and the utilization max–min gap predicts the causal-effect gap ($\rho = 0.406$). Masking random heads at the same decisions yields essentially no effect gap (ratio 1.00; $\rho = -0.019$), while using random heads to select their own extrema yields only a small effect (ratio 1.16; $\rho = 0.042$). Thus, the trace identifies when the same segment has greater influence specifically through the selected retrieval heads.

Utilization identifies which memory segments matter. We evaluate every memory segment separately at the first decision of each task. For each segment m_i, we record its utilization score $u_{i,1}$,

block attention to all tokens in m_i through the selected retrieval heads and the corresponding heads in adjacent layers, and measure the resulting change $|\Delta L_i|$ in the recorded decision. For comparison, we repeat the procedure with an equal-size set of random heads, using their own segment scores.

Figure 4 shows that the association between utilization scores and intervention effects is concentrated in the highest-scoring quintile rather than increasing smoothly across all segments. The task-averaged mean $|\Delta L|$ remains between 0.00207 and 0.00295 from Q1 through Q4, then rises to 0.01075 in Q5. In contrast, the random-head effect remains nearly flat across quintiles ($\rho = 0.009$), producing a Q5–Q1 gap of only 0.00008, compared with 0.00869 for the retrieval heads. Because the overall mean $|\Delta L|$ differs by only $1.63\times$, this separation of more than $100\times$ cannot be explained by uniformly stronger retrieval-head interventions. And in 93.2% of tasks, a segment's utilization score $u_{i,1}$ correlates positively with its intervention effect $|\Delta L_i|$ (Appendix Table 8). The trace therefore identifies a small set of highly utilized segments with substantial causal influence.

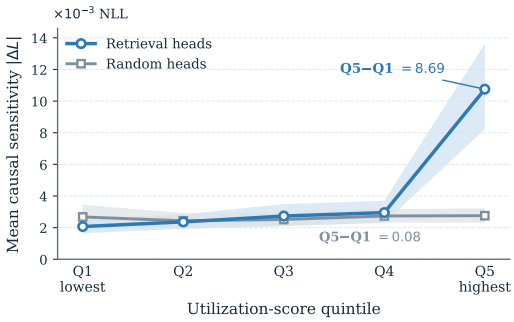


Figure 4: **Segment-level causal sensitivity across utilization-score quintiles.** Lines show task-averaged means; bands are task-bootstrap 95% CIs.

Together, the group-level ablation, same-segment temporal comparison, fixed-decision segment comparison, and random-head controls support interpreting $\mathbf{U}$ as a decision-local proxy for memory utilization through the selected retrieval heads. Appendices E.6 and E.5 provide complementary task-success interventions and checks of trace construction.

2.4 From Expected and Observed Use to Diagnosis

Having grounded $\mathbf{U}$ as a decision-local utilization proxy, we use it to distinguish the three access patterns introduced in Section 2.1: accessed-but-ineffective support, missed support, and competition from other memory. For each failed execution, we use two LLM calls with distinct roles to separate semantic localization from utilization-grounded diagnosis. The *Semantic Reflector* maps the retrieved memory, trajectory, and environment outcome to a reflection r, the first erroneous decision $t^\star$, and the set $I_{t^\star}$ of memory segments expected to support the corrected action:

$$(r, t^\star, I_{t^\star}) \leftarrow \text{Reflector}(M, \tau, y). \tag{5}$$

The *Attention-Guided Refiner* then compares this expected support with the observed utilization profile $\mathbf{U}_{:,t^\star}$, diagnoses the access pattern, and proposes a corresponding edit:

$$d_{t^\star} \leftarrow \phi(I_{t^\star}, \mathbf{U}_{:,t^\star}, y), \qquad \widetilde{M} \leftarrow \psi(M, r, d_{t^\star}). \tag{6}$$

Under the fixed high-utilization rule, ϕ assigns one of the three cases from Section 2.1: the intended segment is high and no other segment is high; neither the intended segment nor any other segment is high; or at least one non-intended segment is high. These are, respectively, accessed-but-ineffective support, missed relevant memory, and distraction. The first prompts inspection of whether the accessed segment contains misleading content or insufficiently emphasizes the correct guidance; the latter two call for increased target salience or disambiguation of the target and competing memory. Cases without an aligned intended segment or a resolved utilization pattern are assigned to *Other*; exact thresholds are given in Appendix G.

We use controlled interventions to validate the three diagnoses and their corresponding memory edits. At the localized decision $t^\star$, we measure the model's preference for the correct response over the recorded wrong response as $\mathcal{M} = \overline{\log p(a^{\text{correct}})} - \overline{\log p(a^{\text{wrong}})}$, where $\overline{\log p(\cdot)}$ is the mean token-level log-likelihood of each response. We intervene on the five retrieval heads before softmax: *boosting* adds $\log 4$ to the target segment's attention logits, while *masking* sets them to $-\infty$. Figure 5(b) shows that, when relevant memory is missed, boosting the intended segment m_s increases $\mathcal{M}$ by 0.0307, compared with 0.0004 for a different segment with a similar number of tokens and 0.0058 through random heads. In (c), for distraction, masking the action-conflict

segment m_d increases $\mathcal{M}$ by 0.0774; combining this mask with boosting m_s yields the largest increase (0.1161), while the two controls remain near zero. Panel (a) examines the more complex accessed-but-ineffective case. As discussed for case (i) in Section 2.1, the intended segment m_s may support the correct action while also containing misleading expressions that support the wrong action, requiring its content to be corrected rather than simply emphasized. Indeed, masking m_s lowers both correct- and wrong-response probabilities in 13/34 cases, providing direct evidence that the same segment can sometimes support both responses when it contains misleading expressions. Appendix D.2 analyzes the robustness of the diagnoses to the high-attention threshold.

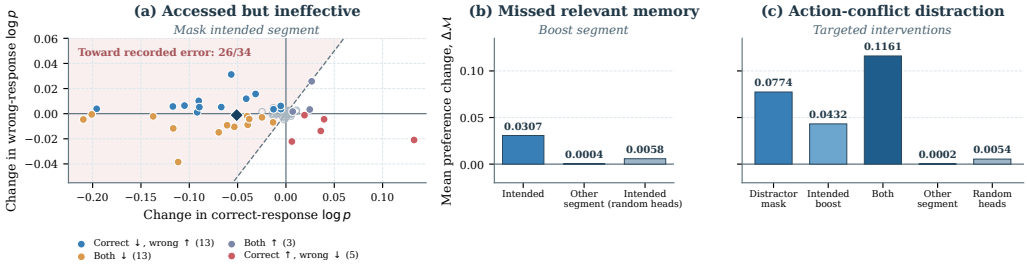


Figure 5: **Controlled interventions validate the diagnoses and corresponding memory edits.** (a) Changes in correct- and wrong-response log probabilities after masking the intended segment in accessed-but-ineffective cases; hollow points apply the same mask through random heads, and points above the diagonal shift preference toward the wrong response. (b–c) Mean preference change, $\Delta\mathcal{M} = \mathcal{M}_{\text{intervention}} - \mathcal{M}_{\text{base}}$, where positive values indicate a shift toward the correct response. In the *Other segment* bars, the same intervention is applied through retrieval heads to a non-intended segment with the closest token count; random-head bars apply the same intervention to the original target through random heads. *Both* masks the distractor and boosts the intended segment.

3 Attention-Guided Memory Refinement

The previous section validated the memory-use trace as a proxy for memory utilization. Building on this signal, we now demonstrate its utility in a complete agentic system, **Attention-Guided Memory Refinement (AGMR)**. As in Section 2.4, AGMR involves two LLM modules—the *Semantic Reflector* and the *Attention-Guided Refiner*—alongside the agent that executes the tasks.

AGMR operates on an existing flat trajectory-memory base $\mathcal{B}$. For each task, it retrieves a memory entry, and the agent executes the task while $\mathbf{U}$ is recorded; the trace reuses the decoding pass that already produces each action and adds no additional forward pass (Appendix E.9). For a failed execution, the Reflector produces the reflection r, the first erroneous decision $t^\star$, and the intended segments $I_{t^\star}$, and the Refiner derives the diagnosis $d_{t^\star}$ from $\mathbf{U}$ and realizes it as a textual memory edit. The Refiner never boosts or masks attention directly; its natural-language edits pursue the effects suggested by the controlled experiments in Section 2.4. For accessed-but-ineffective support, it corrects misleading content or strengthens guidance that failed to induce the correct action. For missed support, it rewrites the intended segment with clearer conditions, action cues, and more explicit guidance to make it easier to access at the relevant decision. For distraction, it revises the competing high-utilization segment m_d by removing ambiguous cues or clarifying when its guidance applies, and strengthens the intended segment m_s.

Algorithm 1 **Attention-Guided Memory Refinement**

Require: Task x, memory base $\mathcal{B}$
Retrieve M and execute x to obtain $(\tau, y, \mathbf{U})$
if y is failure **then**
  Reflect to obtain $(r, t^\star, I_{t^\star})$
  $d_{t^\star} \leftarrow \phi(I_{t^\star}, \mathbf{U}_{:,t^\star}, y)$
  $\widetilde{M} \leftarrow \text{TextEdit}(M, r, d_{t^\star})$
else
  Identify redundant candidates $\mathcal{R}$
  $\widetilde{M} \leftarrow \text{Simplify}(M, \mathcal{R}, \mathbf{U})$
end if
Re-execute x with $\widetilde{M}$
Commit $\widetilde{M}$ if verified; otherwise retain M

AGMR also evolves memory on successful executions, through a conservative simplification branch: the Reflector proposes semantically redundant segments, and the Refiner filters them with $\mathbf{U}$ using their average and peak scores, compressing or removing only segments that are both semantically re-

dundant and weakly utilized. This protects content whose use is concentrated at a particular decision while keeping the memory concise. Algorithm 1 summarizes the procedure.

Every candidate update is verified by re-executing the same task: an edit for a failed execution is committed only if it corrects or improves the outcome y, and a simplification is committed only if it preserves success; otherwise, AGMR rolls back to the original memory M. In the main experiments, this procedure refines the memory base on training tasks before the resulting memory is frozen for evaluation. The exact editing prompts and benchmark-specific criteria are provided in Appendices E.3 and G.

4 Experiments

4.1 Experimental Setup

We evaluate AGMR on ALFWorld Shridhar et al. (2020), ScienceWorld Wang et al. (2022), and WebShop Yao et al. (2022a), covering embodied household interaction, science-oriented multi-step reasoning, and web-based decision making. The agent backbones are Qwen2.5-7B-Instruct Qwen et al. (2025) and Llama-3.1-8B-Instruct Grattafiori et al. (2024). We report task performance on all benchmarks, average execution steps on ALFWorld and ScienceWorld, and the average token length of retrieved memory on all benchmarks. Average steps are omitted for WebShop because higher reward may legitimately require additional attribute inspection and clicks. All results in Tables 2 and 3 are averages over three independent runs, each of which repeats the complete memory-construction procedure before evaluation.

Baselines include ReAct Yao et al. (2022b), ICL with a fixed demonstration, AWM Wang et al. (2024b), Dynamic Cheatsheet (DC) Suzgun et al. (2026), and raw retrieved trajectory memory (Traj) from the ETO SFT data Song et al. (2024). AGMR starts from Traj and refines it offline on the training set, so the comparison with Traj most directly measures the effect of memory refinement. For fair comparison, the Semantic Reflector, Attention-Guided Refiner, AWM workflow extraction, and DC Memory Curator all use Gemini-3.1-pro-preview. Retrieval heads are selected once per backbone using only ALFWorld training transition cases. The resulting fixed top-five set is reused unchanged on ALFWorld, ScienceWorld, and WebShop, without target-benchmark reselection or adaptation. Thus, the ScienceWorld and WebShop evaluations test cross-benchmark and cross-prompt transfer from ALFWorld-selected heads. Further configurations, sensitivity analyses, online results, and cost statistics are provided in Appendix E. Section 2.3 provides the core intervention grounding for **RQ2**. The experiments below address **RQ3** through held-out performance and factor-matched controls that isolate the contribution of the utilization trace.

4.2 Overall Performance

Table 2 shows that AGMR achieves the best task performance across all three environments and both backbone models. It also obtains the fewest all-episode average execution steps on ALFWorld and ScienceWorld. The most informative comparison is with Traj: AGMR improves performance in every setting while generally shortening the retrieved memory. Thus, its gains do not arise from simply adding context; the update changes memory quality by strengthening useful information and simplifying low-value content with the guidance of memory use traces. AWM is more compact because it converts concrete trajectories into concise workflow abstractions, discarding much of their action-level detail. AGMR instead retains the trajectory-memory format, so Traj is the format-matched comparison for isolating refinement quality; the AWM comparison reflects a different performance–compression trade-off. A sequential Qwen-ALFWorld run further improves offline AGMR from 81.34 to 84.33 success while reducing average steps from 16.08 to 15.66; Appendix E.8 gives the protocol.

To isolate the attention signal more directly, we additionally construct two factor-matched controls (Table 3). The two-call text-only variant retains the two-stage refinement pipeline, LLM-call budget, retry policy, and re-execution verification, but withholds the utilization trace $\mathbf{U}$ from the second call. The Random-5-head variant keeps the AGMR prompt and protocol unchanged while replacing the selected retrieval heads with five random heads to construct trace $\mathbf{U}$. AGMR performs best in both settings while requiring the fewest execution steps. Because the controls retain the two-

Table 2: Performance comparison across environments and backbone models.

Model	Env	Metric	ReAct	ICL	AWM	DC	Traj	AGMR
Qwen-2.5 7B-Instruct	ALFWorld	Perf. ↑	41.79	45.52	64.18	52.24	70.90	**81.34**
		Avg step ↓	28.40	26.71	22.22	25.09	19.36	**16.08**
		Mem token ↓	–	–	**312.01**	1013.73	852.36	752.53
	SciWorld	Perf. ↑	29.88	29.38	50.39	63.73	67.69	**77.20**
		Avg step ↓	19.56	18.35	16.92	15.52	14.62	**13.78**
		Mem token ↓	–	–	**435.32**	1478.85	1040.62	990.89
	WebShop	Reward ↑	56.66	60.12	56.11	54.04	51.04	**63.86**
		Mem token ↓	–	–	**185.65**	618.14	1372.22	503.77
Llama-3.1 8B-Instruct	ALFWorld	Perf. ↑	0.00	1.49	7.46	11.19	26.12	**47.76**
		Avg step ↓	40.00	39.74	38.30	36.67	32.54	**25.52**
		Mem token ↓	–	–	**312.01**	1133.51	852.36	845.91
	SciWorld	Perf. ↑	26.50	34.97	52.14	72.28	75.63	**83.74**
		Avg step ↓	19.59	18.37	16.23	13.99	13.22	**12.04**
		Mem token ↓	–	–	**435.32**	1168.99	1040.62	918.53
	WebShop	Perf. ↑	33.06	52.10	41.56	29.68	41.45	**63.26**
		Mem token ↓	–	–	**185.65**	658.96	1372.22	567.57

Table 3: Factor-matched attention controls. Each result is reported as Perf./Steps (↑/↓).

Refinement setting	Attention input	Qwen–ALFWorld	Qwen–SciWorld	Llama–ALFWorld	Llama–SciWorld
Two-call text-only	None	76.13/17.36	72.85/14.15	41.42/28.06	79.72/12.53
Random-5-head	Random heads	75.37/18.28	70.91/14.38	38.97/29.81	77.68/12.79
AGMR	Retrieval heads	**81.34/16.08**	**77.20/13.78**	**47.76/25.52**	**83.74/12.04**

stage pipeline, call budget, retry policy, and verification procedure, these comparisons isolate the contribution of the retrieval-head utilization trace under the evaluated protocol.

A scenario-stratified analysis on Qwen–ALFWorld (Figure 9) shows that AGMR's gains over AWM and DC are concentrated in missed-memory and distraction cases, rather than in accessed-but-ineffective cases involving misleading or insufficient memory content that can often be diagnosed semantically. Appendix F presents representative memory-use traces $\mathbf{U}$ for the same task instances before and after refinement. In these paired cases, AGMR changes memory utilization in the predicted direction: previously missed target segments receive increased utilization, while distracting segments no longer dominate. These results provide direct evidence that AGMR's natural-language edits can realize the attention-level effects prescribed by the corresponding diagnoses.

5 Conclusion

Existing self-evolving memory systems update persistent experience from trajectories, outcomes, and post-hoc reflection. We introduced execution-time memory utilization as read-side evidence, extracted a memory-use trace from retrieval-sensitive attention, and used it to guide a memory updater. Interventions show that the trace predicts causal sensitivity through the identified retrieval pathway, and end-to-end evaluation shows that AGMR improves task performance while generally reducing memory length. More broadly, internal execution traces offer a promising way to ground agent self-evolution in how stored experience is accessed during decision making.

Limitations AGMR is currently instantiated for open-weight agents with accessible attention weights and trajectory memory. The intervention results ground $\mathbf{U}$ as a utilization measure through the identified retrieval pathway. Extending the framework to additional model interfaces, memory structures, and online settings is a promising direction.

Related Works Prior memory methods update from outcomes and reflection, while attention methods use internal signals for transient context selection. AGMR instead uses an execution-time memory-use trace for persistent updates; Appendix A provides a detailed comparison.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experimental design; assist with implementing, debugging, and editing research code; support the creation and revision of scientific figures; and help interpret and present empirical results. We also used these tools for brainstorming, literature discovery and summarization, and drafting and editing portions of the manuscript to improve its organization, clarity, and formatting.

We did not treat AI-generated outputs as authoritative sources or allow them to independently determine the scientific claims or conclusions of this work. The authors reviewed and revised all AI-assisted text, checked relevant citations against the source literature, inspected and tested AI-assisted code, and cross-checked figures and reported results against the underlying implementations and experimental records. We take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

REFERENCES

Chirag Agarwal, Sree Harsha Tanneru, and Himabindu Lakkaraju. Faithfulness vs. plausibility: On the (un) reliability of explanations from large language models. *arXiv preprint arXiv:2402.04614*, 2024.

Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooran Rajamanoharan, Neel Nanda, and Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful. *arXiv preprint arXiv:2503.08679*, 2025.

Aviv Bick, Eric Xing, and Albert Gu. Understanding the skill gap in recurrent language models: The role of the gather-and-aggregate mechanism. *arXiv preprint arXiv:2504.18574*, 2025.

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. Flex: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025.

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.

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

Yu-Neng Chuang, Guanchu Wang, Chia-Yuan Chang, Ruixiang Tang, Shaochen Zhong, Fan Yang, Andrew Wen, Mengnan Du, Xuanting Cai, Vladimir Braverman, et al. Faithlm: Towards faithful explanations for large language models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3802–3824, 2026.

Gaodan Fang, Vatche Isahagian, KR Jayaram, Ritesh Kumar, Vinod Muthusamy, Punleuk Oum, and Gegi Thomas. Trajectory-informed memory generation for self-improving agent systems. *arXiv preprint arXiv:2603.10600*, 2026.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025.

Weizhi Fei, Xueyan Niu, Guoqing Xie, Yingqing Liu, Bo Bai, and Wei Han. Efficient prompt compression with evaluator heads for long-context transformer inference. *arXiv preprint arXiv:2501.12959*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, 2025a.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025b.

Tomoyuki Kagaya, Thong Jing Yuan, Yuxuan Lou, Jayashree Karlekar, Sugiri Pranata, Akira Kinose, Koki Oguri, Felix Wick, and Yang You. Rap: Retrieval-augmented planning with contextual memory for multimodal llm agents, 2024.

Patrick Kahardipraja, Reduan Achtibat, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. The atlas of in-context learning: How attention heads shape in-context retrieval augmentation. *arXiv preprint arXiv:2505.15807*, 2025.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. Memgpt: towards llms as operating systems., 2023.

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

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, 2024.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, 2026.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

Wenshan Wang, Yihang Wang, Yixing Fan, Huaming Liao, and Jiafeng Guo. Quito: Accelerating long-context reasoning through query-guided context compression. In *China Conference on Information Retrieval*, pp. 136–148. Springer, 2024a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024b.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, et al. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

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

Wuwei Zhang, Fangcong Yin, Howard Yen, Danqi Chen, and Xi Ye. Query-focused retrieval heads improve long-context reasoning and re-ranking. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23802–23816, 2025b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

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

Yunlong Zhao, Haoran Wu, and Bo Xu. Leveraging attention to effectively compress prompts for long-context llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26048–26056, 2025.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. 2023.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

A Related Work

Self-evolving agent memory. Agent memory ranges from retrieved trajectory exemplars and contextual planning memories to workflow abstractions, procedural knowledge, and structured long-term stores Zheng et al. (2023); Kagaya et al. (2024); Packer et al. (2023); Hu et al. (2025a); Xu et al. (2025); Hu et al. (2025b); Zhong et al. (2024); Chhikara et al. (2025); Fang et al. (2025); Cao et al. (2025); Ouyang et al. (2025). Recent systems continually revise these memories through verbal reflection, experience accumulation, consolidation, or adaptive curation Shinn et al. (2023); Zhao et al. (2024); Suzgun et al. (2026); Zhang et al. (2025a); Cai et al. (2025); Fang et al. (2026); Yan et al. (2025). These methods substantially advance the memory-write process, but their update decisions are generally inferred from outcomes, trajectories, and generated textual feedback. Execution-time, segment-by-decision evidence about how retrieved memory participated in specific actions is rarely incorporated into persistent memory updates.

Mechanistic signals for context utilization. Mechanistic studies identify sparse retrieval, query-dependent, gathering, or routing heads that support contextual selection and information integration Wu et al. (2024); Kahardipraja et al. (2025); Zhang et al. (2025b); Bick et al. (2025). Related methods use attention to compress or filter long contexts Zhao et al. (2025); Fei et al. (2025); Wang et al. (2024a) or retain high-value cache entries Zhang et al. (2023); Liu et al. (2023); Li et al. (2024). These signals are primarily used for interpretation, transient context selection, or inference efficiency. Our focus is different: we aggregate retrieval-sensitive attention into an execution-time memory-use trace and use it as evidence for changing persistent memory. Prior work has therefore developed memory writing and contextual reading largely in isolation; this work connects the two sides of the memory loop.

B Verification Safeguard

Let $Q_x(M)$ denote execution quality on the originating task x and let $C(M)$ denote memory complexity, such as token length. Suppose a candidate $\widetilde{M}$ is committed only when $Q_x(\widetilde{M}) > Q_x(M)$, or when $Q_x(\widetilde{M}) = Q_x(M)$ and $C(\widetilde{M}) < C(M)$.

Under deterministic execution and exact evaluation, every committed update satisfies $\big(Q_x(M'), -C(M')\big) \succeq_{\text{lex}} \big(Q_x(M), -C(M)\big)$, where $\succeq_{\text{lex}}$ is lexicographic ordering. Accordingly, verification provides a task-local monotonic safeguard for every committed update.

Algorithm 2 summarizes the overall workflow of Attention-Guided Memory Refinement (AGMR). Given a task and a memory base, AGMR retrieves a flat trajectory memory and executes the agent to obtain a trajectory, outcome, and memory-use trace derived from attention. Depending on whether the execution succeeds or fails, the framework applies a diagnosis–update–verification process: the *Semantic Reflector* localizes candidate memory segments, the *Attention-Guided Refiner* uses attention-derived utilization to select a targeted intervention, and re-execution determines whether the candidate should be committed. Only verified updates are incorporated into the memory base. This algorithm provides a high-level overview of AGMR, complementing the detailed description in Section 3.

C Retrieval Head Identification

C.1 Construction of Retrieval Supervision Dataset

D Complete AGMR Procedure

To identify retrieval heads, we construct a supervision dataset of $(\texttt{key_context}, \texttt{formatted_prompt})$ pairs for each backbone model. The goal is to collect cases where a specific memory snippet is known to be useful for a key decision step, so that attention heads can be scored by whether they attend to this snippet before the step is generated.

We build this dataset from tasks whose outcomes are improved by adding retrieved trajectory memory. Taking Qwen-2.5-7B-Instruct on ALFWorld as an example, the agent achieves 41.79% success without memory. After inserting the most similar retrieved memory trajectory into the prompt, success increases to 70.90%. From this setting, we collect tasks that fail without memory but succeed with memory. These transition cases provide natural supervision, since the retrieved memory is likely to contain information that helps correct the original failure.

For each such task, we use an LLM analyzer to compare the original failed trajectory with the subsequent successful memory-augmented trajectory. The analyzer identifies the failure reason, the key step in the successful trajectory that contributes to the failure-to-success transition, and the corresponding memory snippet that supports this key step. This supporting snippet is recorded as `key_context`.

Based on this analysis, each supervision example is stored as a tuple $(\texttt{task_id}, \texttt{key_context}, \texttt{formatted_prompt})$. Here, `formatted_prompt` denotes the complete model input immediately before generating the identified key step, and `key_context` denotes the memory snippet expected to be retrieved at that moment. For Qwen, this stored input

Algorithm 2 Complete Attention-Guided Memory Refinement (AGMR)

Require: Task x, memory base $\mathcal{B}$, maximum retry number $K_{\max}$
Ensure: Updated memory base $\mathcal{B}$

Retrieve flat trajectory memory $M = [m_1, \ldots, m_K]$ from $\mathcal{B}$
Execute with M, obtaining trajectory τ, outcome y, and utilization trace $\mathbf{U}$
if y is failure **then**
    Semantic Reflector produces $(r, t^\star, I_{t^\star})$ from (M, τ, y)
    Diagnose $d_{t^\star} = \phi(I_{t^\star}, \mathbf{U}_{:,t^\star}, y)$
    Generate candidate $\widetilde{M} = \psi(M, r, d_{t^\star})$
    Re-execute and retry refinement up to $K_{\max}$ times if verification fails
else
    Semantic Reflector identifies redundant candidates $\mathcal{R}$
    Filter $\mathcal{R}$ using average and maximum utilization from $\mathbf{U}$
    Generate simplified candidate $\widetilde{M}$
    Re-execute the task with $\widetilde{M}$
end if
if verification succeeds **then**
    Commit $M' = \widetilde{M}$ to $\mathcal{B}$
else
    Discard $\widetilde{M}$ and retain $M' = M$
end if
return $\mathcal{B}$

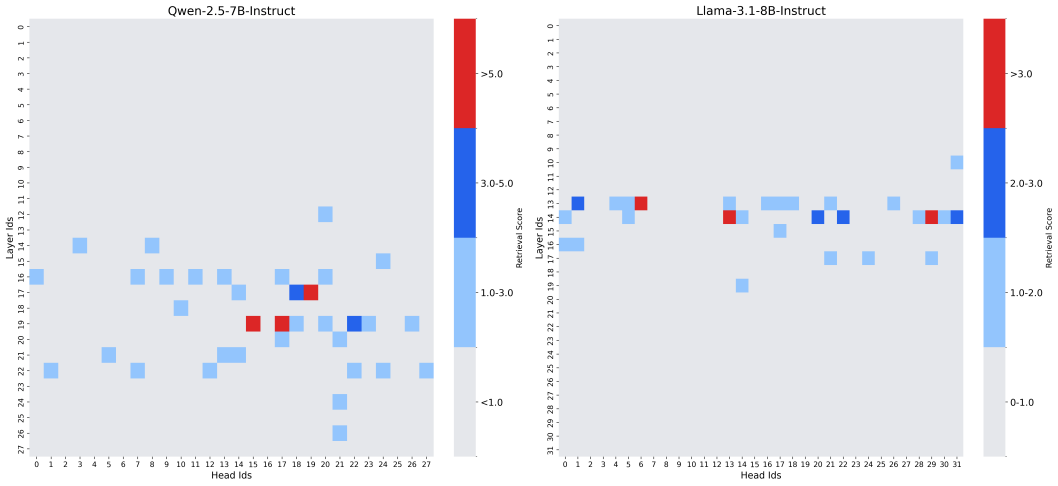


Figure 6: Retrieval Score heatmaps for all attention heads in Qwen-2.5-7B-Instruct (left) and Llama-3.1-8B-Instruct (right). Each entry shows how much attention a head assigns from the prompt's last token to the annotated `key_context` across the retrieval supervision set. Only a small subset of heads exhibits markedly high Retrieval Scores, and these heads are concentrated in middle layers, suggesting localized retrieval-related functionality.

ends at the assistant-role boundary; we do not append a synthetic `Think:` or `Thought:` cue, matching the prefill boundary at which AGMR records utilization during deployment. The same construction procedure is applied to Llama-3.1-8B-Instruct, producing model-specific retrieval supervision datasets for head scoring; the resulting datasets contain 47 Qwen and 34 Llama transition cases.

D.1 Retrieval Score and Head Distribution

Given the retrieval supervision dataset, we compute a Retrieval Score for every attention head. For a supervision pair (`key_context`, `formatted_prompt`), we examine the attention distribution

from the last token of `formatted_prompt` to all previous context tokens. For each supervision example, the task-level score is the attention mass assigned from the final prompt token to the annotated `key_context`. Summing this quantity over all supervision examples gives the retrieval score in Equation (2). A high Retrieval Score indicates that the head consistently assigns strong attention to memory snippets that are useful for the next decision step.

Figure 6 visualizes the Retrieval Score distributions for Qwen-2.5-7B-Instruct and Llama-3.1-8B-Instruct. In both models, only a small subset of heads obtains markedly high scores, suggesting that memory retrieval behavior is concentrated in a limited number of specialized heads rather than uniformly distributed across all heads.

We select the top-5 heads with the highest retrieval scores as the retrieval-head set $\mathcal{H}^\star$. Specifically, for Qwen2.5-7B-Instruct, we select $(17, 18)$, $(17, 19)$, $(19, 15)$, $(19, 17)$, and $(19, 22)$; for Llama-3.1-8B-Instruct, we select $(13, 6)$, $(14, 13)$, $(14, 29)$, $(14, 31)$, and $(13, 1)$, where each pair denotes $(\text{Layer}, \text{Head})$.

D.2 Empirical Selection and Diagnosis Stability

We assess empirical stability with three complementary checks. For each backbone, a bootstrap replicate resamples the N supervision pairs with replacement, recomputes all head scores, and records the Jaccard overlap between its top-five set and the full-data set; we report the mean and 2.5/97.5 percentiles over 10,000 replicates. We also form 1,000 random disjoint half-splits and report top-five overlap between halves. Finally, on every diagnosed failure, we independently perturb the two high-utilization multipliers (1.5 times the column mean and 0.5 times the global maximum) by $\pm 10\%$ and report the fraction of labels unchanged. Table 4 reports all three analyses.

Table 4: Empirical stability of retrieval-head selection and diagnosis thresholds.

Object	**Statistic**	**Result**
Qwen heads	Bootstrap top-5 Jaccard, mean [95% interval]	0.867 [0.667, 1.000]
Llama heads	Bootstrap top-5 Jaccard, mean [95% interval]	0.811 [0.667, 1.000]
Qwen heads	Disjoint-half Jaccard / overlap count	0.739 / 4.22 of 5
Llama heads	Disjoint-half Jaccard / overlap count	0.662 / 4.01 of 5
Diagnosis	Label unchanged on joint $\pm 10\%$ threshold grid	88.3% [77.8%, 94.2%]

Across random non-overlapping half-splits, the selected sets share 4.22 heads for Qwen and 4.01 heads for Llama on average. For Llama, using 25%, 50%, and 75% of its 34 supervision examples gives mean overlap with the full-data set of 0.723, 0.811, and 0.838, respectively. We use the same cumulative single-token decoding rule as the original Llama head-discovery run to map annotated character spans to tokens, thereby evaluating the deployed head set rather than reselecting it under a different tokenizer mapping. Thus, both configurations retain a stable retrieval-head core across supervision resamples.

For diagnosis stability, we de-duplicate candidate retries and recover 60 Qwen-ALFWorld construction failures whose saved prompts contain the complete intended segment, score vector, column mean, and global maximum. We recompute the deterministic priority rule over the Cartesian grid obtained by independently multiplying the two threshold coefficients by 0.9, 1.0, and 1.1. The label is unchanged at every one of the nine settings for 53/60 cases (88.3%, Wilson 95% CI [77.8%, 94.2%]); varying only the mean or global-maximum coefficient gives 96.7% and 91.7% stability, respectively.

Decision-level causal audit protocol. For the 53 jointly stable cases, we reconstruct the exact refinement-round memory and failed trajectory from the saved conversation log and replay Qwen-2.5-7B-Instruct at the reflector-localized erroneous decision. The positive target is the reflector's corrected Thought/Action response and the negative target is the recorded erroneous response; both are teacher-forced from the same prompt, and their token log probabilities are averaged separately before forming $\mathcal{M}$. An attention mask sets the selected segment edges to $-\infty$ before softmax and then renormalizes the remaining weights. The boost intervention instead adds $\log 4$ to those logits, giving the selected span four times its original attention odds before normalization. Both interventions cover every token in the segment and operate only in Qwen's five frozen retrieval

heads. The layer-matched random control uses heads $(17, 2)$, $(17, 5)$, $(19, 27)$, $(19, 19)$, and $(19, 3)$; the segment control selects the non-intended segment with the closest token length, with index order breaking ties. Confidence intervals resample cases 200,000 times with seed 20260918. Among the 16 stable accessed-but-ineffective cases, masking the intended segment changes $\mathcal{M}$ by -0.0394 through retrieval heads and -0.0020 through random heads; their mean absolute-effect contrast is 0.0449 (95% CI [0.0216, 0.0752]). A same-environment mask–boost comparison reproduces the mask effect (-0.0388 versus -0.0031 through random heads) and finds that boosting the intended segment increases $\mathcal{M}$ by 0.0354 on average, compared with 0.0025 through random heads. The boost is positive in 15/16 cases, and its paired advantage over the random-head boost is 0.0329 (95% CI [0.0205, 0.0451]). To examine whether an accessed segment supports the corrected response, the erroneous response, or both, we augment these cases with 18 threshold-stable, task-disjoint cases from two later construction batches, retaining the newer construction when a task appears in both. Across the resulting 34 cases, masking lowers the erroneous-response log probability in 18 cases and lowers both response probabilities in 13; 26/34 masks shift relative preference toward the recorded error. Among the 26 stable missed cases, boosting intended support changes $\mathcal{M}$ by 0.0335, compared with -0.0005 for the matched-segment control and 0.0083 through random heads. Repeating the same analysis on all 60 parsed cases, including the seven threshold-unstable cases, preserves the accessed preference-sensitivity contrast (0.0487, 95% CI [0.0269, 0.0756]) and the missed target-boost effect (0.0326 [0.0205, 0.0440]).

D.3 Ablation Validation of Retrieval Heads

To verify that the identified high-score heads are functionally important for memory-assisted behavior, we conduct an attention-head masking experiment. Evaluation is performed under the raw trajectory-memory setting, where the agent receives retrieved trajectory memory in the prompt. For Qwen-2.5-7B-Instruct, we mask the top 5 heads ranked by Retrieval Score and compare them with 5 randomly selected heads. For Llama-3.1-8B-Instruct, we analogously mask the top 5 retrieval heads and 5 random heads.

As shown in Figure 2, masking the high-Retrieval-Score heads substantially reduces the performance gains brought by trajectory memory. In contrast, masking randomly selected heads has little effect. This result indicates that the selected high-score heads are not merely correlated with useful memory snippets, but play an important functional role in extracting relevant information from memory. Therefore, we use the top five retrieval-sensitive heads of each backbone model as the basis for memory-utilization tracking in AGMR.

E Experiments Details

E.1 SciWorld Configuration

For SciWorld, we evaluate a predefined 147-instance subset comprising task types with `max_steps` < 60 to support exact attention extraction throughout execution. The excluded task types are:

- `task-1-boil`,
- `task-1-change-the-state-of-matter-of`,
- `task-1-freeze`,
- `task-1-melt`,
- `task-10-measure-melting-point-(known-substance)`,
- `task-4-grow-fruit`,
- `task-5-chemistry-mix`.

The identical subset is used for AGMR and all baselines, with no method-specific instance filtering.

E.2 Dataset, Retrieval, and Run Protocol

The SciWorld test list contains 147 instances after excluding the seven long-horizon task types above; no further method-specific filtering is applied.

For all memory conditions, retrieval embeds the current task description and every memory-entry task name with `BAAI/bge-large-en-v1.5`; embeddings use attention-mask-aware mean pooling, ℓ_2 normalization, and cosine similarity mapped from $[-1, 1]$ to $[0, 1]$. The system retrieves exactly the single highest-scoring trajectory (top-$k = 1$) when its mapped similarity is at least 0.5; otherwise it supplies no trajectory memory.

Segmentation is lossless: concatenating the segments exactly reconstructs the prompt. ALFWorld and SciWorld memory is divided into the instruction/task prefix followed by interaction blocks that normally combine one Observation with its following Thought/Action; fallback blocks preserve unmatched Observation or Thought/Action text. WebShop uses the analogous task prefix and interaction blocks for its `[SEP]`-delimited page format. The current-task suffix is never included among editable memory segments. Observation-token spans are excluded only when computing the deployed utilization score; the whole-segment causal intervention in Appendix E.7 includes them.

Offline construction traverses the training-task list in its stored order without shuffling and writes each accepted update before proceeding; the completed memory base is then frozen for test evaluation. The open-weight agent uses greedy decoding (temperature 0, top-$p = 0$, maximum 512 new tokens), so no sampling seed affects its actions. The Semantic Reflector and Refiner use `Gemini-3.1-pro-preview`, temperature 1.0, and high thinking.

Failure-branch verification permits at most three candidate/re-execution attempts in ALFWorld and SciWorld and two in WebShop; a failed sequence rolls back the entry.

E.3 Hyperparameters for Memory Refinement

We use benchmark-specific hyperparameters for utilization-aware simplification and re-execution verification. For successful-task simplification, a segment is considered weakly utilized according to the attention statistics defined in Section 3. For ALFWorld, we use a relative criterion based on each memory entry: a segment is treated as weakly utilized if

$$c_\ell^{\max} < 3 \cdot \mathrm{median}_k(c_k^{\max}) \quad \text{and} \quad \frac{c_\ell^{\max}}{\bar{c}_\ell} < 7.$$

This criterion removes segments whose peak attention remains near the entry-level median scale and whose utilization is not strongly concentrated around any specific decision step.

For SciWorld and WebShop, we use global thresholds after intersecting with segments judged semantically redundant by the Semantic Reflector. In SciWorld, the implementation computes $c^{\max}_{\text{global,capped}} = \min(c^{\max}_{\text{global}}, 1500)$ and retains candidates with $c_i^{\max} > 0.25\, c^{\max}_{\text{global,capped}}$; if more than three remain, it simplifies the three lowest-scoring candidates. This lower-bound filter avoids spending edits on effectively unread segments, while the final ordering favors the least-used candidates among those still measurably accessed. In WebShop, it retains candidates with $c_i^{\max} < 0.9\, c^{\max}_{\text{global}}$ and likewise keeps at most the three lowest-scoring candidates. We state these asymmetric domain rules explicitly because the SciWorld code is not equivalent to a simple "below 0.25" low-attention test.

For re-execution verification in the failure branch, we set the fixed maximum number of refinement attempts to 3 for ALFWorld and SciWorld, and 2 for WebShop. If no candidate update passes re-execution within this limit, AGMR rolls back to the original memory entry.

For verification purposes, an update is also accepted if the re-executed trajectory yields a reward 0.3 higher than that of the original trajectory. For WebShop, this threshold is set to 0.0.

E.4 Hyperparameter Sensitivity

The benchmark-specific values above were chosen from training-side pilot observations rather than the test results. Because reward scales and task difficulty differ, a new benchmark may require coarse calibration; we therefore vary one hyperparameter at a time on Qwen-SciWorld (Table 5). Across

all tested settings, AGMR remains 6.56–9.51 performance points above the Traj baseline of 67.69. Performance is best near the defaults but changes smoothly around them, suggesting that the result does not depend on a single brittle value.

Table 5: One-factor sensitivity on Qwen-SciWorld. Bold values are the defaults.

Hyperparameter	Value	Performance ↑
Utilization threshold	0.15	76.90
	0.25	**77.20**
	0.35	75.94
Reward-improvement threshold	0.0	76.13
	0.3	**77.20**
	0.5	74.25
Maximum refinement attempts	2	75.18
	3	**77.20**
	4	76.79
Traj baseline	–	67.69

E.5 Utilization-Definition Robustness

We compare the deployed score with variants that omit the prompt-length multiplier or average attention per segment token. These alternatives preserve the max-over-step segment ordering (median within-task rank correlations of 0.925–0.972) and yield nearly unchanged within-task associations with causal sensitivity ($\rho = 0.136$–0.140). Thus, the segment rankings and causal associations are stable across alternative score definitions.

The deployed trace observes the final prompt boundary because that state generates the first assistant token. As a query-scope robustness check, we teacher-force each recorded first-decision Thought/Action and average attention over the boundary and every subsequent query row that predicts one of its tokens; the key scope, five heads, and n_t multiplier remain fixed. This evaluates 89 successful tasks, 1,018 segments, and outputs of 26–64 tokens (median 42). The recomputed boundary ranks agree with the saved online trace (fixed-effect $\rho = 0.995$; median task-level $\rho = 1.000$), validating prompt reconstruction despite small floating-point differences between cached and full-sequence forward paths.

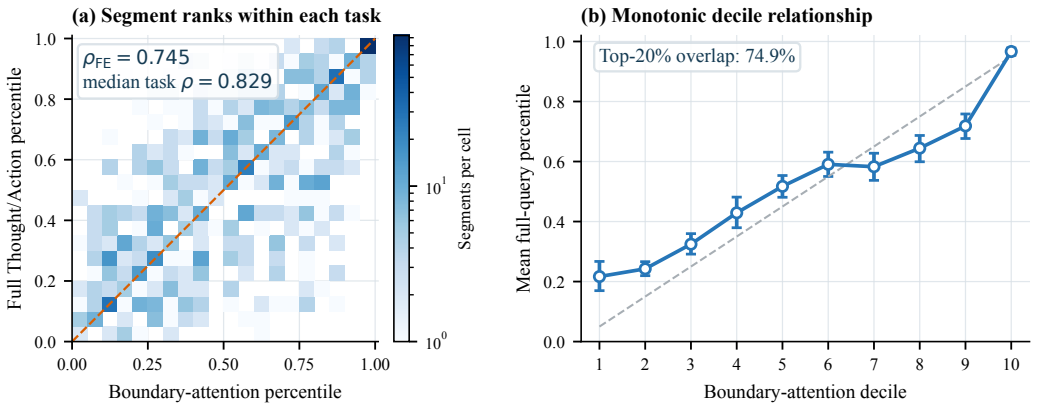


Figure 7: Boundary-query robustness on 89 Qwen–ALFWorld first decisions (1,018 segments). Each point in (a) is ranked within its task; darker cells contain more segments. Panel (b) gives equal weight to each task represented in a boundary-attention decile; error bars are 95% task-bootstrap intervals. The dashed lines denote equality, not fitted trends.

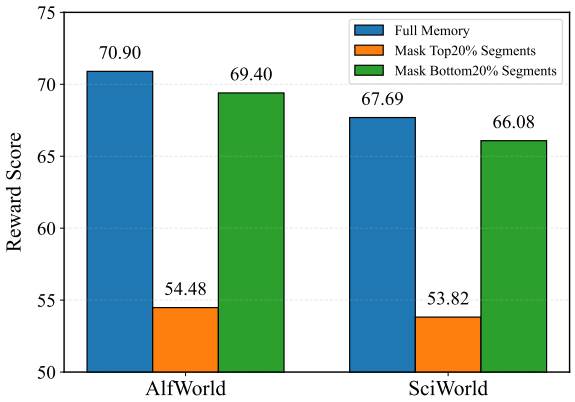


Figure 8: End-to-end top/bottom segment intervention. Removing the top 20% of segments under $u_i^{\max}$ causes a substantially larger performance drop than removing the bottom 20%.

The boundary and full-decision query scopes show substantial agreement: their fixed-effect rank correlation is 0.745 (95% task-bootstrap CI [0.668, 0.799]), the median task-level correlation is 0.829, and their top-20% segment overlap is 74.9% [69.7%, 80.0%]. Full-decision aggregation strengthens the fixed-effect correlation with whole-segment causal perturbations from 0.141 [0.016, 0.266] to 0.174 [0.078, 0.299]. Thus, the inexpensive boundary row provides an efficient online proxy while preserving the memory-utilization conclusion under full-decision aggregation.

E.6 Within-Task Attention-Quantile Intervention

We complement the top-versus-bottom masking result in Figure 8 with a finer within-task intervention. For each Qwen-ALFWorld memory, we rank segments by normalized maximum utilization, divide them into quintiles, and independently remove each quintile before re-executing the task. To make a post-intervention failure unambiguous, the analysis uses the 89 tasks for which the unmodified full-memory run succeeds and the memory contains at least five segments; all 445 task–quintile runs completed.

Table 6: Qwen-ALFWorld outcomes after masking each within-task attention quintile. Failure is measured relative to a successful full-memory run.

Masked segments	**Quintile**	**Success ↑**	**Failure ↓**
Lowest attention	Q1 (0–20%)	82.02	17.98
	Q2 (20–40%)	76.40	23.60
	Q3 (40–60%)	78.65	21.35
	Q4 (60–80%)	71.91	28.09
Highest attention	Q5 (80–100%)	70.79	29.21

The fitted failure-rate slope is 2.70 points per quintile (task-bootstrap 95% CI $[-0.11, 5.51]$; one-sided within-task permutation $p = 0.020$). Masking Q5 causes an 11.24-point higher failure rate than masking Q1 (task-bootstrap 95% CI $[0.00, 22.47]$; paired exact one-sided $p = 0.049$). This intervention shows that the attention ranking predicts aggregate behavioral sensitivity.

E.7 Per-Segment Causal Edge Intervention

The quantile experiment changes a group of segments and evaluates end-to-end task success. We additionally test every individual segment with a localized, equal-length intervention and a continuous same-decision outcome. We use the same 89 successful Qwen-ALFWorld full-memory tasks, comprising 1,018 segment interventions. For every task, the first assistant decision is fixed before inspecting attention or intervention outcomes. At that decision, we record last-query attention to each segment and block the relevant heads from attending to the *entire* target segment, including its observation tokens. No token is deleted: sequence length, positions, residual features, and all non-target attention edges are preserved. Conceptually, masking is applied *before softmax*: the selected

heads' target-key logits are set to negative infinity and softmax is recomputed, so probability mass is renormalized over the remaining causal keys. For compatibility with the model's optimized attention kernel, the hook implements this exactly as a projected output delta between the recomputed masked-softmax output and the original-softmax output; all unselected heads are unchanged. We validate the post-RoPE attention extractor against the model's native eager-attention output before intervention (maximum absolute error 7.1×10^{-4}).

Concretely, for recorded decision tokens $y_{1:T}$, we compute the teacher-forced mean negative log-likelihood

$$L = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(y_t \mid x, y_{<t}), \qquad \Delta L = L_{\text{masked}} - L_{\text{base}}.$$

A positive ΔL means that the segment supported the recorded decision; a negative value means that it opposed that decision. For this reason, signed ΔL tests directional support, whereas $|\Delta L|$ measures *causal sensitivity*: whether removing the read path changes the model's decision distribution in either direction. The latter is the more direct intervention counterpart of utilization, which does not assume that every accessed memory segment is beneficial.

We first mask exactly the five measured retrieval heads at layers 17 and 19 on all 89 tasks. The signed task-fixed-effect Spearman association is $\rho = 0.159$ (task-bootstrap 95% CI $[0.083, 0.244]$; one-sided within-task permutation $p < 10^{-4}$). The median task-level correlation is 0.250, 77.5% of tasks have a positive correlation, and the highest-attention quintile causes a 0.0106 larger signed NLL increase than the lowest quintile (95% CI $[0.0081, 0.0133]$). These directional results agree with the causal-sensitivity analysis.

To hold segment content and intrinsic importance fixed, we introduce a temporal contrast on the same segment. For every segment, we use its already-recorded AGMR trace to select the trajectory decision at which its pre-intervention utilization is maximal and the decision at which it is minimal. At each selected decision, we mask the same segment in exactly the five measured heads and evaluate $|\Delta L|$ for that decision. The selection depends only on the unperturbed attention trace, never on an intervention outcome; comparing the same segment also fixes its text and token length.

On the 45-task development split, the mean $|\Delta L|$ is 0.00787 at the maximum-attention decision and 0.00207 at the minimum-attention decision, a $3.80\times$ ratio (task-bootstrap 95% CI $[3.09, 4.71]$). The held-out results in Table 7 reproduce this difference. At the maximum-attention decision, masking a segment causes a mean perturbation of 0.00697, compared with 0.00215 at its minimum-attention decision, or $3.25\times$ larger (95% CI $[2.81, 3.77]$). The 95% CI for the held-out paired gap excludes zero. Moreover, the magnitude of the attention max–min gap predicts the causal-effect gap across held-out segment pairs (raw Spearman $\rho = 0.406$).

Table 7: Configuration-held-out same-segment temporal intervention on 537 paired segments from 44 tasks. "Ret.-selected" uses AGMR's trace to choose decisions but masks random heads; "own-selected" uses the random heads both to choose decisions and to mask. Brackets denote task-bootstrap 95% confidence intervals.

Statistic	Retrieval heads	Random, ret.-selected	Random, own-selected
Mean $\|\Delta L\|$ at maximum	0.00697	0.00190	0.00206
Mean $\|\Delta L\|$ at minimum	0.00215	0.00191	0.00178
Maximum/minimum ratio	3.25 [2.81, 3.77]	1.00 [0.92, 1.08]	1.16 [1.03, 1.29]
Maximum–minimum gap	0.00482 [0.00391, 0.00589]	−0.000003 [−0.00016, 0.00015]	0.00028 [0.00006, 0.00049]
Attention-gap/effect-gap raw ρ	0.406	−0.019	0.042

For completeness, we compute the proportion of tasks with a positive maximum–minimum absolute-effect gap over the full 89-task set, including both the development and held-out tasks. The proportions are 84/89 (94.4%) for retrieval heads, 46/89 (51.7%) for random heads at retrieval-selected decisions, and 55/89 (61.8%) for random heads at their own selected decisions. These full-set descriptive proportions are separate from the held-out effect estimates and confidence intervals in Table 7.

The directional outcome agrees with the sensitivity analysis. On the held-out tasks, mean signed ΔL is 0.00317 at the maximum-attention decision and -0.00015 at the minimum, giving a paired gap of 0.00332 (95% CI $[0.00223, 0.00470]$) that is positive for 88.6% of tasks. For random heads,

the corresponding gap is only 0.00020 and its interval includes zero ($[-0.00012, 0.00053]$); the retrieval-minus-random signed contrast is 0.00312 (95% CI $[0.00200, 0.00453]$; $p < 10^{-5}$). Thus, high attention predicts not only greater distributional sensitivity but also stronger directional support for the recorded decision.

Repeating the identical max/min selection while masking layer-matched random heads eliminates the temporal effect: the random-head ratio is 1.00, the max–min gap is -0.000003 (95% CI $[-0.00016, 0.00015]$), and the attention-gap/effect-gap correlation is -0.019. We also run a symmetric control in which layer-matched random heads select their own maximum- and minimum-attention decisions and are then masked at those decisions. This produces only a small absolute-effect gap of 0.00028 (ratio 1.16; 95% CI for the gap $[0.00006, 0.00049]$), with an attention-gap/effect-gap correlation of 0.042. Thus, masking high-attention edges has a small generic structural effect, but it is far too small to explain the retrieval-head temporal contrast; the trace identifies *when the same memory segment is causally read through the measured retrieval pathway*, rather than merely ranking segments by static content or length.

Because retrieved information can be propagated through consecutive blocks, we perform a development-set circuit-width analysis on the first 45 tasks. We retain the original five-head attention score as the predictor and compare exact five-head masking, top-10 masking, a 15-edge local neighborhood (the corresponding head index in the preceding, selected, and following block), and a wider 25-edge neighborhood. Using within-task $|\Delta L|$ correlation as the development criterion, the respective correlations are 0.296, 0.294, 0.389, and 0.373. We therefore lock the 15-edge intervention and evaluate it on the remaining 44 tasks, which were not used for configuration selection.

Table 8: Configuration-held-out per-segment causal sensitivity. Each retrieval or random circuit is scored by its corresponding five central heads and masks the entire segment in an equal-width 15-edge neighborhood. Quintile gaps average task-specific contrasts, giving each task equal weight; the overall mean effects pool all segment interventions.

Statistic	**Retrieval heads**	**Random heads**	**Contrast**
$\|\Delta L\|$ task-FE Spearman	0.296	0.009	0.287
FE corr. with raw $\|\Delta L\|$	0.357	-0.003	0.360
Positive-correlation tasks	93.2%	52.3%	40.9 points
Q5–Q1 $\|\Delta L\|$	0.0087	0.0001	0.0086
Mean $\|\Delta L\|$	0.00401	0.00246	1.63×

On the held-out tasks, the selected circuit obtains $\rho = 0.296$ (task-bootstrap 95% CI $[0.228, 0.376]$; one-sided within-task permutation $p < 10^{-4}$), with a median per-task correlation of 0.357 and positive correlations in 93.2% of tasks. Its task-averaged quintile means increase only slightly from 0.00207 in Q1 to 0.00295 in Q4, before rising sharply to 0.01075 in Q5. The corresponding task-averaged Q5–Q1 gap is 0.00869 (95% CI $[0.00618, 0.01162]$). The magnitude of ρ should be interpreted together with this quintile profile. Spearman correlation measures global monotonic ranking over all segments, whereas the observed relationship is strongly top-heavy: Q1–Q4 form a broad plateau and the substantial separation appears in Q5. This plateau attenuates the global rank correlation even though the highest-utilization group is substantially more sensitive to intervention. Accordingly, this experiment supports the trace as a coarse selector of highly utilized segments, rather than as a precise causal ranking of segments throughout the low- and medium-utilization range. The size-matched random circuit is null ($\rho = 0.009$, 95% CI $[-0.086, 0.114]$; $p = 0.434$), and its Q5–Q1 gap is also null. The paired selected-minus-random correlation difference is 0.287 (task-bootstrap 95% CI $[0.185, 0.393]$), while the selected circuit produces 1.63 times the mean absolute effect (95% CI $[1.44, 1.85]$). Thus, on Qwen–ALFWorld, high retrieval-head attention predicts greater causal sensitivity of the same decision specifically through the measured read pathway; the result is not explained by segment length, masking more edges alone, or the matched random-head control. The cross-backbone and cross-benchmark results evaluate end-to-end utility, but do not independently replicate this decision-level causal audit for every model–environment pair.

E.8 Sequential Online Refinement

The main experiments use an offline-refined memory base to keep all test conditions fixed. To test whether AGMR can continue evolving during deployment, we additionally process the Qwen-ALFWorld test tasks sequentially, starting from the offline AGMR memory. Each task is scored before its resulting update can affect later tasks: failed interactions invoke the ordinary diagnosis, update, and verification procedure, while successful trajectories are appended to the memory library and may be retrieved only by subsequent tasks. Table 9 reports the resulting deployment-order run.

Table 9: Sequential online refinement on Qwen-ALFWorld.

Setting	Success ↑	Steps ↓
Traj memory	70.90	19.36
Offline AGMR	81.34	16.08
Offline initialization + online AGMR	**84.33**	**15.66**

Relative to the fixed offline memory, sequential updating adds 2.99 success points and reduces the average trajectory by 0.42 steps.

E.9 Computational and API Cost

Retrieval-head selection is a one-time operation per backbone. During memory evolution, the utilization trace is collected from the decoding pass already used to produce the action, so it requires no additional model forward pass under our implementation. At each decision, only the last-query attention vectors for five heads need to be retained before segment aggregation: at a 32K-token context in float32, this is $5 \times 32{,}768 \times 4 = 655{,}360$ bytes, or 0.625 MiB, before small metadata overheads. Score aggregation is negligible relative to model inference; the main additional cost is the two external calls per refinement attempt, one for semantic reflection and one for attention-guided editing, plus any re-executions required for verification.

Table 10: Total external LLM calls during memory evolution, including retries and candidates that are eventually rolled back. Traj performs no memory evolution.

Benchmark	Traj	AWM	DC	AGMR
SciWorld	0	73	194	192
WebShop	0	181	400	357
ALFWorld	0	74	268	273

AGMR's call count is close to DC (two fewer on SciWorld, 43 fewer on WebShop, and five more on ALFWorld), but is higher than AWM because the latter uses a more compressed workflow-extraction procedure. This cost is paid during memory construction; subsequent evaluation reuses the resulting memory, whose token counts are reported in Table 2. Across the reported runs, Qwen-7B and Llama-8B use less than 24 GB of peak VRAM, and end-to-end AGMR runtime ranges from 4 to 16 hours depending on the model, benchmark, and API latency (Appendix H).

E.10 Other Implementation Details

The initial trajectory-memory base used for AGMR offline refinement is extracted from the SFT dataset released by ETO Song et al. (2024). All memory entries are drawn from training tasks, and the refined memory base is then evaluated directly on the test set.

F Scenario-Wise and Qualitative Analyses

F.1 Scenario-Wise Performance and Diagnosis Frequency

We group Qwen–ALFWorld memory entries according to the three failure patterns diagnosed during refinement and evaluate tasks associated with each group. Figure 9 shows that AGMR's advantage

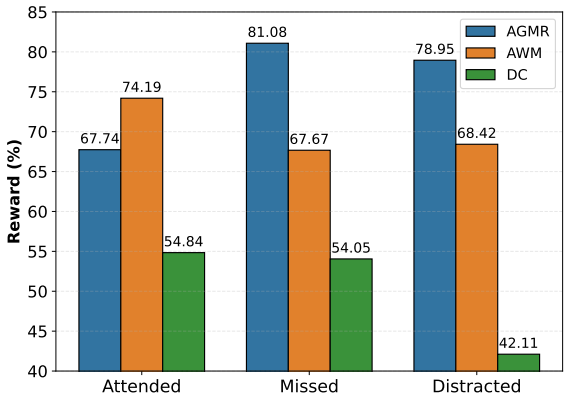


Figure 9: Scenario-wise performance on Qwen–ALFWorld.

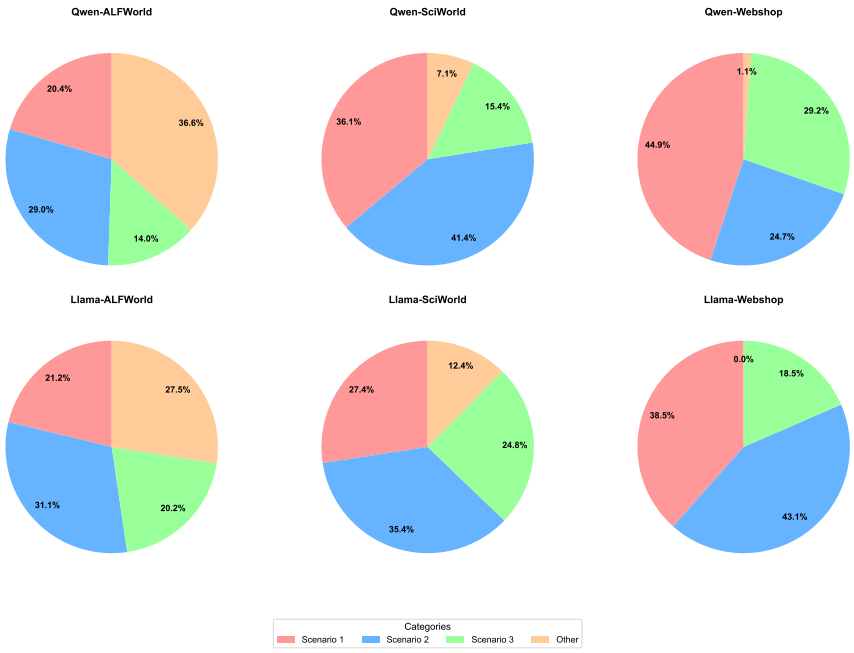


Figure 10: Distribution of diagnosed failure patterns. Scenario 1, 2, and 3 denote accessed-but-ineffective, missed relevant memory, and distracted by irrelevant memory, respectively. *Other* contains cases outside these recurring patterns.

over AWM and DC is concentrated in *missed relevant memory* and especially *distracted by irrelevant memory*. This distinction clarifies the complementary role of utilization evidence. Ineffective content can often be handled through semantic abstraction or rewriting, whereas missed and distracting memory depend on whether intended support was accessed and whether other segments competed for attention.

Figure 10 further shows that utilization-specific failures are not isolated examples. The three recurring patterns together account for 63.4%–100% of failed cases across the six model–benchmark settings, although their prevalence varies by environment and backbone.

The controlled interventions in Figure 5 link these recurring diagnoses to their corresponding mechanisms: accessed targets affect the recorded decision when removed, missed targets selectively improve the corrected response when strengthened, and action-conflict distractors improve it when suppressed. Appendix D.2 gives the complete protocol, uncertainty estimates, and matched-control results.

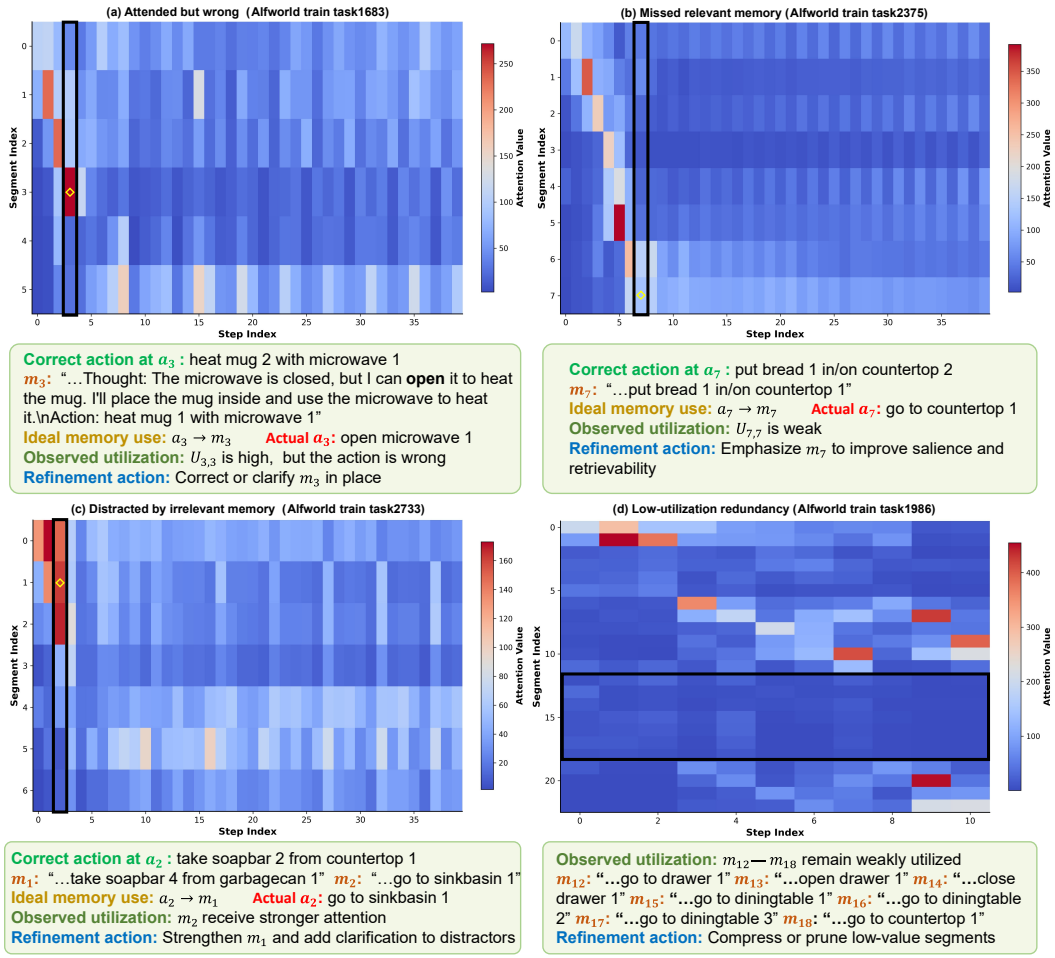


Figure 11: Representative pre-refinement memory-use traces: (a) accessed but ineffective, (b) missed relevant memory, (c) distracted by irrelevant memory, and (d) low-utilization redundancy. Black boxes mark critical error-step columns and diamonds mark semantically intended memory use.

F.2 Qualitative Before-and-After Analysis

Figure 11 expands the critical-decision slices in Figure 1 into their full segment-by-decision traces and additionally includes a successful simplification case. Here, m_i denotes the i-th memory segment, a_t denotes the decision at step t, and $u_{i,t}$ is the corresponding entry of the memory-use trace $\mathbf{U}$. Black boxes mark the critical decision columns, and diamonds mark the memory segments identified by semantic reflection as intended support.

Accessed but ineffective. In Figure 11(a), the first critical error occurs at a_3. The correct action is `heat mug 2 with microwave 1`, and the intended segment m_3 contains a `heat` action; however, the agent instead produces `open microwave 1`. The large value of $u_{3,3}$ shows that m_3 is strongly accessed at this decision, so the failure is not explained by a lack of access. The same segment also contains wording about opening the microwave, which is aligned with the erroneous action. Taken together, the text, output, and utilization pattern are consistent with an accessed-but-ineffective content problem, motivating an in-place correction or clarification of m_3.

Missed relevant memory. In Figure 11(b), the agent should use m_7 at a_7 to produce `put bread 1 in/on countertop 2`, but instead produces `go to countertop 1`. The trace contains strong aligned entries at $u_{5,5}$ and $u_{6,6}$, but $u_{7,7}$ is weak at the critical decision. This pattern is consistent with relevant support being present but insufficiently accessed when needed. AGMR

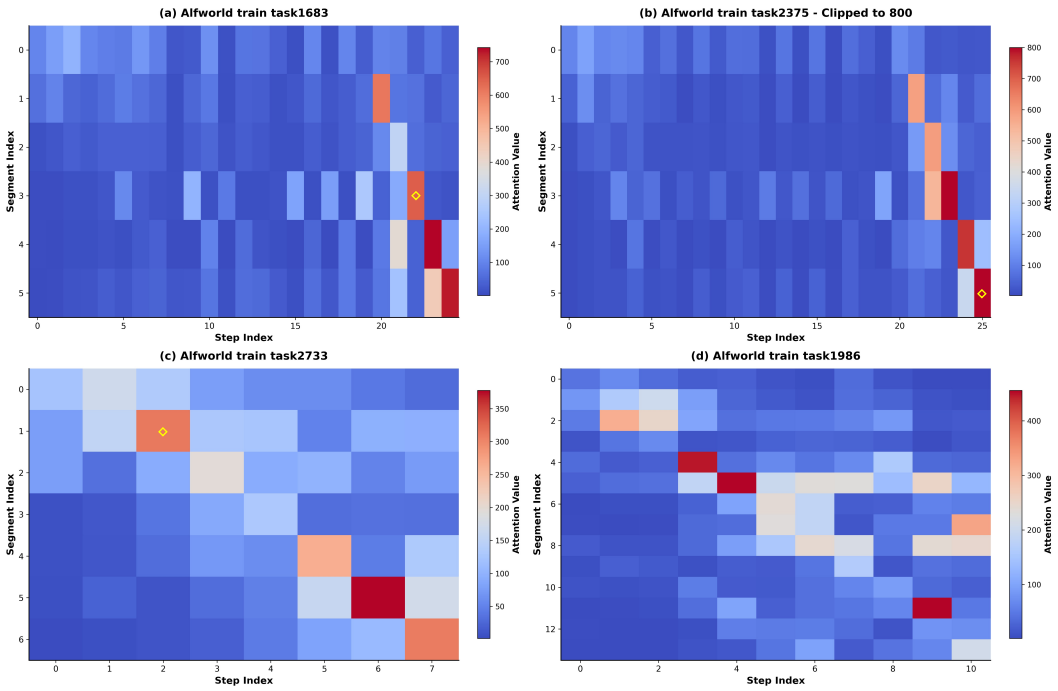


Figure 12: Memory-use traces after AGMR; diamonds denote semantically intended memory use.

therefore rewrites the intended segment with clearer task conditions and action cues to improve its salience and retrievability.

Distracted by irrelevant memory. In Figure 11(c), the correct action at a_2 is `take soapbar 2 from countertop 1`, for which semantic reflection identifies m_1 as intended support, whereas the agent produces `go to sinkbasin 1`. At the same decision, the competing segment m_2 receives stronger utilization than m_1 and contains the same `go to sinkbasin 1` action as the erroneous output; another segment also contains a similar `go to` pattern. This combination is consistent with competition from irrelevant memory. The corresponding text edit strengthens the conditions and action cues in m_1 while clarifying the scope of the competing segment so that its guidance is less likely to apply in the wrong context.

Low-utilization redundancy. Figure 11(d) shows a successful execution in which m_{12}–m_{18} remain weakly utilized throughout the trajectory. These segments mainly record low-value exploratory actions such as `go to {recep}`. For each segment, AGMR summarizes the trace by its average utilization $\bar{u}_i = T^{-1} \sum_t u_{i,t}$ and peak utilization $u_i^{\max} = \max_t u_{i,t}$, and combines these statistics with the Reflector's semantic redundancy judgment. In this example, the weakly utilized exploratory segments become simplification candidates; the exact benchmark-specific filtering and ranking rules are given in Appendix E.3.

Figure 12 revisits the same cases after refinement. Because editing the memory can change the agent's exploratory trajectory, the post-refinement critical decision need not have the same step index as before refinement.

In the accessed-but-ineffective case, AGMR corrects the problematic content in m_3 while preserving access to that segment. Figure 12(a) shows strong utilization of m_3 at the corresponding decisive step, now a_{22}; the agent produces the correct `heat` action and completes the task two steps later. In the missed-memory case, AGMR rewrites the target segment to improve its salience. After simplification, the original m_7 becomes m_5; Figure 12(b) shows strong utilization $u_{5,25}$ when the agent produces the corrected action and subsequently completes the task. In the distraction case, Figure 12(c) shows utilization shifting toward the intended segment m_1 at a_2, while the former distractor m_2 no longer dominates that decision. Finally, Figure 12(d) illustrates utilization-aware simplification: the memory is reduced from 23 to 13 segments, producing a denser trace with fewer segments that

remain unused throughout execution. Together, the before-and-after cases illustrate the complete read–write loop: $(M, \tau, y) \to (r, I_t) \to \mathbf{U}_{:,t} \to d_t \to \widetilde{M} \to M'$.

These paired cases make the diagnosis-to-edit interface concrete and show that, in these examples, AGMR's natural-language edits are accompanied by the predicted changes in memory-use traces. Semantic analysis supplies the intended support and corrected content; the observed trace determines whether the updater repairs that content, increases its salience, disambiguates competing content, or conservatively compresses it; re-execution then decides whether the candidate is committed. These representative cases provide qualitative evidence rather than establishing that textual edits are causally equivalent to the controlled internal interventions. The factor-matched comparison in Table 3 evaluates value beyond semantic reflection, while Figure 5 probes the accessed, missed, and action-conflict distraction components of the diagnosis interface under controlled internal interventions.

G PROMPT TEMPLATES

Failure-branch Semantic Reflector prompt template, illustrated with SciWorld tasks.

```
[Agent Action Trajectory]
omitted
[Successful Memory Trajectory (Expert Trajectory)]
omitted
*Task Description*

You are given two trajectories for the same ScienceWorld laboratory
↪  task:

- Agent Action Trajectory: a step-by-step sequence of Thought-Action
↪  pairs produced by an agent, which ultimately fails to complete
↪  the task.
- Successful Memory Trajectory (Expert Trajectory): a reference
↪  trajectory in which the task is successfully completed.

Your task is to perform a structured failure analysis by aligning the
↪  Agent Action Trajectory with the Expert Trajectory.

The objective is:

1. Construct an Optimal Trajectory based ONLY on the Expert
↪  Trajectory.
2. Identify the FIRST erroneous step in the Agent Action Trajectory
↪  via stepwise alignment with the Optimal Trajectory.
3. Provide the Corrected Step that the agent should have taken,
↪  strictly grounded in the Expert Trajectory.
4. Identify the step in the Expert Trajectory that corresponds to the
↪  Corrected Step, and output it as the Expert Trajectory Reference.

------------------------------------------------------------

Section 1 (MANDATORY): OPTIMAL TRAJECTORY CONSTRUCTION

Before performing any failure analysis, you MUST first construct an
↪  Optimal Trajectory.

Definition:
The Optimal Trajectory is the minimal canonical execution sequence
↪  that:

- successfully completes the task
- contains no mistakes or redundant steps
```

```
- follows the true logical task order (not environment-specific
↪  details)

*Core Rule (CRITICAL)*
The Optimal Trajectory MUST be derived exclusively from the
↪  Successful Memory Trajectory (Expert Trajectory).

This means:
- The Optimal Trajectory should follow the exact action flow of the
↪  Successful Memory Trajectory (Expert Trajectory).
- The Optimal Trajectory should closely mirror the sequence of
↪  actions in the Expert Trajectory.
- The only permitted modification when constructing the Optimal
↪  Trajectory is removing task-specific or environment-specific
↪  details
- The Optimal Trajectory should be viewed as a cleaned and
↪  generalized version of the Expert Trajectory, not a newly
↪  reasoned solution.
- You MUST NOT use the Agent Action Trajectory as a reference for
↪  constructing the Optimal Trajectory. Because the Agent Action
↪  Trajectory represents a failed attempt and may contain incorrect
↪  reasoning or actions.

Output Requirement

The Optimal Trajectory represents the perfect reference execution.
All subsequent failure analysis MUST be performed exclusively
↪  relative to this trajectory.

Example:
- Step 1: Teleport to the workshop and locate the unknown substance
↪  T.
- Step 2: Pick up the unknown substance T and focus on it.
- Step 3: Connect the battery anode to the first wire (e.g., green
↪  wire terminal 1).
- Step 4: Connect the battery cathode to the second wire (e.g., red
↪  wire terminal 1).
- Step 5: Connect the other end of the first wire (green wire
↪  terminal 2) to a terminal of a light bulb (e.g., cathode).
- Step 6: Connect the other terminal of the light bulb (anode) to a
↪  third wire (e.g., orange wire terminal 2).
- Step 7: Connect terminal 1 of the unknown substance T to the second
↪  wire (e.g., red wire terminal 2).
- Step 8: Connect the other end of the third wire (green wire
↪  terminal 1) to terminal 2 of the unknown substance T.
- Step 9: Wait and observe if the light bulb turns on.
- Step 10: Move the unknown substance T to the green box if it is
↪  conductive (bulb on), or to the blue box if it is nonconductive
↪  (bulb off).

------------------------------------------------------------

Section 2: Erroneous Step Localization Objective

You must identify the FIRST erroneous step via sequential alignment.

The Erroneous Step is defined as:
- the earliest point where the agent's thought or action deviates
↪  from the Optimal Trajectory.
Only the first deviation may be selected.

Erroneous Step localization must be performed through sequential
↪  alignment.
```

```
Procedure
- Start from Step 1 of the Optimal Trajectory.
- Check whether the agent correctly completes this step.
- If correct, move to the next step.
- Stop when the first mismatch between the agent behavior and the
↪  required next step appears.

This step is the Erroneous Step.

Key Principle
After completing Step i, the agent should attempt Step i+1.
Any thought, action, or decision that does not aim to complete Step
↪  i+1 is considered a deviation.

A step is also an Erroneous Step if the agent makes a critical
↪  mistake that prevents task success, including:
- a key misjudgment that blocks a viable path to completion
- violating a required output format so the environment cannot
↪  interpret the response

If, after performing the stepwise alignment procedure, you cannot
↪  determine the exact step at which the first deviation occurs, or
↪  you have no clear idea about the failure reason, you MUST skip
↪  the full failure analysis and output exactly:

[Object not found]

This is the only allowed alternative to the STRICT OUTPUT FORMAT
↪  below.

------------------------------------------------------------

Section 3: Corrected Step And Expert Trajectory Reference

The Corrected Step represents the exact action the agent SHOULD have
↪  taken at the failure point to remain consistent with the Optimal
↪  Trajectory.

The Expert Trajectory Reference Thought-Action pair MUST be the step
↪  in the Expert Trajectory that corresponds to, or most directly
↪  informs, the Corrected Step.

------------------------------------------------------------

STRICT OUTPUT FORMAT (NO DEVIATION)

Your output MUST strictly follow the format below.
Do NOT add, remove, or reorder sections.
If the Special Case applies, output ONLY: Object not found.

[Optimal Trajectory]
- Step 1: <description>
- Step 2: <description>
- ...

[Step-by-Step Failure Reasoning]
<Sequential comparison between the agent trajectory and the Optimal
↪  Trajectory,
ending exactly at the first detected deviation>

[Erroneous Step]
Thought: <original agent thought>
```

```
Action: <original agent action>

Failure Reason:
<Why this step violates the Optimal Trajectory and what should have
↪  happened instead>

[Corrected Step]
Correct Thought: <thought aligned with the Optimal Trajectory>
Correct Action: <action aligned with the Optimal Trajectory>

[Expert Trajectory Reference]
Referenced Thought: <expert thought>
Referenced Action: <expert action>
```

Failure-branch Attention-Guided Refiner prompt template, illustrated with SciWorld tasks.

```
You are given the following inputs from a failed ScienceWorld task
↪  execution:

- [Erroneous Step Context]: a trajectory excerpt containing the
↪  failure point

- [LLM Analysis Response], which includes:
  - [Erroneous Step]: the critical step that caused task failure
  - Failure Reason
  - [Corrected Step]: the step that should have been executed instead
  - [Expert Trajectory Reference] (segment[8]): the expert segment
   ↪  that defines the correct behavior

- [Attention Score Distribution]: the LLM's attention weights over
↪  Expert Trajectory segments at the moment the Erroneous Step was
↪  produced

The task is identical or highly similar to the expert trajectory. The
↪  failure point and its correct reference have already been
↪  identified.

Your task is to:

1. Analyze the attention distribution to determine which segments the
↪  LLM actually relied on when producing the erroneous Thought and
↪  Action.
2. Explain how this attention allocation led to the incorrect
↪  reasoning and behavior based on LLM Analysis Response and
↪  Erroneous Step Context.
3. Modify the relevant segment to prevent similar failures in future
↪  executions.

[Erroneous Step Context]
From [Task Trajectory]. Context window includes up to 3
↪  Thought/Action/Observation groups immediately before the
↪  Erroneous Step.
Context window: 3 step(s) before the Erroneous Step.
omitted

=== LLM Analysis Response ===
omitted

================================================
```

```
[Attention Score Distribution]
At the erroneous step (step 8), the LLM's attention distribution
↪  across memory segments was:
omitted

================================================

[Attention Analysis Guidelines]

Calculate the average attention score: 63.0237
Expert segment[8] received attention score: 242.4957
Max attention score at erroneous step: 242.4957 (at segment[8])
Global maximum attention score in full matrix: 407.8840
High attention definition: score > 1.5Œ average AND score > 0.5Œ
↪  global maximum
High attention thresholds: > 94.5356 (1.5Œ average) and > 203.9420
↪  (0.5Œ global maximum)

Based on the attention distribution, determine which scenario
↪  applies:

**Scenario 1: High attention to expert segment, but still failed**
- If segment[8] has high attention ( > 1.5Œ average AND > 0.5Œ global
↪  maximum )
- AND no other segment has comparably high attention
- This suggests the LLM did reference the expert segment but it may
↪  contain errors or insufficient emphasis
- Action: Modify segment[8] based on the Analysis Response to correct
↪  any errors and strengthen the guidance

**Scenario 2: Low attention to expert segment, no strong attention
↪  elsewhere**
- If segment[8] does NOT satisfy the high-attention definition
- AND no other segment meets the high attention definition (all
↪  scores relatively uniform)
- This suggests the LLM failed to retrieve or recognize the expert
↪  guidance, and no strong competing signal existed.
- Action: Modify segment[8] to make it more prominent and easier to
↪  retrieve (add emphasis, keywords, clearer structure)

**Scenario 3: Other segments have high attention (attention
↪  competition / distraction)**
- Regardless of whether segment[8] itself has high or low attention,
- If one or more segments other than segment[8] meet the
↪  high-attention definition ( > 1.5Œ average AND > 0.5Œ global
↪  maximum ), especially when those high-attention non-expert
↪  segments involve the same action type as the one taken in the
↪  [Erroneous Step] (e.g., both involve a take action)

- This suggests the LLM was misled by irrelevant segment[s] that
↪  competed for attention
- High-attention segments (excluding expert segment): segment[9]
↪  (208.4000)
- Action: Modify BOTH the expert segment AND the misleading
↪  segmen[s]:
  - Strengthen segment[8] to make it more relevant and prominent
  - Modify the misleading segment[s] to reduce irrelevant information
  ↪  or clarify their scope

[Important Notes]

- The attention score distribution may vary in complex ways beyond
↪  these three scenarios
```

```
- Use your judgment to analyze the actual attention pattern
- Focus on understanding WHY the LLM failed at the mechanism level
- Modify segments to address the root cause identified through
↪  attention analysis

==================================================

[OUTPUT FORMAT - STRICT, NO DEVIATION]

Step 1: First, provide your step-by-step analysis

[Step-by-Step Analysis]
1. Attention Pattern Analysis:
   <Analyze which scenario applies based on the attention scores>

2. Root Cause Identification:
   <Identify why the LLM made the error at the mechanism level, and
   ↪  determine WHICH segment(s) require modification as a direct
   ↪  result of this error>

3. Modification Strategy:
   <Explain HOW the identified segment(s) will be modified.
   The strategy should primarily focus on the erroneous *Thought* in
   ↪  the [Erroneous Step].
   It should describe how the incorrect reasoning or intention
   ↪  expressed
   in that Thought can be corrected, constrained, or redirected,
   with the explanation grounded in both the attention pattern
   ↪  analysis
   and the error analysis provided in the === LLM Analysis Response
   ↪  ===.>

Step 2: Then, output the modified segments

For EACH segment you modify, output the modified content in the
↪  following format:

enhanced_segment[SEGMENT_ID]: <modified segment content>

Before the "[/INST] Thought" section in enhanced_segment, you must
↪  output a text segment in the following format:
"*IMPORTANT NOTE*: I MUST <do the correct action> In this
↪  environment... If the agent performs <wrong action>, the episode
↪  immediately fails."


Example:
enhanced_segment[3]: [INST] Observation: This outside location is
↪  called the outside. Here you see:
        the agent
        a substance called air
        a blue jay egg
        a giant tortoise egg
        a substance called wood
You also see:
        A door to the greenhouse (that is open)
```

```
        A door to the kitchen (that is open) *IMPORTANT NOTE*: I MUST
        ↪  enter the menu index (e.g., Action: 0) if "Ambiguous
        ↪  request: Please enter the number for the action you
        ↪  intended" appears. In this environment, if a numbered
        ↪  list appears, repeating the text command causes an error.
        ↪  If the agent performs a text repetition, the episode
        ↪  immediately fails. [/INST] Thought: I see several eggs
        ↪  outside, which are animals. I will focus on the blue jay
        ↪  egg. If the ambiguous prompt appears, the Action must
        ↪  output 0 instead of repeating the command.
Action: focus on blue jay egg

------------------------------------------------
Formatting rules:
- Do NOT use quotation marks
- Preserve the Observation/Thought/Action structure from the original
↪  segment
- Do NOT output anything else besides the [Step-by-Step Analysis]
↪  section and enhanced_segment lines
- When writing enhanced_segment content, prefer simple, direct words
↪  and sentences.
- Use concise, command-style wording whenever possible to clearly
↪  specify the required action.
- Avoid long or complex explanations inside enhanced_segment.
================================================
Length Constraint for each enhanced_segment:
- Apply minimal edits.
- Do not significantly increase the segment length.
- Keep enhanced_segment within 150% of the original segment length.
- Avoid redundant explanations.
```

Call 1: redundancy analysis. The placeholders below are replaced by the complete successful agent trajectory and every indexed segment of the retrieved successful trajectory. Whitespace and repeated formatting reminders are normalized for print; the decision rules, constraints, and output schema are complete.

```
{AGENT_SUCCESSFUL_TRAJECTORY}

[Successful Memory Trajectory (Expert Trajectory)]
Reminder: segment indices start from 0, not 1.

{ALL_INDEXED_MEMORY_SEGMENTS}

Task Description

You are given two trajectories for the same successfully completed household task:

- Agent Action Trajectory: a step-by-step interaction sequence produced by an
  agent that successfully completes the task.
- Successful Memory Trajectory (Expert Trajectory): a reference trajectory
  segmented into multiple segments, where each segment represents a contiguous
  Thought-Action block.

Both trajectories lead to successful task completion.

Redundancy Analysis Objective

Your goal is to identify redundant segments in the Successful Memory Trajectory
(Expert Trajectory) by evaluating their necessity and instructional value
relative to the Agent Action Trajectory.

Global Constraint
- Segment[0] MUST NEVER be included in the Redundant Segments list, regardless
  of its content or apparent redundancy.
- It is VALID that no segments are redundant. In this case, output:
```

```
  [Redundant Segments]
  Segment Indices: []

A segment (or a set of tightly coupled segments) is considered REDUNDANT iff:
- Removing it would NOT prevent successful task completion; and
- It is not a necessary subgoal, state transition, or decision point; and
- Compared with the Agent Action Trajectory, it provides LOW reference value
  for learning an effective policy.

Analysis Procedure
1. Decompose the task into its minimal required subgoals and necessary steps.
2. For each Expert segment ask: if removed, would the remaining trajectory
   still be sufficient to reach the final goal?
3. Identify every segment satisfying the redundancy criteria.

STRICT OUTPUT FORMAT
[Task Decomposition]
- Required Subgoal 1: <description>
- ...

[Redundant Segments]
Segment Indices: [<i>, <j>, <k>]

[Justification]
Segment <i>: <concise reason>
...
```

Call 2: utilization-constrained modification. The implementation inserts Call 1's complete response, every segment's max/average score, the median peak score, and the two precomputed index lists.

```
=== Redundancy Analysis Result ===
{CALL_1_RESPONSE}

[Attention Score Analysis]
For each segment, we calculated:
- max_attention_score: maximum attention value across all steps
- avg_attention_score: average attention value across all steps
{ALL_INDEXED_SEGMENTS_AND_SCORES}
median_max_attention_score: {MEDIAN}

[Task Description]
You are given:
1. The redundancy analysis identifying redundant Expert segments
2. The attention score distribution during task execution

For each eligible segment, choose SIMPLIFY, DELETE, or NO ACTION.
A segment may be SIMPLIFIED or DELETED only if:
- It was identified as redundant; AND
- max_attention_score < 3 * median_max_attention_score; AND
- max_attention_score / avg_attention_score < 7.

[Allowed Operations]
1. SIMPLIFY: retain residual instructional/structural value more concisely.
2. DELETE: remove a segment with no meaningful value.
3. NO ACTION: use when the eligibility conditions do not hold.

[Analysis Instructions]
Redundant Segment Indices: {REDUNDANT_INDICES}
Precomputed eligible Segment Indices: {ELIGIBLE_INDICES}
Apply SIMPLIFY or DELETE to every eligible segment and NO ACTION to all others.

[Simplification Constraints]
- Remove unnecessary observations and verbose descriptions.
- Keep essential information (core action and minimal context).
- Preserve the Observation/Thought/Action structure and correctness.

If deleting a segment makes the immediately following Observation or Thought
inconsistent with the preceding Action, you may minimally correct that
Observation/Thought, but must copy its Action verbatim.

[OUTPUT FORMAT -- STRICT]
[Analysis]
<reasoning>
simplified_segment[ID]: <full simplified segment>
DELETE(segment[ID])
modified_segment[ID]: <full segment; Action verbatim>

If no operation is performed, output exactly:
No segment that needs to be simplified
```

For SciWorld, Call 1 receives only the indexed expert trajectory and selects all nonzero-index low-value or observation-heavy candidates; Call 2 preserves every Action verbatim and simplifies at most three semantically redundant candidates passing the capped-score rule in Appendix E.3. For WebShop, Call 1 additionally emits `<mask_observation>` whenever an explicit Observation is present; Call 2 copies `Observation: <masked>` and Actions verbatim and simplifies at most three candidates passing the WebShop threshold. These domain-specific static templates are included in the released implementation alongside the exact template above.

H COMPUTATION RESOURCES

All experiments are conducted on NVIDIA RTX PRO 6000 GPUs with 97887MiB memory. The peak GPU memory consumption varies with the model size. Specifically, Qwen-7B and Llama-8B generally require less than 24 GB of VRAM. The runtime of AGMR ranges from 4 to 16 hours, which depends on the backbone model, benchmark task, and API response speed.