MEM-π: ADAPTIVE MEMORY THROUGH LEARNING WHEN AND WHAT TO GENERATE

Anonymous authors
Paper under double-blind review

ABSTRACT

We present Mem-π, a framework for adaptive memory in large language model (LLM) agents that *generates guidance on demand* rather than retrieving it from an external memory store. Existing memory-augmented agents typically rely on similarity-based retrieval from episodic memory banks or skill libraries, returning static entries that often misalign with the current query context. We instead realize memory as a dedicated language or vision-language model, separate from the downstream agent, that is fine-tuned to produce context-specific guidance for complex tasks. Using a decision-content decoupled reinforcement learning (RL) objective, the memory policy learns *when* to intervene and *what* to generate. It abstains when guidance would not help and otherwise produces concise, task-relevant guidance. Across agent benchmarks in web navigation, terminal tool use, and embodied environments, Mem-π improves by 14.6% on average relative to the strongest external baseline on each benchmark.

1 INTRODUCTION

Large language models (LLMs) (Ouyang et al., 2022; Team et al., 2023; Dubey et al., 2024; Hurst et al., 2024; DeepSeek-AI, 2026) increasingly serve as autonomous agents (Liu et al., 2025a), but they are stateless and cannot accumulate reusable experience across interactions (Sumers et al., 2023; Tao et al., 2024). Agents therefore add external memory (Zhang et al., 2025c; Huang et al., 2026) built from prior interactions, including episodic memory banks (Zhong et al., 2024; Cai et al., 2025) and reusable skill libraries (Wang et al., 2023a; Xia et al., 2026a; Zhou et al., 2026). As shown in Figure 1, early *workflow-based memory* uses predefined rules for construction, retrieval, and update (Packer et al., 2023; Fu et al., 2024; Ouyang et al., 2025; Zhao et al., 2024; Wang et al., 2025c), while *learning-based memory* optimizes these operations with feedback from downstream task outcomes (Yan et al., 2025; Zhou et al., 2025; Zhang et al., 2026c; 2025b).

Both remain retrieval-based and inherit two failures. First, a top-k cutoff returns partial evidence when no single entry covers the task, while a larger k can bury useful evidence in a longer context (Liu et al., 2024). Second, similarity ranking can select damaging near-miss entries (Cuconasu et al., 2024) that share interface vocabulary with the current task but differ in the decisive detail. Both failures follow from entries that are fixed before the query arrives.

Adaptive retrieval methods decide whether, when, or how much to retrieve based on task complexity (Jeong et al., 2024), generation-time need (Jiang et al., 2023), or confidence and self-reflection (Wang et al., 2023b; Asai et al., 2023), and some further rework retrieved memory (Wu et al., 2026; Ji et al., 2026; Song et al., 2026; Hu et al., 2026). These approaches still construct guidance from stored entries. Recent methods instead generate guidance from parametric memory, including SEAM (Li et al., 2026), Advisor Models (Asawa et al., 2025), and ParamMem (Yao et al., 2026), but do so for every task, including those the agent can solve independently. What remains missing is a memory mechanism that generates context-specific guidance for a frozen, possibly closed agent and intervenes only when that guidance provides measurable utility to the agent.

We present Mem-π, which models memory as a separate generative policy π_{mem} with this capability. It decides *when* to generate based on the guidance's measured utility for that agent and learns *what* to generate for the current context. Encoding experience in π_{mem} fuses many experiences into one piece of guidance, bounds its footprint by model size, and lets a small model serve a larger or closed agent.

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.

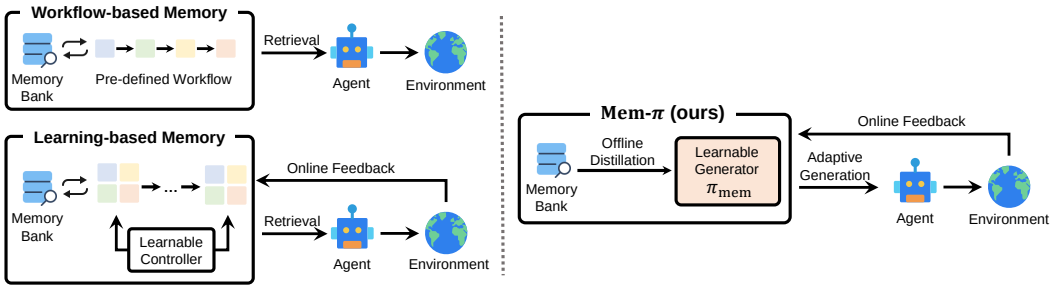


Figure 1: Comparison of (a) workflow-based memory systems, where memory operations are governed by predefined retrieval and update pipelines, (b) learning-based memory systems, where memory operations are jointly optimized with downstream agent outcomes, and (c) our Mem−π, which models memory as a generative policy π_{mem} separate from the downstream agent and internalizes reusable experience through offline experience distillation and online adaptation distillation.

We train Mem−π in two stages. *Experience distillation* first compresses an offline experience bank into the memory policy via supervised learning, internalizing reusable behaviors. *Adaptation distillation* then refines the policy with reinforcement learning on downstream utility, the difference between the frozen agent's success with the generated guidance and without it on the same task. Its decision-content decoupled objective, built on Group Relative Policy Optimization (GRPO) (Shao et al., 2024), compares generation and abstention on the same task through structured counterfactual rollouts and separates decision-level from content-level advantages.

We evaluate Mem−π on web navigation (WEBARENA (Zhou et al., 2023), WORKARENA (Drouin et al., 2024)), terminal tool use (LIFELONGAGENTBENCH (Zheng et al., 2025)), and text-based embodied environments (ALFWORLD (Shridhar et al., 2020b)). Adaptive memory generation outperforms retrieval-based memory baselines across all four benchmarks, with relative gains of 22% over the base agent on average and 59% on WEBARENA, where it improves over RAG by 37%.

2 DESIGN OF MEM−π

We model adaptive memory as a generative policy π_{mem} parameterized by θ and instantiated as a dedicated language or vision-language model Mem−π, separate from the downstream agent. Mem−π produces guidance that is injected into the agent's context at inference time. Let $\mathcal{E}$ denote an offline experience bank of context-guidance pairs (x, m), where $x = (q, o)$ consists of a task specification q and an environment observation $o \in \mathcal{O}$, and $m \in \mathcal{M}$ is textual guidance for the downstream agent. The observation o may include structured text and visual inputs such as screenshots. Figure 2 details the two training stages. First, *experience distillation* learns a mapping $\pi^{(1)}_{\text{mem}} : (q, o) \mapsto m$ from $\mathcal{E}$, converting explicit offline experiences into parametric knowledge. Second, *adaptation distillation* initializes $\pi^{(2)}_{\text{mem}}$ from $\pi^{(1)}_{\text{mem}}$ and optimizes the shared parameters θ through reinforcement learning from downstream agent outcomes. It introduces abstention and a decision-content decoupled objective to align both the generation decision and guidance content with task utility. Experience distillation follows context-supervised pretraining (Gao & Callan, 2022; W et al., 2023), where models reconstruct knowledge from context and internalize it into their parameters. Let m_t denote the t-th token of the target memory m, and let $m_{<t}$ denote its preceding tokens. We optimize $\pi^{(1)}_{\text{mem}}$ with the standard autoregressive supervised objective:

$$\mathcal{J}^{(1)}_{\text{mem}}(\theta) = \mathbb{E}_{(x,m)\sim\mathcal{E}}\left[\sum_{t=1}^{|m|} \log \pi^{(1)}_{\text{mem}}(m_t \mid x, m_{<t}; \theta)\right]. \tag{1}$$

2.1 ADAPTATION DISTILLATION

While experience distillation provides a strong initialization, the supervised policy cannot determine *when* memory generation is useful or potentially harmful, and its guidance is not directly optimized

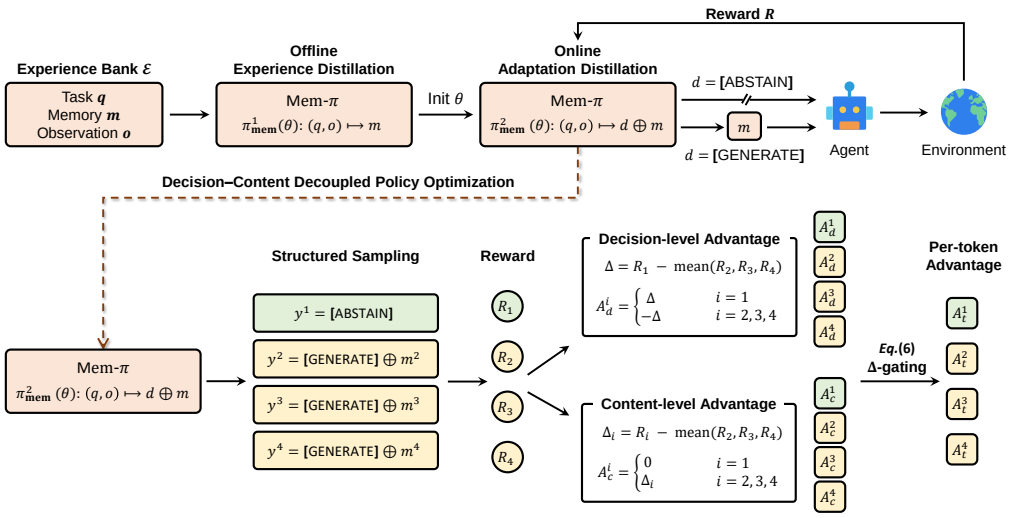


Figure 2: Overview of Mem-π. We train the generative memory policy π_{mem} in two stages. *Experience Distillation* distills reusable experience from an offline experience bank via supervised learning. *Adaptation Distillation* then optimizes π_{mem} against downstream agent outcomes using decision-content decoupled policy optimization, which pairs abstain and generate rollouts and decomposes the GRPO advantage into decision-level and content-level signals.

for the downstream agent. Adaptation distillation addresses these limitations by refining π_{mem} with RL using agent outcomes as the reward signal. We add two decision tokens, `[GENERATE]` and `[ABSTAIN]`, to the vocabulary and initialize their embeddings symmetrically so that both decisions have comparable initial probabilities. Specifically, we define $\pi^{(2)}_{\text{mem}} : (q, o) \mapsto y$, where $y = d \oplus m$, $d \in \{\texttt{[GENERATE]}, \texttt{[ABSTAIN]}\}$, $m \in \mathcal{M} \cup \{\varnothing\}$, and $\oplus$ denotes string concatenation. When $d = \texttt{[GENERATE]}$, the policy emits guidance $m \in \mathcal{M}$ and prepends it to the agent input to form $x \oplus m$. When $d = \texttt{[ABSTAIN]}$, we set $m = \varnothing$, and the agent operates on x.

We adopt GRPO (Shao et al., 2024), which estimates advantages from grouped samples without a value model. For each x, GRPO samples G outputs $\{y^1, \ldots, y^G\}$ and computes $\hat{A}^j = (r^j - \text{mean}(\mathbf{r}))/(\text{std}(\mathbf{r}) + \epsilon_{\text{std}})$, where $\mathbf{r} = (r^1, \ldots, r^G)$. The GRPO objective is:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_x \left[\frac{1}{G} \sum_{j=1}^{G} \frac{1}{|y^j|} \sum_{t=1}^{|y^j|} \Big(\min\Big(\rho_t^j \hat{A}^j,\ \text{clip}\big(\rho_t^j, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}\big) \hat{A}^j \Big) - \beta\, D^{(t)}_{\text{KL}} \Big) \right], \quad (2)$$

where $\rho_t^j = \pi^{(2)}_{\text{mem}}(y_t^j \mid x, y^j_{<t}; \theta) / \pi_{\text{old}}(y_t^j \mid x, y^j_{<t}; \theta_{\text{old}})$ is the importance ratio between the current policy and the old rollout policy, and $D^{(t)}_{\text{KL}}$ is the token-level KL divergence from π_{ref}. Here, π_{old} is a frozen snapshot of $\pi^{(2)}_{\text{mem}}$, and π_{ref} is the Stage 1 policy $\pi^{(1)}_{\text{mem}}$ before adaptation distillation.

Reward design. The reward $r = R(x, y)$ consists of a downstream task reward and, for generated memories, a length regularizer R_m. Given $y = d \oplus m$, we define:

$$R(x, y) = \begin{cases} \text{TaskReward}\big(\pi_{\text{agent}}(\cdot \mid x \oplus m)\big) + R_m(m), & \text{if } d = \texttt{[GENERATE]} \\ \text{TaskReward}\big(\pi_{\text{agent}}(\cdot \mid x)\big), & \text{if } d = \texttt{[ABSTAIN]}, \end{cases} \quad (3)$$

where π_{agent} is the frozen agent's action distribution, and $\text{TaskReward}(\cdot) \in \{0, 1\}$ indicates task success under the memory-augmented or original context. Following length-aware reward shaping in reasoning and agentic LLMs (Aggarwal & Welleck, 2025; Yu et al., 2025b; Liu et al., 2025c), we use $R_m(m) = -\lambda_{\text{len}} |m| / L_{\max}$ to discourage verbose or overly specific guidance, where $|m|$ is the number of memory tokens, $L_{\max}$ is the generation budget, and $\lambda_{\text{len}} > 0$ controls the penalty.

2.2 Decision-Content Decoupled Policy Optimization

Applying standard GRPO directly to $y = d \oplus m$ conflates two learning signals because d governs *whether* memory is generated, while m governs *what* guidance is produced. Since Stage 2 begins

from a supervised policy that favors generation, standard *i.i.d.* sampling may produce groups without abstain rollouts and prevent comparison between the decisions. Moreover, the length imbalance between d and m allows content-level updates to dominate the flat per-token objective. We therefore use structured counterfactual rollouts to decompose the GRPO advantage into decision- and content-level signals and route each signal to the corresponding token positions.

Structured counterfactual rollout. For each context x, we construct a structured rollout group of G rollouts, with one abstain branch and $G-1$ generate branches:

$$y^1 = \texttt{[ABSTAIN]} \oplus \varnothing, \qquad y^j = \texttt{[GENERATE]} \oplus m^j, \quad j = 2, \ldots, G. \tag{4}$$

Every group thus compares generation with abstention. We use one abstain rollout because it has no content to sample, and spend the remaining rollouts on diverse guidance.

Decision-content advantage decomposition. We decompose the learning signal into a cross-branch decision advantage and a within-branch content advantage:

$$V_{\text{abs}} = R(x, y^1), \qquad V_{\text{gen}} = \frac{1}{G-1} \sum_{j=2}^{G} R(x, y^j), \qquad \Delta = V_{\text{abs}} - V_{\text{gen}}. \tag{5}$$

Here, Δ measures the benefit of abstaining over generating for the current context. The *decision* advantage is $A_d^j = +\Delta$ for the abstain rollout ($j = 1$) and $A_d^j = -\Delta$ for generate rollouts ($j > 1$), so the better branch receives positive advantage. The *content* advantage ranks guidance within the generate branch. We set $A_c^j = 0$ for $j = 1$, and $A_c^j = \big(R(x, y^j) - V_{\text{gen}}\big) / \big(\text{std}(\mathbf{r}_{\text{gen}}) + \epsilon_{\text{std}}\big)$ for $j > 1$, where $\mathbf{r}_{\text{gen}} = \{R(x, y^j)\}_{j=2}^{G}$. This term is standard GRPO within the generate branch.

Token-level credit assignment. To route these signals to the appropriate token positions, let T_d denote the length of the decision prefix and define:

$$A_t^j = \begin{cases} A_d^j, & t \le T_d \\ \mathbb{1}[\Delta < 0]\, A_c^j, & t > T_d \end{cases} \tag{6}$$

Decision tokens receive A_d^j, while content tokens receive A_c^j only when generation improves over abstention, *i.e.*, $\Delta < 0$. This Δ-gating prevents content-level updates when generation is not beneficial. Substituting A_t^j into the GRPO objective yields the Stage 2 adaptation objective:

$$\mathcal{J}_{\text{mem}}^{(2)}(\theta) = \mathbb{E}_x \left[\frac{1}{G} \sum_{j=1}^{G} \frac{1}{|y^j|} \sum_{t=1}^{|y^j|} \Big(\min\Big(\rho_t^j A_t^j,\ \text{clip}\big(\rho_t^j, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}\big) A_t^j \Big) - \beta D_{\text{KL}}^{(t)} \Big) \right] \tag{7}$$

Compared with the standard GRPO objective in Eq. 2, the only objective-level change is replacing the scalar group-relative advantage $\hat{A}^j$ with the per-token advantage A_t^j. This preserves the GRPO framework while decision tokens learn *when* to generate through cross-branch comparison, and content tokens learn *what* to generate through within-branch ranking. Appendix C.2 provides sample bank entries and implementation details.

3 Experiments

Benchmarks. **WebArena** (Zhou et al., 2023) spans five domains (Shopping, CMS, GitLab, Reddit, Maps) and uses the 647/165 train/test split of WebAgent-R1 (Wei et al., 2025b) and WebRL (Qi et al., 2024). **WorkArena** (Drouin et al., 2024) has 33 templates across four ServiceNow (ServiceNow, 2023) categories, menus and dashboards (M&D), forms, list filtering and sorting (Filt), and service catalog and knowledge (SK), with 20 training and 10 disjoint evaluation seeds per template. **LifelongAgentBench (LAB)** (Zheng et al., 2025) uses the Database (DB) and Operating System (OS) subsets with the 7:3 split of MemRL (Zhang et al., 2026c), 150 test tasks each. **ALFWorld** (Shridhar et al., 2020b) uses the official split with 134 unseen test tasks. We use task success rate (SR) as the reward and construct the offline experience bank with JEF-Hinter (Nekoei et al., 2025), which distills raw interaction traces into compact guidance. Across these settings, agents handle long-horizon browser interaction, cross-goal enterprise workflows, terminal commands with state-based verification, and partially observable household tasks.

Table 1: Task success rate (SR %) across four agent benchmarks with `GPT-5.4-mini` as the base agent. Averages weight categories by test-task count, except the uniform WEBARENA site mean. Avg. is the mean over benchmarks. **Bold**: best per column. Underline: second best.

Method	WEBARENA						WORKARENA					ALFWORLD							LAB			Avg.
	SHOP	CMS	GL	RED	MAP	*Avg*	M&D	FORM	FILT	SK	*Avg*	P&P	EXAM	CLN	HEAT	COOL	P2P	*Avg*	DB	OS	*Avg*	
Base Agent	28.4	14.6	31.2	28.8	32.4	27.1	31.9	55.7	11.6	76.9	41.8	88.3	82.7	86.1	85.0	85.5	78.8	84.8	28.5	25.0	26.8	45.1
RAG	28.6	20.2	37.4	38.2	32.4	31.4	33.6	60.6	9.0	78.1	42.2	89.1	83.4	86.8	85.5	85.8	79.6	85.5	29.9	27.2	28.5	46.9
Mem0	29.8	22.4	38.6	36.4	32.2	31.9	33.9	61.8	9.5	79.1	43.0	89.7	84.1	87.3	85.8	87.2	80.5	86.2	31.9	28.1	30.0	47.8
Memory-R1	30.6	24.8	40.2	38.8	31.6	33.2	35.4	62.3	10.5	80.5	44.1	89.9	85.0	87.7	86.3	87.7	81.1	86.7	32.6	29.8	31.2	48.8
MemRL	31.2	26.4	41.8	40.0	30.8	34.0	36.5	63.6	10.6	80.5	44.5	90.8	85.5	88.7	86.5	87.1	81.2	87.1	33.0	30.7	31.9	49.4
`Mem-π` (Stage 1)	30.6	17.4	46.8	47.8	32.4	35.0	41.6	65.8	9.0	81.5	45.5	92.1	88.0	91.2	89.0	90.0	84.1	89.5	35.3	32.9	34.1	51.0
`Mem-π`	**34.6**	**42.8**	**50.2**	**52.6**	**35.4**	**43.1**	**45.1**	**70.6**	**13.6**	**85.3**	**49.7**	**94.2**	**90.2**	**92.3**	**91.5**	**91.6**	**86.7**	**91.4**	**38.4**	**35.0**	**36.7**	**55.2**

Baselines. Beyond base agents with no memory, we compare two memory paradigms. *(i) Workflow-based memory*: **RAG** (Lewis et al., 2020) retrieves the top-k JEF-Hinter (Nekoei et al., 2025) experiences using BM25 (Robertson & Zaragoza, 2009), with $k = 1$ as in JEF-Hinter, while **Mem0** (Chhikara et al., 2025) combines RAG with rule-based management. *(ii) Learning-based memory*: **Memory-R1** (Yan et al., 2025) trains a memory manager with outcome-driven RL, while **MemRL** (Zhang et al., 2026c) learns Q-values for utility-aware retrieval. No method reads raw trajectories. Every memory method receives the same pre-distilled, training-split hint corpus, with test-task hints excluded.

Agent and memory configuration. The memory model `Mem-π` and the downstream agent are two separate models with independent parameters, even when they share the same backbone architecture. We use the Memory-R1 (Yan et al., 2025) backbone, `Qwen2.5-7B-Instruct` (Yang et al., 2024), for the memory policy π_{mem}. Training-based methods use only training-split tasks and their corresponding JEF-Hinter hints. The same split isolation is applied to the RAG and Mem0 banks. All results are evaluated on held-out test tasks. Every method therefore reads the same training-split experience, while compute differs, since RAG and Mem0 train nothing and `Mem-π`, Memory-R1 and MemRL learn from environment rewards. Stage 2 rewards come from a `Qwen2.5-7B-Instruct` agent fine-tuned under the WebAgent-R1 (Wei et al., 2025b) setting. The downstream agent remains frozen throughout memory training, so optimization updates only π_{mem}. We initialize the `[ABSTAIN]` and `[GENERATE]` decision tokens to near-equal logits, which gives approximately 50% abstention at the start of Stage 2. We evaluate the frozen memory policy with the proprietary `GPT-5.4-mini` (OpenAI, 2026), so all main results test cross-agent transfer. The memory policy is invoked once per episode before the agent's first action. In the visual setting, the memory model receives the initial screenshot and visual grounding extracted by `gemini-2.5-flash` (Google, 2025), using `Qwen2.5-VL-7B-Instruct` (Bai et al., 2025) as the visual backbone. This visual variant adds 2.7 pp to overall WEBARENA SR, and all main results use the text-only variant.

Implementation and evaluation. Stage 1 uses AdamW with learning rate 2×10^{-5}, batch size 32, and 3 epochs on $8\times$ NVIDIA H100-80GB GPUs. Stage 2 uses G=4, with one forced abstain and three generate branches of at most $L_{\max}$=256 tokens sampled at temperature 1.0 and top-p 0.95, learning rate 1×10^{-6}, batches of 8 tasks, 200 steps, ϵ_{clip}=0.2, β=0.01, and λ_{len}=0.1. Results average three seeds scored by each benchmark's official verifier. Standard deviations are at most 1.2 pp, while `Mem-π` leads the second-best method by at least 1.9 pp on every benchmark. Appendix C provides benchmark, baseline, implementation and variance details.

3.1 MAIN RESULTS

`Mem-π` performs best among the compared methods on every benchmark and domain. As summarized in Table 1, `Mem-π` leads every WEBARENA domain, with the largest absolute gains in CMS (+28.2 pp) and Reddit (+23.8 pp), suggesting that memorized experience is especially useful for their structured navigation patterns. On WORKARENA, `Mem-π` improves the base agent from 41.8% to 49.7% on average, with a +14.9 pp gain on Form. On ALFWORLD, `Mem-π` achieves 91.4%, a +6.6 pp improvement over the already-strong `GPT-5.4-mini` baseline. **Experience distillation alone already matches or surpasses RL-based baselines.** Without RL training, `Mem-π` (Stage 1) achieves 35.0% on WEBARENA, against 33.2% for Memory-R1 and 34.0% for MemRL. This result shows that offline parametric knowledge provides a strong initialization.

The RL stage provides substantial additional gains on WebArena. Moving from Stage 1 to the full model yields +8.1 pp on WEBARENA overall, with the largest jumps on CMS (+25.4 pp), Reddit (+4.8 pp), and Shopping (+4.0 pp). ALFWORLD gains a more modest +1.9 pp, which we attribute to the limited headroom above the frontier agent's high baseline.

3.2 ABLATION STUDY

RQ1: Are both training stages necessary? We compare `Mem-`π against two single-stage variants. *(i) w/o Stage 1 init* skips the experience distillation (SFT phase) and applies online RL directly to `Qwen2.5-7B-Instruct`. *(ii) Unified single-stage* collapses both stages into one RL phase that jointly optimizes the downstream task reward R_{task}, the same length regularizer R_m used in `Mem-`π, and an additional BERTScore-based similarity reward R_{sim} (Zhang et al., 2019) computed between the generated memory and the corresponding reference guidance from the training bank, so that the single RL stage has both an imitation signal toward reference hints and a downstream task signal. As shown in Table 2, **both training stages are essential, with unified training suffering the largest drop.** Removing Stage 1 initialization degrades WEBARENA by 5.2 pp, suggesting that without a well-initialized memory distribution, online RL struggles to converge. Unified single-stage training incurs a larger drop (-6.8 pp on WEBARENA), indicating that jointly optimizing the imitation reward R_{sim} and R_{task} cannot match `Mem-`π's staged optimization. We attribute this to a mismatch between the two rewards: R_{sim} encourages imitation of reference memories, whereas R_{task} rewards memories that improve task success. Since useful memories for new tasks may differ from the references, optimizing both rewards in a single stage can produce conflicting gradients.

Table 2: Ablation results (SR %) on WEBARENA and ALFWORLD. Subscripts show drop from full model.

Variant	WA	ALF
`Mem-`π (Full)	**43.1**	**91.4**
Training Stages		
w/o Stage 1 init	$37.9_{-5.2}$	$86.9_{-4.5}$
Unified single-stage	$36.3_{-6.8}$	$85.7_{-5.7}$
RL Objective		
w/o structured rollout	$38.3_{-4.8}$	$87.1_{-4.3}$
w/o Δ-gating	$41.3_{-1.8}$	$89.6_{-1.8}$
w/o R_m	$41.9_{-1.2}$	$90.5_{-0.9}$

RQ2: Does Stage 2 decision-content policy optimization help? We design three variants targeting its individual components. *(i) w/o structured rollout* reverts to vanilla GRPO without paired abstain and generate branches, *(ii) w/o* Δ*-gating* replaces gated fusion in Eq. 6 with a naive sum of decision- and content-level advantages, and *(iii) w/o* R_m drops the length-aware memory-quality reward. Variant *(i)* shows that **structured counterfactual sampling has the largest effect among the tested Stage 2 components.** Removing structured rollout costs 4.8 pp on WEBARENA and 4.3 pp on ALFWORLD, the largest drops among RL-objective ablations. Without paired generate-abstain comparisons, most sampling groups lack a routing signal, limiting the policy's ability to learn *when* to generate memory. Variants *(ii)* and *(iii)* show that **Δ-gating and R_m contribute complementary signals.** Removing Δ-gating causes consistent drops on both WEBARENA (-1.8 pp) and ALFWORLD (-1.8 pp), suggesting that naive fusion dilutes the optimization signal by allowing content-level updates even when abstention is preferable. Removing R_m leads to mild degradation, suggesting that the length regularizer encourages concise memory and helps reduce noise in generated guidance. Vanilla GRPO even falls 2.4 pp below its Stage 1 initialization on ALFWORLD, because it gives the decision token the same sequence-level advantage as the guidance content, which mixes the two learning signals.

RQ3: Which factors beyond retrieval account for the gain? We test whether the gain comes from invoking a second 7B model or from abstention, a mechanism retrieval can also adopt. We compare an untrained zero-shot hinter with no bank access and RAG with learned abstention under the same Stage 2 objective, reward, and 200-step budget. The 39.8 always-use cell forces `[GENERATE]` on the deployed `Mem-`π at test time, with no separate training. **Supervision and outcome optimization account for most of the gain.** As shown in Table 3, the zero-shot hinter reaches 29.3%, 2.2 pp above the Base Agent. All four cells in panel (a) invoke the same 7B backbone once. The remaining 13.8 pp to full `Mem-`π comes from experience distillation and outcome optimization. **Context-adaptive generation contributes more than learned routing.** Learned abstention lifts RAG by 4.1 pp to 35.5%, still 7.6 pp below the full model. The 11.7 pp improvement over RAG decomposes as $4.1 + 7.6$ or $8.4 + 3.3$, with generation contributing more in either order.

Table 3: Decomposing the WEBARENA gain (SR %) over the Base Agent at 27.1. (a) Rows vary Stage 1 supervision, and columns add Stage 2 RL. (b) Rows vary the memory content and columns the routing rule, with the routing gain in the last column.

(a) Supervision × Stage 2 RL

	no Stage 2	*+ Stage 2*
No bank supervision	29.3	37.9
JEF-Hinter supervision	35.0	**43.1**
distillation gain	+5.7	+5.2

(b) Memory content × routing

	always use	*learned abstain*	*routing*
Retrieved hint	31.4	35.5	+4.1
Generated guidance	39.8	**43.1**	+3.3
generation gain	+8.4	+7.6	

3.3 IN-DEPTH ANALYSIS

RQ4: Does adaptive memory generalize to unseen domains and degraded banks? We hold out one WEBARENA domain at a time and perturb the experience bank. **The gain persists on a held-out domain.** Leave-one-domain-out evaluation yields 37.7% against 43.1% in-domain, 10.6 pp above the Base Agent and 6.3 pp above full-bank RAG. **Generative memory is more robust to bank degradation.** When 50% of the hints are rewired to other same-domain tasks before any method uses the bank, RAG falls to 26.8% and Mem0 to 27.5%, while Mem-π holds 39.5%, at least 9.0 pp above the strongest baseline in every setting. Abstention rises from 34% to 53% as the bank degrades.

RQ5: Is the generated guidance reliable? On the 109 WEBARENA tasks where Mem-π generates, a blinded judge scores each hint three times with a fixed rubric, and we aggregate the scores by majority vote. **Generated guidance is more often correct than retrieved guidance.** Mem-π is fully correct on 88.1% of tasks against 67.0% for RAG, with misleading rates of 8.3% and 19.3%. **Stage 2 makes guidance more correct.** Stage 2 raises the fully correct rate from 81.7% to 88.1%, and rubric quality predicts agent success better than BERTScore (Spearman ρ=0.43 vs. 0.11).

RQ6: How does adaptive abstention improve performance? We analyze the relationship between adaptive abstention behavior and task difficulty on the WEBARENA benchmark. Tasks are grouped into five equal-width bins according to the base agent's success rate, where a lower success rate indicates higher task difficulty. For each bin, we report Mem-π's average abstention rate and the corresponding success-rate improvement over the base agent. **Mem-π abstains on easy tasks, generates on hard tasks, and improves performance most where memory is needed.** As shown in Figure 3, abstention is strongly correlated with task difficulty. On the easiest tasks, with base-agent SR from 80% to 100%, Mem-π abstains in approximately 71% of cases, while on the hardest tasks, the abstention rate drops to around 13%. The SR-improvement bars show the complementary trend: improvement peaks at +9.7 pp on the hardest bin and drops to +1.3 pp on the easiest bin. Final abstention rates are 34% on WEBARENA, 28% on WORKARENA, 21% on ALFWORLD and 39% on LAB, and on WEBARENA the rate converges within about 100 steps from the symmetric initialization.

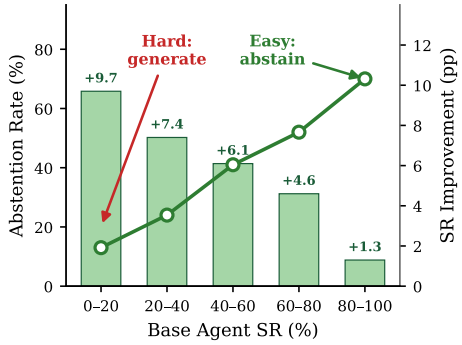


Figure 3: Abstention rate (line, left axis) and SR improvement over base agent (bars, right axis) across task-difficulty bins.

RQ7: Does adaptive memory generalize across agents? We compare RAG and Mem-π using the training-time agent (`Qwen2.5-7B-Instruct`) and an unseen frontier agent (`GPT-5.4-mini`), without retraining the memory model. **Mem-π transfers to the unseen `GPT-5.4-mini` without retraining.** As shown in Table 4, Mem-π gains 3.7 to 9.4 times as much as RAG. Most of its gain transfers on WEBARENA, where both base agents start near 27%, while the smaller ALFWORLD gain reflects the higher 84.8% start of `GPT-5.4-mini`. This sug-

Table 4: Cross-agent transfer (SR %). Subscripts show gain over base agent.

Method	Qwen2.5-7B		GPT-5.4-mini	
	WA	**ALF**	**WA**	**ALF**
Base Agent	27.9	72.9	27.1	84.8
+ RAG	$32.1_{+4.2}$	$75.0_{+2.1}$	$31.4_{+4.3}$	$85.5_{+0.7}$
+ Mem-π	$\mathbf{46.1}_{+18.2}$	$\mathbf{84.7}_{+11.8}$	$\mathbf{43.1}_{+16.0}$	$\mathbf{91.4}_{+6.6}$

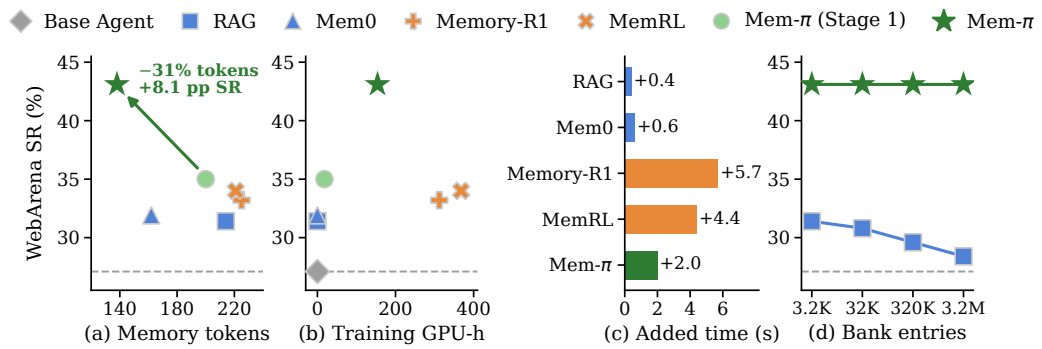


Figure 4: Efficiency on WEBARENA. (a) Performance vs. memory-token usage. (b) Performance vs. training GPU-hours. (c) Time added to a 121.4 s episode. (d) Performance as the experience bank grows. Dashed lines mark the Base Agent.

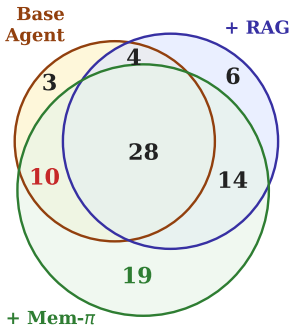
gests that memory learned from a weaker agent can encode task guidance at a sufficiently explicit level to remain useful for stronger unseen agents. **Routing accuracy remains high after cross-agent transfer.** With eight rollouts per task and agent, `GPT-5.4-mini` makes $6/165$ routing errors on WEBARENA and $4/134$ on ALFWORLD, compared with $8/165$ and $6/134$ for the training-time agent. Its accuracy on non-neutral tasks, where the decision changes the outcome, is 90.9% and 90.2%. After transfer, the share of generated guidance that helps on WEBARENA falls from 45.9% to 40.4%.

RQ8: Is adaptive generative memory token- and compute-efficient? We compare task success rate against the average number of memory tokens prepended to the agent input. The count covers retrieved content for external-memory baselines and generated content for generative-memory methods. For `Mem-π`, abstention contributes zero tokens. **`Mem-π` achieves the best performance-efficiency tradeoff.** As shown in Figure 4, `Mem-π` uses 138 memory tokens per task on average, 31% fewer than Stage 1 (200 tokens) and 39% fewer than Memory-R1 (225 tokens), while improving Stage 1 from 35.0% to 43.1%. By learning when to abstain, `Mem-π` reduces memory-token overhead and improves task success, yielding a better performance-efficiency tradeoff. **Parametric memory keeps overhead bounded as the bank grows.** `Mem-π` trains in 154 GPU-hours per seed, against 312 for Memory-R1 and 368 for MemRL. It adds 2.0 s to a 121.4 s episode, against 5.7 s and 4.4 s, while RAG trains nothing and is faster on the original bank. From 3.2K to 3.2M bank entries, RAG falls from 31.4% to 28.4%, its index grows from 22 MB to 19.4 GB and its p95 retrieval latency from 0.04 s to 0.92 s, while `Mem-π` stays at 43.1% and 1.8 s per call as it never reads the bank at inference.

RQ9: Why does adaptive memory outperform retrieval-based memory? As shown in Figure 5, a success-set analysis over 165 WEBARENA tasks finds 29 tasks that `Mem-π` solves and RAG does not, comprising the 19 and 10 shown in the figure, against 10 that RAG solves and `Mem-π` does not. **Generation adapts the guidance to the current query.** When a query specifies a count, identifier, or format that differs from the most similar memory bank entry, retrieval copies the source verbatim, while generation rewrites it from parametric knowledge to match the current input. Case 1 illustrates this: the retrieved source is a top-2 task while the query is top-3, so RAG's hint reads "read the first two rows", whereas `Mem-π` conditions on the "3" and produces "take the first three rows". **Abstention discards guidance that no longer applies.** When a retrieved hint encodes assumptions, such as a specific product family or identifier, that do not transfer to the current query, any non-empty hint inherits this bias and misleads the agent. Case 2 illustrates this: the retrieved source narrows the search to "game card case", but the query asks for the *best* storage option that fits 40 cards. `Mem-π` emits `[ABSTAIN]`, letting the base agent search broadly, while injecting the narrowed hint would have caused failure. The eight-region taxonomy has a worked case per region. Appendix C.13 gives the full analysis.

4 RELATED WORK

Retrieval-centric agent memory. Agent memory has progressed from static pipelines (Packer et al., 2023; Shinn et al., 2023; Wang et al., 2023a) to operations learned from downstream outcomes (Zhang



Case 1. *Top search terms (CMS).*
Task: "List the top 3 search terms in my store."
RAG: ✗ "…locate the 'Top Search Terms' table…read the *first two rows* and report each 'Search Term' with its 'Uses' count."
Mem−π: ✓ "…take the *first three rows* from the 'Search Term' column and output just those term names…"
Why: The retrieved hint comes from a top-2 source task and transports its "2" verbatim. Generation conditions on the explicit "3" in the query and rewrites the count.

Case 2. *Storage option search (Shopping).*
Task: "…find the best storage option to fit all 40 [Switch game] cards."
RAG: ✗ "…search…for 'game card case' or 'cartridge holder', open listings that show 40+ slots…"
Mem−π: ✓ [ABSTAIN]
Why: The query asks for the *best* storage option. The retrieved hint locks search to a narrow product family that excludes broader organizers. Abstention restores the base agent's open search.

Figure 5: Case analysis of Mem−π, RAG, and the base agent. Left: success-set Venn diagram across the three methods, where the colored numbers mark two Mem−π win patterns over RAG: 19 tasks solved only by Mem−π, and 10 solved by both the base agent and Mem−π but not RAG. Right: one sampled case for each pattern, comparing Mem−π and RAG under the same task query.

et al., 2024; Hu et al., 2025; Huang et al., 2026). One line distills trajectories into retrievable rules or strategies (Zhao et al., 2024; Wang et al., 2025c; Wu et al., 2025a; Yang et al., 2026b;a), including context-conditional guidelines (Fu et al., 2024) and lessons from successful and failed experiences (Ouyang et al., 2025). Another line trains memory controllers (Yu et al., 2026; 2025a; Zhang et al., 2026b;c; Zhan et al., 2025), co-evolves memory with the agent (Xu et al., 2025; Zhou et al., 2025; Xiao et al., 2026; Wang et al., 2026), or learns structured operations, architectures, and hierarchical skills (Yan et al., 2025; Zhang et al., 2025b; Wang et al., 2025b; Xia et al., 2026a). However, growing banks can degrade under repeated rewriting (Zhang et al., 2026a). These methods optimize how stored entries are accessed and managed, while Mem−π generates context-specific guidance from experience internalized in its parameters rather than replaying a fixed bank entry.

Generative memory. In parallel, generative memory encodes experience into parameters and produces information on demand (Qian et al., 2024; Liu et al., 2025b; Cheng et al., 2026; Ding et al., 2026; Jaiswal et al., 2026). This line spans context or knowledge compression into weights (Choi et al., 2022; Chevalier et al., 2023; Li et al., 2024; Padmanabhan et al., 2024; Wang et al., 2024), retriever imitation, multimodal compression, and reversible compression (Cao et al., 2025; Wei et al., 2025a; Wu et al., 2025b; Wang et al., 2025a). In agent settings, ParamMem (Yao et al., 2026) encodes reflection patterns, and MemGen (Zhang et al., 2025a) learns penalized invocation of latent memory. Other methods train separate guides with GRPO (Vishe et al., 2026; Cai et al., 2026a; Li et al., 2026; Asawa et al., 2025), expose retrieval as an action (Cai et al., 2026b), compare success with and without retrieved experience for copiloting (Feng et al., 2026) or distillation (Xie et al., 2026), or internalize experience into adapters, fast weights, and online planners (Guo et al., 2026; Xia et al., 2026b; Ren et al., 2026; Qiao et al., 2026). However, repeated internalization can collapse (Chen et al., 2026). Mem−π serves a frozen and possibly closed agent with a separate guidance generator whose abstention is trained on that agent's outcome without memory. Appendix B compares these methods along five axes: guidance form, downstream-agent access, intervention rule, use of a no-memory counterfactual, and outcome supervision. Together, these axes distinguish how memory is represented, when it intervenes, and whether its content is optimized for the receiving agent's outcomes.

5 Conclusion

We presented Mem−π, a generative memory policy that learns *when* and *what* guidance to provide a frozen downstream agent. Its two-stage training first internalizes a train-only experience bank, then optimizes generation and abstention for downstream utility with decision-content decoupled RL. Across four benchmarks, Mem−π raises mean SR from 45.1% to 55.2%, 5.8 pp above the strongest external baseline, and transfers to a stronger unseen agent without retraining. Ablations identify experience distillation and paired generate-abstain rollouts as key contributors, while case studies show that generation can correct query-specific details copied incorrectly by retrieval. The current design assumes a fixed bank, one pre-action memory call, and no source-level provenance. Future work should support continual updates, feedback-triggered regeneration, and traceable outputs.

AI USE STATEMENT

We used generative AI tools for literature search and writing assistance, including grammar suggestions and drafting of text. The authors carefully reviewed all AI-assisted content and made every edit to the final manuscript. The authors designed, ran, and checked all experiments, configurations, and measurements.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

Parth Asawa, Alan Zhu, Abigail O'Neill, Matei Zaharia, Alexandros G. Dimakis, and Joseph E. Gonzalez. How to train your advisor: Steering black-box LLMs with advisor models. *arXiv preprint arXiv:2510.02453*, 2025. doi: 10.48550/arXiv.2510.02453. ICML 2026.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Yishuo Cai, Xingyu Guo, Xuancheng Huang, Jinhua Du, Can Huang, Wenxuan Huang, Wenhan Ma, Yuyang Hu, Aohan Zeng, Jie Tang, and Xu Sun. From player to master: Enhancing test-time learning of LLM agents via reinforcement learning over memory. *arXiv preprint arXiv:2606.08656*, 2026a. doi: 10.48550/arXiv.2606.08656. ICML 2026.

Yuxuan Cai, Wei Li, Jie Zhou, Qin Chen, Xin Li, Bo Zhang, and Liang He. Ask only when needed: Proactive retrieval from memory and skills for experience-driven lifelong agents. *arXiv preprint arXiv:2604.20572*, 2026b. doi: 10.48550/arXiv.2604.20572.

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.

Jiaqi Cao, Jiarui Wang, Rubin Wei, Qipeng Guo, Kai Chen, Bowen Zhou, and Zhouhan Lin. Memory decoder: A pretrained, plug-and-play memory for large language models. *arXiv preprint arXiv:2508.09874*, 2025.

Jingwen Chen, Wenkai Yang, Shengda Fan, Wenbo Nie, Chenxing Sun, Shaodong Zheng, Yangen Hu, Lu Pan, Ke Zeng, and Yankai Lin. Rethinking continual experience internalization for self-evolving LLM agents. *arXiv preprint arXiv:2606.04703*, 2026. doi: 10.48550/arXiv.2606.04703.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Yukun Li, et al. Conditional memory via scalable lookup: A new axis of sparsity for large language models. *arXiv preprint arXiv:2601.07372*, 2026.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

De Chezelles, Thibault Le Sellier, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F Xu, Siva Reddy, Quentin Cappart, et al. The browsergym ecosystem for web agent research. *arXiv preprint arXiv:2412.05467*, 2024.

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.

Eunbi Choi, Yongrae Jo, Joel Jang, and Minjoon Seo. Prompt injection: Parameterization of fixed inputs. *arXiv preprint arXiv:2206.11349*, 2022.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for RAG systems. *arXiv preprint arXiv:2401.14887*, 2024. doi: 10.1145/3626772.3657834. SIGIR 2024.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Ning Ding, Fangcheng Liu, Kyungrae Kim, Linji Hao, Kyeng-Hun Lee, Hyeonmok Ko, and Yehui Tang. Meki: Memory-based expert knowledge injection for efficient llm scaling. *arXiv preprint arXiv:2602.03359*, 2026.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? In *International Conference on Machine Learning*, pp. 11642–11662. PMLR, 2024.

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

Tao Feng, Chongrui Ye, Tianyang Luo, Jingjun Xu, Xueqiang Xu, Haozhen Zhang, Zhigang Hua, Yan Xie, Shuang Yang, Ge Liu, and Jiaxuan You. ExpGraph: Model-agnostic experience learning with graph-structured memory for LLM agents. *arXiv preprint arXiv:2605.30712*, 2026. doi: 10.48550/arXiv.2605.30712.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. *Advances in Neural Information Processing Systems*, 37:119919–119948, 2024.

Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2843–2853, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.203. URL https://aclanthology.org/2022.acl-long.203/.

Google. Gemini 2.5: Updates to our family of thinking models. https://developers.googleblog.com/gemini-2-5-thinking-model-updates/, 2025. Accessed: 2026-09-26.

Yuchen Guo, Junli Gong, Weicheng Wang, Hongmin Cai, Yiu-ming Cheung, and Weifeng Su. PEAM: Parametric embodied agent memory through contrastive internalization of experience in Minecraft. *arXiv preprint arXiv:2605.27762*, 2026. doi: 10.48550/arXiv.2605.27762.

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*, 2025.

Zhanghao Hu, Qinglin Zhu, Runcong Zhao, Di Liang, Hanqi Yan, Yulan He, and Lin Gui. Beyond RAG for agent memory: Retrieval by decoupling and aggregation. *arXiv preprint arXiv:2602.02007*, 2026. doi: 10.48550/arXiv.2602.02007.

Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, Xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, et al. Rethinking memory mechanisms of foundation agents in the second half. *arXiv preprint arXiv:2602.06052*, 2026.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Ajay Jaiswal, Lauren Hannah, Han-Byul Kim, Duc Hoang, Arnav Kundu, Mehrdad Farajtabar, and Minsik Cho. Memoryllm: Plug-n-play interpretable feed-forward memory for transformers. *arXiv preprint arXiv:2602.00398*, 2026.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.389. URL https://aclanthology.org/2024.naacl-long.389/.

Shuo Ji, Yibo Li, and Bryan Hooi. Memory is reconstructed, not retrieved: Graph memory for LLM agents. *arXiv preprint arXiv:2606.06036*, 2026. doi: 10.48550/arXiv.2606.06036.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474, 2020.

Xuancheng Li, Haitao Li, Yujia Zhou, Yiqun Liu, and Qingyao Ai. Beyond experience retrieval: Learning to generate utility-optimized structured experience for frozen llms. *arXiv preprint arXiv:2602.02556*, 2026.

Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. Prompt compression for large language models: A survey. *arXiv preprint arXiv:2410.12388*, 2024.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*, 2025a.

Junming Liu, Yifei Sun, Weihua Cheng, Haodong Lei, Yirong Chen, Licheng Wen, Xuemeng Yang, Daocheng Fu, Pinlong Cai, Nianchen Deng, et al. Memverse: Multimodal memory for lifelong learning agents. *arXiv preprint arXiv:2512.03627*, 2025b.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, et al. Dler: Doing length penalty right-incentivizing more intelligence per token via reinforcement learning. *arXiv preprint arXiv:2510.15110*, 2025c.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Hadi Nekoei, Aman Jaiswal, Patrice Bechard, Oleh Shliazhko, Orlando Marquez Ayala, Mathieu Reymond, Massimo Caccia, Alexandre Drouin, Sarath Chandar, and Alexandre Lacoste. Just-in-time episodic feedback hinter: Leveraging offline knowledge to improve llm agents adaptation. *arXiv preprint arXiv:2510.04373*, 2025.

OpenAI. Introducing GPT-5.4 mini and nano. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/, 2026. Accessed: 2026-09-26.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

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, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Shankar Padmanabhan, Yasumasa Onoe, Michael Zhang, Greg Durrett, and Eunsol Choi. Propagating knowledge updates to lms through distillation. *Advances in Neural Information Processing Systems*, 36, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery. *arXiv preprint arXiv:2409.05591*, 2024.

Jingyang Qiao, Weicheng Meng, Yu Cheng, Zhihang Lin, Zhizhong Zhang, Xin Tan, Jingyu Gong, Kun Shao, and Yuan Xie. Memory intelligence agent. *arXiv preprint arXiv:2604.04503*, 2026. doi: 10.48550/arXiv.2604.04503.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3505–3506, 2020.

Tao Ren, Weiyao Luo, Hui Yang, Rongzhi Zhu, Xiang Huang, Yuchuan Wu, Bingxue Chou, Jieping Ye, Jiafeng Liang, Yongbin Li, and Yijie Peng. Scaling self-evolving agents via parametric memory. *arXiv preprint arXiv:2606.04536*, 2026. doi: 10.48550/arXiv.2606.04536.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

ServiceNow. Vancouver release notes. https://docs.servicenow.com/bundle/vancouver-release-notes/, 2023. Accessed: 2026-05-04.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10740–10749, 2020a.

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*, 2020b.

Yuanyi Song, Yukai Wang, Xinbei Ma, Zhihui Fu, Jianghao Lin, Weiwen Liu, Jun Wang, Huarong Deng, Yong Yu, and Weinan Zhang. Retrieval-driven memory reconsolidation for long-term LLM agents. *arXiv preprint arXiv:2609.16053*, 2026. doi: 10.48550/arXiv.2609.16053.

Theodore R Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387*, 2024.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Yash Vishe, Rohan Surana, Xunyi Jiang, Zihan Huang, Xintong Li, Nikki Lijing Kuang, Tong Yu, Ryan A. Rossi, Jingbo Shang, Julian McAuley, and Junda Wu. Skill-R1: Agent skill evolution via reinforcement learning. *arXiv preprint arXiv:2605.09359*, 2026. doi: 10.48550/arXiv.2605.09359.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning. `https://github.com/huggingface/trl`, 2020. Apache-2.0 licensed software.

Xing W, Guangyuan Ma, Wanhui Qian, Zijia Lin, and Songlin Hu. Query-as-context pre-training for dense passage retrieval. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1906–1916, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.118. URL `https://aclanthology.org/2023.emnlp-main.118/`.

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*, 2023a.

Xiaoqiang Wang, Suyuchen Wang, Yun Zhu, and Bang Liu. R^3Mem: Bridging memory retention and retrieval via reversible compression. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4541–4557, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.235. URL `https://aclanthology.org/2025.findings-acl.235/`.

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. Self-knowledge guided retrieval augmentation for large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10303–10315, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.691. URL `https://aclanthology.org/2023.findings-emnlp.691/`.

Yu Wang, Xinshuang Liu, Xiusi Chen, Sean O'Brien, Junda Wu, and Julian McAuley. Self-updatable large language models with parameter integration. *arXiv preprint arXiv:2410.00487*, 2024.

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

Zhenting Wang, Huancheng Chen, Jiayun Wang, and Wei Wei. Memex (rl): Scaling long-horizon llm agents via indexed experience memory. *arXiv preprint arXiv:2603.04257*, 2026.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *International Conference on Machine Learning*, pp. 63897–63911. PMLR, 2025c.

Rubin Wei, Jiaqi Cao, Jiarui Wang, Jushi Kai, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. Mlp memory: A retriever-pretrained memory for large language models. *arXiv preprint arXiv:2508.01832*, 2025a.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. WebAgent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7909–7928, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.401. URL https://aclanthology.org/2025.emnlp-main.401/.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025a.

Rong Wu, Daocheng Fu, Licheng Wen, Xuemeng Yang, Shu Zou, Jianbiao Mei, Yuxin Wang, Hairong Zhang, Yu Yang, Tao Hu, Cong Zhang, Botian Shi, and Pinlong Cai. MemHarness: Memory is reconstructed, not replayed. *arXiv preprint arXiv:2607.28272*, 2026. doi: 10.48550/arXiv.2607.28272.

Wenyi Wu, Zixuan Song, Kun Zhou, Yifei Shao, Zhiting Hu, and Biwei Huang. Towards general continuous memory for vision-language models. *arXiv preprint arXiv:2505.17670*, 2025b.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026a.

Siyu Xia, Chenheng Zhang, Yanting Wu, Haoxuan Li, Jiajun Chai, Xiaohan Wang, Guojun Yin, Wei Lin, Zhouchen Lin, Haifeng Zhang, and Jun Wang. UniMem: Complementary episodic-to-parametric memory for boundary-agnostic task streams. *arXiv preprint arXiv:2607.26017*, 2026b. doi: 10.48550/arXiv.2607.26017.

Han Xiao, Guozhi Wang, Hao Wang, Shilong Liu, Yuxiang Chai, Yue Pan, Yufeng Zhou, Xiaoxin Chen, Yafei Wen, and Hongsheng Li. Ui-mem: Self-evolving experience memory for online reinforcement learning in mobile gui agents. *arXiv preprint arXiv:2602.05832*, 2026.

Can Xie, Yuyi Zhou, Wen Yang, Ziyi Zhang, Siyao Song, Yingzhuo Deng, Shuo Ren, and Jiajun Zhang. EDGE: Experience-distillation for guided exploration in agentic reinforcement learning. *arXiv preprint arXiv:2608.21946*, 2026. EMNLP 2026 (Main Conference).

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.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Ke Yang, Zixi Chen, Xuan He, Jize Jiang, Michel Galley, Chenglong Wang, Jianfeng Gao, Jiawei Han, and ChengXiang Zhai. Plugmem: A task-agnostic plugin memory module for llm agents. *arXiv preprint arXiv:2603.03296*, 2026a.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, et al. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026b.

Tianjun Yao, Yongqiang Chen, Yujia Zheng, Pan Li, Zhiqiang Shen, and Kun Zhang. Parammem: Augmenting language agents with parametric reflective memory. *arXiv preprint arXiv:2602.23320*, 2026.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025a.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *arXiv preprint arXiv:2601.01885*, 2026.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F Wong, and Yu Cheng. Exgrpo: Learning to reason from experience. *arXiv preprint arXiv:2510.02245*, 2025.

Dylan Zhang, Yanshan Lin, Zhengkun Wu, Yihang Sun, Bingxuan Li, Dianqi Li, and Hao Peng. Useful memories become faulty when continuously updated by LLMs. *arXiv preprint arXiv:2605.12978*, 2026a. doi: 10.48550/arXiv.2605.12978.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025a.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*, 2025b.

Kehao Zhang, Shangtong Gui, Sheng Yang, Wei Chen, and Yang Feng. Learning to remember: End-to-end training of memory agents for long-context reasoning. *arXiv preprint arXiv:2602.18493*, 2026b.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026c.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2019.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*, 2024.

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

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (17):19632–19642, 2024.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12):3848–3860, 2023.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025.

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.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, et al. Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering. *arXiv preprint arXiv:2604.08224*, 2026.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025.

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

A LIMITATIONS

Mem-π generates guidance once per episode and does not regenerate it after failures or later observations within the episode. Its routing decision conditions on the task and observation but does not condition on the downstream agent, so after transfer to a stronger agent, part of the guidance budget produces no change in outcome, as analyzed in Appendix C.7. Correcting a stale memory requires editing the experience bank and re-distilling the memory policy. This process preserves the source of each bank entry but does not provide per-guidance attribution. The resulting grounding is outcome-level and therefore provides weaker source attribution than retrieving a stored entry.

Every evaluated method reads an experience bank built by the same procedure, JEF-Hinter. The robustness results cover only the tested perturbations, namely moderate sparsity and up to 50% same-domain rewiring, and do not support claims about systematic or adversarial corruption or banks constructed by other procedures, as detailed in Appendix C.5. The controls that isolate the sources of gain in Section 3.2 and the leave-one-domain-out evaluation in Appendix C.4 are limited to WEBARENA. The hint-level reliability analysis in Appendix C.6 uses an LLM judge without human annotation. Training and inference costs are measured on one hardware setup, using 8 H100 GPUs for training and one for inference.

B POSITIONING AGAINST RELATED WORK

Table 5 places Mem-π against the closest lines of work discussed in Section 4, one row per method and one column per property that separates them.

Table 5: Positioning of Mem-π against the closest lines of work. *Closed agent*: the method needs only text access to the downstream agent. *Learned intervention*: a trained decision on whether memory reaches the agent. *Vs. no-memory outcome*: that decision (or the method's reward) compares the same agent with and without memory on the same task. *Outcome-trained content*: the delivered guidance is optimized on downstream success. ✓: yes, ✗: no, ○: partially.

Method	Guidance at inference	Closed agent	Learned intervention	Vs. no-memory outcome	Outcome-trained content
Retrieval and memory managers					
RAG (Lewis et al., 2020)	stored entry	✓	✗	✗	✗
ReasoningBank (Ouyang et al., 2025)	stored strategy	✓	✗	✗	✗
Memory-R1 (Yan et al., 2025)	stored entry	✗	○	✗	✗
MemRL (Zhang et al., 2026c)	stored entry	✓	○	✗	✗
Learned hints and generative guides					
JEF-Hinter (Nekoei et al., 2025)	distilled hint, retrieved	✓	✗	✗	✗
SEAM (Li et al., 2026)	generated text	✓	✗	✗	✓
Advisor Models (Asawa et al., 2025)	generated text	✓	✗	✗	✓
MemGen (Zhang et al., 2025a)	latent tokens	✗	✓	✗	✓
Learning when experience reaches the agent					
ProactAgent (Cai et al., 2026b)	stored entry	✗	✓	✓	✗
MemHarness (Wu et al., 2026)	rewritten entry	✗	○	✗	✓
ExpGraph (Feng et al., 2026)	stored entries	✓	✗	○	✗
Mem-π	generated text	✓	✓	✓	✓

C EXPERIMENTAL DETAILS

C.1 BENCHMARK DETAILS

WEBARENA. WEBARENA (Zhou et al., 2023) is a realistic web-navigation benchmark consisting of 812 tasks across five fully functional web domains. *Shopping* (162 tasks) involves e-commerce actions such as product search, filtering by attribute, cart management, and order review. *CMS* (169 tasks) requires interacting with a content management system to create, edit, and publish posts and pages. *GitLab* (91 tasks) covers collaborative software development workflows including repository management, issue tracking, merge requests, and code review. *Reddit* (232 tasks) involves social forum interactions: posting, commenting, voting, searching threads, and moderating content.

Maps (158 tasks) requires navigating a map service to search for locations, compute directions, and extract geographic information. Each task requires an agent to interact with dynamic web interfaces through browser actions such as clicking, typing, searching, navigating across pages, and extracting information. These tasks are long-horizon and often require maintaining intermediate state, reasoning over web-page observations, and recovering from incorrect actions. Following WebAgent-R1 (Wei et al., 2025b), we adopt the standard train/test split and evaluate on held-out tasks.

WORKARENA. WORKARENA (Drouin et al., 2024) evaluates agents in enterprise workflow scenarios built on the ServiceNow cloud platform. The benchmark covers four representative workflow categories. *Dashboard & Menu Navigation* requires locating information across nested menus and dashboards. *Enterprise Forms* covers multi-field structured forms with domain-specific validation. *List Filter/Sort* requires complex filters, sorting criteria, and pagination over enterprise data lists. *Knowledge/Service Base* covers knowledge articles and service catalog entries. WORKARENA contains 33 atomic task templates, each of which can be instantiated with different random seeds. We follow the cross-goal generalization setting of BrowserGym (Chezelles et al., 2024): 20 seeds per template for training and 10 disjoint seeds for evaluation, testing whether agents can generalize learned interaction patterns to unseen goals within the same workflow family.

LIFELONGAGENTBENCH (LAB). LIFELONGAGENTBENCH (Zheng et al., 2025) is designed to evaluate lifelong learning and experience reuse in interactive terminal-based environments. It contains 1,396 total task instances across three environments: Database (DB), Operating System (OS), and Knowledge Graph (KG). Following prior work (Zhang et al., 2026c), we focus on the DB and OS subsets.

The *DB subset* (500 tasks) evaluates 22 SQL-related skills: basic `SELECT`, filtering (`WHERE`), grouping (`GROUP BY`), sorting (`ORDER BY`), aggregation (`COUNT`/`SUM`/`AVG`/`MAX`/`MIN`), nested subqueries, multi-table `JOIN`s, set operations (`UNION`/`INTERSECT`), and data manipulation (`INSERT`/`UPDATE`/`DELETE`). Execution results are verified automatically via SQL engine output.

The *OS subset* (500 tasks) evaluates 29 Bash-command skills: file and directory operations (`ls`, `cp`, `mv`, `find`), permission management (`chmod`, `chown`), user and group management (`useradd`, `groupmod`), text processing (`grep`, `sed`, `awk`, `wc`), compression (`tar`, `gzip`), process inspection (`ps`, `top`, `kill`), and system monitoring (`df`, `du`, `uptime`). Correctness is verified by checking final OS state.

For both subsets, we construct a 7:3 train/test split with random seed 42 (350 training and 150 test tasks per subset), evaluating whether `Mem-`π can distill reusable terminal-interaction experience from training tasks and transfer to unseen tasks within the same tool-use domain.

ALFWORLD. ALFWORLD (Shridhar et al., 2020b) is a text-based embodied household environment aligned with embodied simulator states (ALFRED (Shridhar et al., 2020a)). It converts household manipulation tasks into textual observations and actions while preserving long-horizon planning difficulty and partial observability. The benchmark includes six task types: *pick-and-place* (find an object and place it at a specified location), *examine-in-light* (find an object and examine it under a light source), *clean-and-place* (clean an object in a sink and place it at a target), *heat-and-place* (heat an object in a microwave and place it at a target), *cool-and-place* (cool an object in a refrigerator and place it at a target), and *pick-two-and-place* (find two instances of an object and place them together). These tasks require agents to locate objects, navigate between receptacles, manipulate object states, and place objects at target locations under partial observability.

The original ALFWORLD split contains 3,553 training tasks, 140 validation-seen tasks, and 134 validation-unseen tasks. Following prior work (Zhang et al., 2026c), we use the 3,553 training tasks as the experience pool for memory distillation and evaluate transfer on the 134 validation-unseen tasks. This setting tests whether the agent can reuse procedural experience from prior household interactions rather than memorizing specific trajectories.

C.2 IMPLEMENTATION DETAILS

Implementation stack. We implement `Mem-`π on top of PyTorch (Paszke et al., 2019) and the Hugging Face `transformers` library (Wolf et al., 2020). RL training is built on TRL (von Werra et al., 2020), with rollout generation served by vLLM (Kwon et al., 2023). `Mem-`π is initialized from `Qwen2.5-7B-Instruct` (Yang et al., 2024) for the default text-only setting used in all

main-table results. The multimodal variant uses `Qwen2.5-VL-7B-Instruct` (Bai et al., 2025) and is reserved for the visual-input ablation in Appendix C.9. The training-time agent is a separately fine-tuned `Qwen2.5-7B-Instruct` kept frozen throughout memory training. Both models use the standard chat template of their base. We add `[ABSTAIN]` and `[GENERATE]` to the vocabulary as special tokens. Each is initialized with the mean embedding of semantically related existing tokens, using *skip/bypass* for `[ABSTAIN]` and *recall/retrieve* for `[GENERATE]`. We then average the two embeddings to obtain a shared initialization, giving the decision tokens near-equal logits and approximately 50% abstention probability at the start of Stage 2.

Experience bank. We sample 5 hints per task with JEF-Hinter (Nekoei et al., 2025) from π_{agent} trajectories. Each hint is a short procedural summary of the verified flow for one task. To prevent test-set leakage, only hints derived from training tasks (per the train/test splits in Section C.1) are used as supervised targets in Stage 1 and as references for the BERTScore-based similarity reward R_{sim} used in the unified-stage ablation variant in Section 3.2. Hints derived from test tasks are excluded from training. Figure 7 shows one sample entry per benchmark to illustrate what the bank contains: task query, semantic key, hint text, and (for the visual benchmarks) the initial screenshot and grounding text used by the VL variant.

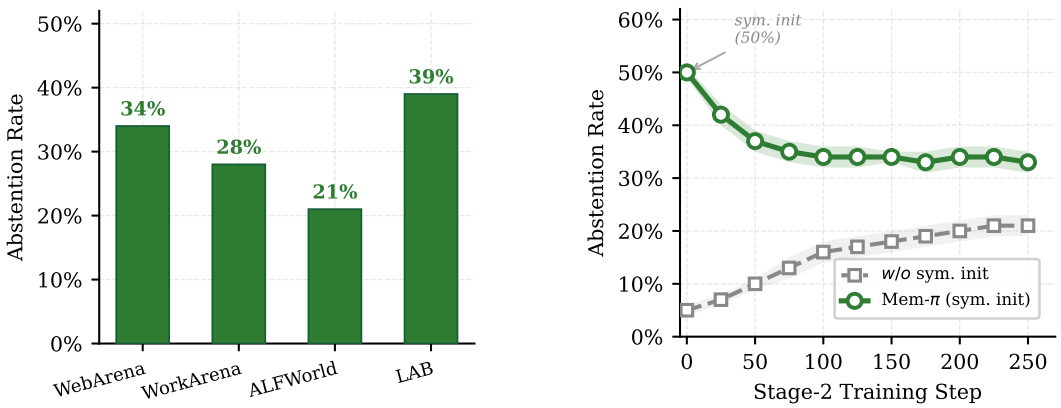
Stage 1: experience distillation. `Mem-`π is trained to imitate the hints in $\mathcal{E}$ via the autoregressive supervised objective in Eq. 1. We use AdamW (Loshchilov & Hutter, 2018) with β_1=0.9, β_2=0.999, learning rate 2×10^{-5}, cosine schedule with 5% warmup, batch size 32, weight decay 0.01, gradient clipping at 1.0, max sequence length 2,048 tokens, and 3 epochs. Stage 1 is parallelized with PyTorch FSDP (Zhao et al., 2023) across 8× NVIDIA H100-80GB GPUs.

Stage 2: adaptation distillation. Stage 2 fine-tunes `Mem-`π with the decision-content decoupled GRPO objective in Eq. 7, implemented as a custom `GRPOTrainer` subclass on TRL. For each task we sample a structured rollout (Eq. 4) of G=4 branches: one forced `[ABSTAIN]` (no generation) and three `[GENERATE]` branches each producing a memory of up to $L_{\max}$=256 tokens at sampling temperature 1.0 and `top_p` 0.95. Optimization uses AdamW with learning rate 1×10^{-6}, β_1=0.9, β_2=0.999, weight decay 0, batch size 8 tasks per step, and 200 optimization steps. The clip ratio is ϵ_{clip}=0.2, the KL coefficient is β=0.01, and the advantage normalization constant is $\epsilon_{\text{std}}=10^{-6}$ (Eq. 2). The length regularizer R_m uses λ_{len}=0.1 with generation budget $L_{\max}$=256 tokens (Eq. 3). The unified-stage ablation variant (Section 3.2, RQ1) additionally uses a BERTScore-based similarity reward R_{sim} (Zhang et al., 2019), computed as the F1 of layer 12 `bert-base-uncased` between the generated memory and the Stage 1 reference hint. Stage 2 uses DeepSpeed ZeRO-2 (Rasley et al., 2020) with vLLM colocated for inference on the same 8× H100-80GB GPUs. The policy model and the rollout vLLM instance share GPU memory via vLLM's sleep mode.

Baselines. RAG (Lewis et al., 2020) and Mem0 (Chhikara et al., 2025) retrieve the top-k=1 most similar hints from the same $\mathcal{E}$ used by `Mem-`π. Mem0 additionally applies its rule-based update pipeline. Memory-R1 (Yan et al., 2025) and MemRL (Zhang et al., 2026c) use their *official architectures and training procedures under the shared-bank protocol* of Appendix C.3. We keep each method's published architecture, action space, hyperparameters and optimization procedure, and adapt only the memory source so that every method reads the same corpus.

Evaluation protocol. For WEBARENA (Zhou et al., 2023) and WORKARENA (Drouin et al., 2024), we use the official benchmark verifiers from BrowserGym. For LAB (Zheng et al., 2025), correctness is verified by SQL execution (DB) and OS state checks through the benchmark's built-in verifiers. For ALFWORLD (Shridhar et al., 2020b), success is determined by the environment's terminal-condition checker. Reported numbers are means over three independent seeds. Standard deviations are in Table 14.

Length regularizer scope and saturation. The length regularizer $R_m(m)=-\lambda_{\text{len}}|m|/L_{\max}$ in Eq. 3 is applied only to `[GENERATE]` branches, where $m \neq \varnothing$. For `[ABSTAIN]` branches, we set $m=\varnothing$ and the reward reduces to $\text{TaskReward}(\pi_{\text{agent}}(\cdot \mid x))$, as in the second case of Eq. 3. Generation is capped by the decoding budget $L_{\max}$=256 via vLLM's `max_new_tokens`, so $|m| \in [1, L_{\max}]$ for every `[GENERATE]` branch and $R_m \in [-\lambda_{\text{len}}, 0]$. With λ_{len}=0.1, the penalty saturates at -0.1 and is an order of magnitude smaller than the binary task reward $\text{TaskReward} \in \{0, 1\}$. We strip prompt-leakage tokens, including chat-template scaffolding and instruction echoes, before counting $|m|$, so the penalty reflects only substantive memory content. By Eq. 3 and Eq. 4, R_m enters V_{gen} and therefore appears in the decision advantage A_d as well as the content advantage A_c. The



(a) Abstention rate per benchmark.

(b) Training dynamics on WEBARENA.

Figure 6: **Abstention statistics.** Left: final per-benchmark abstention rates after Stage 2 training. Right: abstention rate vs. training step, comparing symmetric vs. standard embedding initialization.

resulting shift in A_d has magnitude equal to the average per-task length penalty, at most $\lambda_{\text{len}}=0.1$ and approximately 0.05 at observed memory lengths. This is an order of magnitude below the per-task $\text{TaskReward} \in \{0, 1\}$ gap whenever generation flips downstream success, so the routing gradient on A_d remains dominated by task outcome rather than length cost.

C.3 BASELINE INPUTS

None of the evaluated memory methods consumes raw interaction trajectories. Every method receives the same pre-distilled JEF-Hinter hints from the training split, and test-task hints are excluded throughout. Table 6 states, per baseline, what is read and how it is used.

Table 6: Offline memory input and mechanism per method under the shared-bank protocol. *Outcome-trained* marks methods that additionally consume environment-verified rollout rewards, which RAG and Mem0 do not.

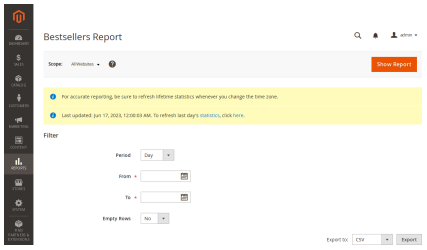
Method	Offline memory input	How it uses the input	Outcome-trained
RAG	Training-split hints	BM25 top-1 retrieval, injected verbatim	No
Mem0	Same hints	Retrieval with rule-based management	No
Memory-R1	Same hints as candidate items	Learned ADD/UPDATE/DELETE/NOOP	Yes
MemRL	Same hints as episodic items	Learned Q-values for utility-aware selection	Yes
Mem-π (Stage 1)	Same hints as SFT targets	Context-conditioned autoregressive generation	No
Mem-π	Same hints for Stage 1 init	Generation and abstention refined by outcomes	Yes

Scope of the protocol. This design holds the memory *corpus* fixed and compares how methods use and adapt it. It departs from running every baseline exactly as published in one case. Memory-R1 is designed to *construct* memory from raw trajectories. Supplying it pre-distilled hints exercises its memory-management controller but never its distillation operator. We chose information parity over operator fidelity because the alternative changes two variables at once, the information and the operator set, which would make any resulting difference unattributable. A raw-trajectory arm for Memory-R1 is therefore a separate experiment.

C.4 LEAVE-ONE-DOMAIN-OUT GENERALIZATION

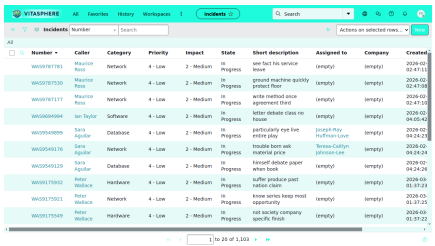
In each of five runs we remove one WEBARENA website from the JEF-Hinter bank *and* from both Mem-π training stages, then evaluate only on that website's test tasks, so the model has seen neither a task nor a hint from the target site. RAG retains the complete bank, including in-domain hints, which makes the comparison conservative. Following the per-domain columns of Table 1, the macro mean is unweighted.

WEBARENA · *shopping_admin (CMS)*



Task query. "What is the top-1 best-selling product in 2022."
Memory (hint). "Use the reports with date filters, not the dashboard widget: go to *Reports* > *Products* > *Bestsellers*, set *Period* to *Year*, enter *From* `01/01/2022` and *To* `12/31/2022` (MM/DD/YYYY), confirm scope if needed, click *Show Report*, then sort by *Order Quantity* to get the top-1."
Grounding (excerpt). *Page type*: Magento admin Bestsellers Report. *Key elements*: sidebar with *Reports* selected; main area has a *Period* dropdown (default "Day"), *From*/*To* date inputs, a *Show Report* button, and informational banners about lifetime-statistics refresh.

WORKARENA · *filter-incident-list (List)*



Task query. "Create a filter for the list to extract all entries where *Assigned to* is *ITIL User*, OR *Configuration item* is empty, OR *Short description* is *My desk phone does not work*."
Memory (hint). "Open the filter builder via the *Show/hide filter* button (not the *All* breadcrumb), set the group combobox to *Any of these conditions must be met* or use *Add OR Condition* on a row, configure each condition (field, operator like `is`/`is empty`, value), then click *Run filter* to apply."
Grounding (excerpt). *Page type*: ServiceNow Incident list. *Key elements*: top nav (*All*/*Favorites*/*History*/*Workspaces*/*Incidents*); sortable column headers (*Number*, *Caller*, *Category*, *Priority*, *Impact*, *State*, *Short description*, *Assigned to*, *Company*, *Created*); pagination "1 to 20 of 4,071"; filter / hamburger icons.

ALFWORLD · *clean-and-place*

Task query. "Place a rinsed plate in the fridge."
Memory (hint). "1. Look around to find the dirty item you want to clean. 2. Go to a suitable cleaning location (sink or basin). 3. Clean the item thoroughly using the cleaning location. 4. Go to the appropriate storage location (fridge or cabinet). 5. Ensure the storage location is open before placing the cleaned item inside. *Common pitfall*: forgetting to open the storage location before placing the item leads to failure."

LAB · *db: column_alias / os: awk*

Task query (LAB-DB). "Delete patient records where the number of refills left is below the average refills left across all records and the prescription date is before the latest prescription date for the medication `Aspirin`."
Memory (hint). "1. Identify the main condition for your `DELETE` operation and determine whether it requires comparison with aggregate values (*e.g.*, `AVG` or `MAX`). 2. Use subqueries to compute these aggregates and reference them in the `WHERE` clause. 3. Combine multiple conditions with `AND`; ensure the subquery on `MAX(prescription_date)` is restricted by medication."

Task query (LAB-OS). "Parse `/var/log/auth.log` to count failed SSH login attempts per user, save the result to `/report.txt`, and set permissions to read-only for the owner."
Memory (hint). "1. Verify the input file exists with `test -f` before piping into `grep`. 2. Use `grep 'Failed password'` to extract failure lines, then `awk {print $NF}` to take the username column. 3. Pipe through `sort | uniq -c` to count per user and redirect to `/report.txt`. 4. Apply `chmod 400 /report.txt` for owner-read-only."

Figure 7: Sample experience entries drawn from the offline bank $\mathcal{E}$ used to train `Mem-`π, one per benchmark. Each entry contains a task query (`source_trace_goals` in JEF-Hinter (Nekoei et al., 2025)) and the guidance (JEF-Hinter hint) text. For WEBARENA and WORKARENA, the bank additionally stores the initial screenshot and a structured grounding description used by the VL memory variant in Appendix C.9. ALFWORLD and LAB are text-only environments and have no visual channel.

Table 7: Leave-one-domain-out on WEBARENA (SR %). Mem−π (leave-one-out) never sees the held-out website, while RAG keeps full in-domain bank access.

Held-out domain	Base Agent	RAG (full bank)	Mem−π (in-domain)	Mem−π (leave-one-out)
Shopping	28.4	28.6	34.6	**31.9**
CMS	14.6	20.2	42.8	**32.4**
GitLab	31.2	37.4	50.2	**44.1**
Reddit	28.8	38.2	52.6	**46.3**
Maps	32.4	32.4	35.4	**33.8**
Macro mean	27.1	31.4	43.1	**37.7**

Leave-one-out Mem−π beats both the Base Agent and full-bank RAG on every fold, so the result is not carried by one favorable domain. Its gain over the Base Agent ranges from $+1.4$ pp on Maps, where memory helps least in-domain, to $+17.8$ pp on CMS. Its 37.7% macro mean also exceeds every full-domain baseline in Table 1, including Mem0 at 31.9%, Memory-R1 at 33.2%, MemRL at 34.0%, and the always-generating Stage 1 model at 35.0%. Those methods retain target-domain experience.

C.5 EXPERIENCE-BANK QUALITY STRESS TEST

The clean bank holds five verified hints for each of 647 training tasks, 3,235 entries in total. Each perturbed bank is built *before* training or retrieval indexing and shared identically across RAG, Mem0, Memory-R1, MemRL, Mem−π Stage 1 and full Mem−π. Train/test isolation and training schedules are unchanged, and each row averages three independently sampled perturbations. The Base Agent uses no bank and is constant at 27.1%.

Corruption is defined mechanically so that it is reproducible and involves no similarity model or human judgment: a rewired entry keeps its original task context as the retrieval key and receives a hint sampled uniformly from a different training task in the *same* WEBARENA domain, so site vocabulary and action space stay correct while the procedure becomes wrong. The cross-domain control samples the replacement from a different website instead.

Table 8: WEBARENA SR (%) under matched bank perturbations. *Margin* is Mem−π minus the strongest external baseline in that row.

Experience bank	RAG	Mem0	Memory-R1	MemRL	Mem−π S1	Mem−π	Abstain	Margin
Clean bank	31.4	31.9	33.2	34.0	35.0	**43.1**	34%	+9.1
50% task coverage	30.1	30.7	31.8	32.6	33.4	**41.8**	39%	+9.2
25% task coverage	28.8	29.3	30.4	31.2	31.5	**40.2**	45%	+9.0
10% task coverage	27.5	28.0	29.0	29.7	29.4	**38.8**	51%	+9.1
10% rewired (same dom.)	30.4	30.9	32.2	33.0	34.1	**42.4**	37%	+9.4
25% rewired (same dom.)	28.9	29.4	30.6	31.4	32.1	**40.9**	44%	+9.5
50% rewired (same dom.)	26.8	27.5	28.4	29.1	29.6	**39.5**	53%	+10.4
25% rewired (cross-dom.)	27.9	28.4	29.6	30.3	32.6	**41.5**	46%	+11.2

Stage 1 always generates and degrades with bank quality, whereas Stage 2 can fall back on abstention and on a reward that does not come from the bank. Full Mem−π also stays above the 37.9% reached without any Stage 1 supervision (Table 2) in every perturbed condition. This ablation is a separately trained model rather than a formal lower bound on the perturbed runs. The cross-domain control produces the same ordering with a wider margin, which makes same-domain rewiring the conservative choice rather than one selected to favor generation.

C.6 HINT-LEVEL RELIABILITY PROTOCOL

Stage 1 is trained to reconstruct JEF-Hinter guidance for *training* tasks, so scoring those same pairs would mostly measure recovery of Stage 1 supervision. We therefore evaluate frozen policies on the WEBARENA test split, where no test-task hints were used in training and several valid procedures may exist. `Mem-`π generates on 109 of the 165 test tasks and abstains on 56. The 109 generated cases are the evaluation set, and outputs from RAG, Memory-R1 and `Mem-`π Stage 1 are collected on the same tasks.

A blinded `GPT-5.4-mini` judge receives the task instruction, initial observation, anonymized hint, and one environment-verified successful trajectory. The trajectory is grounding evidence, not the unique valid solution. The five criteria are *factual and procedural correctness* (are the entities, interface elements, constraints and actions valid), *task consistency* (does the hint address this task rather than a related one), *actionability* (would following it help the agent progress), *unsupported content* (does it assert anything not supported by the task, observation or trajectory), and *misleadingness* (could following it produce an incorrect action or objective). A hint is *fully correct* only if it is factually valid, task-consistent, actionable, and free of both unsupported and misleading content. Method identities are removed, output order is randomized, and each hint is judged three times under majority vote.

For reference alignment we build three evaluation-only reference hints per task from separately collected, environment-verified successful test trajectories. These are constructed only after all policies and outputs are frozen and are never used for training, model selection or inference. Because several procedures may be valid we report the maximum BERTScore (Zhang et al., 2019) over the three references. Rubric quality tracks agent success more closely than reference similarity, with Spearman correlations of ρ=0.43 for rubric quality and ρ=0.11 for the maximum BERTScore. Stage 1 attains a slightly higher BERTScore than the full model despite lower rubric quality and lower success (Table 9).

Table 9: Hint-level quality on the 109 WEBARENA test tasks where `Mem-`π generates, judged blind against five criteria. Reference alignment is the maximum BERTScore over three evaluation-only references built after all outputs were frozen. Agent success is on the same 109-task subset.

Method	Fully correct	Unsupported	Misleading	Actionable	Max BERTScore	Agent SR
Base Agent	—	—	—	—	—	13.8
RAG	67.0	20.2	19.3	75.2	0.806	25.7
Memory-R1	75.2	13.8	12.8	82.6	0.818	29.4
`Mem-`π (Stage 1)	81.7	10.1	11.0	87.2	**0.838**	31.2
`Mem-`π	**88.1**	**6.4**	**8.3**	**92.7**	0.831	**37.6**

C.7 CROSS-AGENT ROUTING CALIBRATION

We freeze `Mem-`π and evaluate every task under both decisions. The generation branch forces `[GENERATE]` using the same mechanism as the Stage 2 structured rollouts. The abstention branch emits nothing and is identical to the Base Agent. One hint is decoded per task and reused across both agents and all generation-branch rollouts, so differences reflect the agents' need for guidance rather than different memory outputs.

For task i and agent a, we run K=8 independent downstream rollouts per branch and let $\hat{P}^a_{i,\text{gen}}$ and $\hat{P}^a_{i,\text{abs}}$ be the success fractions, with $\hat{\Delta}^a_i = \hat{P}^a_{i,\text{gen}} - \hat{P}^a_{i,\text{abs}}$. Generation is *helpful* when $\hat{\Delta}^a_i > 0.25$, *harmful* when $\hat{\Delta}^a_i < -0.25$, and neutral otherwise. Since success moves in increments of $1/8$, this criterion requires the branches to differ by at least three rollouts. A routing error is generating on a harmful task or abstaining on a helpful one, and the correct-choice rate is computed over non-neutral tasks only. Because π_{mem} does not condition on agent identity, the partition is fixed at 109 generated and 56 abstained on WEBARENA, and 106 generated and 28 abstained on ALFWORLD. Only the measured benefit of each decision changes across agents.

We additionally compare four routing strategies on a second, disjoint set of eight rollouts per branch, so that selection and scoring do not share data. *Per-task selected branch* chooses using the first eight-rollout comparison and is scored on the fresh rollouts. Neutral tasks keep `Mem-`π's learned

Table 10: Counterfactual routing quality per downstream agent. Accuracy on non-neutral tasks does not fall after transfer to the unseen stronger agent.

Benchmark and agent	Helpful *among gen.*	Harmful *among gen.*	Help missed *among abstained*	Routing errors */ all tasks*	Correct choice *non-neutral*
WEBARENA, `Qwen2.5-7B`	45.9%	4.6%	5.4%	8/165	89.5%
WEBARENA, `GPT-5.4-mini`	40.4%	3.7%	3.6%	**6/165**	**90.9%**
ALFWORLD, `Qwen2.5-7B`	34.0%	3.8%	7.1%	6/134	88.2%
ALFWORLD, `GPT-5.4-mini`	27.4%	2.8%	3.6%	**4/134**	**90.2%**

decision. This held-out evaluation estimates the performance of per-task routing with fixed guidance. Its gap from learned routing indicates the improvement available to the tested selector.

Table 11: Routing strategies (SR %). Learned routing beats always-generating on every agent and benchmark, and sits within 1.3 pp of the per-task selected branch.

Benchmark and agent	Always abstain	Always generate	**Mem-π routing**	Per-task selected	Gap
WEBARENA, `Qwen2.5-7B`	27.9	42.9	**46.1**	47.4	1.3
WEBARENA, `GPT-5.4-mini`	27.1	39.8	**43.1**	44.2	1.1
ALFWORLD, `Qwen2.5-7B`	72.9	82.9	**84.7**	85.5	0.8
ALFWORLD, `GPT-5.4-mini`	84.8	90.0	**91.4**	92.2	0.8

For `GPT-5.4-mini`, the helpful share among generations falls to 40.4% on WEBARENA, so a majority of the memory budget is spent on generations that do not change the outcome. This is a property of transferring a fixed, agent-independent policy to a stronger agent rather than of generative memory as such, and the budget stays bounded at 138 tokens per task and independent of bank size. The result motivates conditioning the routing decision on the downstream agent in future work.

C.8 TRAINING COMPUTE AND DEPLOYMENT COST

We exclude the cost of constructing the JEF-Hinter bank, which every method shares. All learned methods train in BF16 on 8×H100-80GB, and GPU-hours are wall-clock training time multiplied by eight. At inference, learned memory models run on one H100-80GB at batch size 1, while RAG and Mem0 run on 16 AMD EPYC cores with 128 GB RAM and a warmed in-memory index. *Memory preparation* is the time from receiving the task and initial accessibility-tree (axtree) observation to returning guidance, including tokenization, prefill, retrieval or controller execution, and decoding. *Full episode* is wall-clock time from environment reset to termination. Because memory changes the trajectory and episode length, it is measured independently rather than added to the Base Agent time.

Table 12: Training compute and WEBARENA deployment cost. `Mem-`π trains for roughly half the GPU-hours of the outcome-trained baselines and prepares memory faster, while adding 2.0 s to a 121.4 s episode.

Method	Training time / seed	GPU-hours / seed	Memory preparation	Full episode
Base Agent	—	0	—	121.4 s
RAG	—	0	0.03 s	121.8 s
Mem0	—	0	0.09 s	122.0 s
Memory-R1	39.0 h	312	4.8 s	127.1 s
MemRL	46.0 h	368	3.6 s	125.8 s
`Mem-`π Stage 1	2.3 h	18	2.7 s	124.0 s
`Mem-`π Stage 2	17.0 h	136	—	—
`Mem-`π total	**19.3 h**	**154**	**1.8 s**	**123.4 s**

The 1.8 s memory call is roughly 0.7 s of axtree tokenization and prefill and 1.1 s of decision and guidance decoding. This matters because Figure 4 counts only tokens inserted into the downstream agent and therefore omits the memory model's own prefill over a long observation. The two

measurements are complementary. We make no claim of compute parity with RAG and Mem0. They require no training and add 0.03 to 0.09 s per task, against a one-time 154 GPU-hours and 1.8 s per task for Mem−π, which gains 11.7 and 11.2 pp over them on WEBARENA. That one-time cost amortizes across downstream agents because the same frozen policy transfers to `GPT-5.4-mini` without retraining.

Bank-size scaling. Starting from the clean 3,235-entry bank, we add same-domain distractor hints derived exclusively from training-split tasks, preserving all original entries and introducing no test-task hints. RAG rebuilds and searches the complete BM25 index. Mem−π stays frozen and performs identical prefill and decoding at every size.

Table 13: Deployment cost as the experience bank grows. Mem−π is bank-independent after distillation, while retrieval pays in index size, latency and accuracy.

Bank entries	RAG index	Build time	RAG p50 / p95	RAG SR	Mem−π latency	Mem−π SR
3.2K	22 MB	0.7 s	0.02 / 0.04 s	31.4	1.8 s	**43.1**
32K	210 MB	5.8 s	0.03 / 0.08 s	30.8	1.8 s	**43.1**
320K	2.0 GB	54 s	0.07 / 0.24 s	29.6	1.8 s	**43.1**
3.2M	19.4 GB	9.4 min	0.24 / 0.92 s	28.4	1.8 s	**43.1**

At the original bank size RAG remains far cheaper. The trade-off reverses only as historical memory accumulates: Mem−π holds 1.8 s and 43.1% throughout because it no longer touches the bank at inference, while RAG's p95 latency rises more than twentyfold and its success falls by 3.0 pp through retrieval interference.

C.9 VISUAL OBSERVATIONS FOR MEMORY GENERATION

In addition to textual states, the web navigation tasks provide multimodal observations such as screenshots. We build a vision-language variant of Mem−π using `Qwen2.5-VL-7B-Instruct` as the memory-policy backbone and compare it against the text-only `Qwen2.5-7B-Instruct` variant on WEBARENA domains.

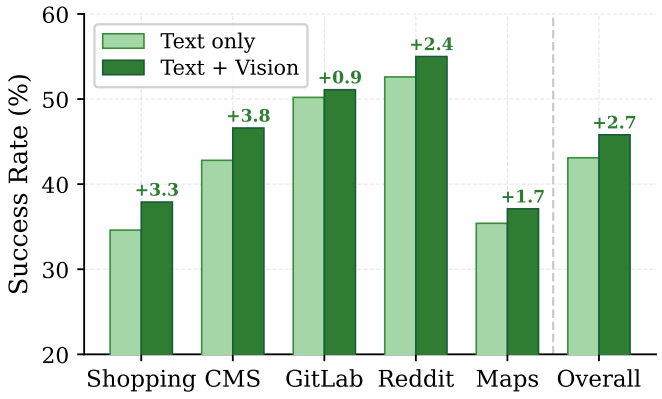


Figure 8: Performance of Mem−π with and without visual observations on WEBARENA.

Visual observations provide consistent gains, with larger benefits in visually grounded domains. The multimodal variant outperforms its text-only counterpart across all domains, improving overall SR by 2.7 pp. The gain is largest on CMS (+3.8 pp) and Shopping (+3.3 pp), where page layout and product images carry grounding signal that is hard to recover from text, and smallest on GitLab (+0.9 pp), whose code-centric content renders to little additional visual information. All main-paper results use the text-only variant.

Table 14: Full main results over three seeds. Subscripts show standard deviations. **Bold**: best per column. Underline: second best.

Method	WEBARENA	WORKARENA	ALFWORLD	LAB	Avg.
Base Agent	$27.1_{\pm 0.4}$	$41.8_{\pm 0.6}$	$84.8_{\pm 0.8}$	$26.8_{\pm 0.5}$	45.1
RAG	$31.4_{\pm 0.5}$	$42.2_{\pm 0.7}$	$85.5_{\pm 0.9}$	$28.5_{\pm 0.6}$	46.9
Mem0	$31.9_{\pm 0.6}$	$43.0_{\pm 0.8}$	$86.2_{\pm 1.0}$	$30.0_{\pm 0.7}$	47.8
Memory-R1	$33.2_{\pm 0.7}$	$44.1_{\pm 0.9}$	$86.7_{\pm 1.1}$	$31.2_{\pm 0.8}$	48.8
MemRL	$34.0_{\pm 0.8}$	$44.5_{\pm 1.0}$	$87.1_{\pm 1.2}$	$31.9_{\pm 0.9}$	49.4
Mem-π (Stage 1)	$\underline{35.0_{\pm 0.6}}$	$\underline{45.5_{\pm 0.8}}$	$\underline{89.5_{\pm 0.9}}$	$\underline{34.1_{\pm 0.7}}$	$\underline{51.0}$
Mem-π	$\mathbf{43.1_{\pm 0.5}}$	$\mathbf{49.7_{\pm 0.7}}$	$\mathbf{91.4_{\pm 0.8}}$	$\mathbf{36.7_{\pm 0.6}}$	**55.2**

C.10 ABSTENTION TRAINING DYNAMICS

We report per-benchmark final abstention rates and track the evolution of abstention probability during Stage 2 training, starting from the symmetrically initialized Stage 1 checkpoint (≈50% initial abstention probability).

Abstention rates are benchmark-dependent and converge quickly. Final abstention rates vary across benchmarks: 34% on WEBARENA, 28% on WORKARENA, 21% on ALFWORLD, and 39% on LAB (Figure 6a). The higher rate on LAB reflects a greater proportion of novel configurations where no relevant experience exists. The lower rate on ALFWORLD reflects its narrow household-task distribution, where memorized patterns apply broadly.

Symmetric initialization is essential for cold-start exploration. Figure 6b shows that with symmetric initialization, the abstention rate converges within approximately 100 training steps to a stable task-appropriate level. Without symmetric initialization, the `[ABSTAIN]` token starts with near-zero probability and recovers only slowly to approximately 21%, never reaching the converged rate of the symmetric variant. This confirms that balanced initialization of the decision tokens is critical for enabling RL to explore both branches effectively from the start.

C.11 FULL RESULTS WITH STANDARD DEVIATIONS

Table 14 reports per-benchmark means and standard deviations across three independent seeds. Mem-π ranks first on every benchmark and on the four-benchmark average (55.2), and Mem-π (Stage 1) ranks second on every benchmark and on the average (51.0). Standard deviations are at most 1.2 pp and smaller than the corresponding gaps between the top two means.

C.12 PROMPT TEMPLATES

We list the prompt templates used by the memory model π_{mem} and the base agent on each benchmark. The memory model shares a single template across benchmarks — emit `[ABSTAIN]` or `[GENERATE] <hint>` — and only the input modality varies: a multimodal variant (used for WEBARENA and WORKARENA with screenshots) and a text-only variant (used for ALFWORLD, LAB, and the text-only ablation of WEBARENA/WORKARENA). The base agent is benchmark-specific: action languages, observation formats, and termination conditions all differ. Whenever π_{mem} emits `[GENERATE]`, the resulting hint is inserted into the agent prompt as a Strategic Guidance block; on `[ABSTAIN]`, the standard prompt is used unchanged.

Memory model prompt — vision-language

[system]
You are a hinter. Given a web task plus a screenshot of the current page and a short screen description, write one concise, actionable hint grounded in what is actually visible, or abstain when you cannot help.
[user]
`<image: {task_id}.png>`
Screen description:
`{screen_description}`
Task:
`{task_description}`
First decide: output `[ABSTAIN]` if you cannot provide useful guidance, or `[GENERATE]` followed by a concise, actionable hint that references elements visible in the screenshot or screen description.

Memory model prompt — text

[system]
You are a hinter. Given a web task, write one concise, actionable hint that helps the agent take its next step, or abstain when you cannot help.
[user]
Task:
`{task_description}`
First decide: output `[ABSTAIN]` if you cannot provide useful guidance, or `[GENERATE]` followed by a concise, actionable hint.

Agent prompt — WEBARENA

[system]
You are a professional web browsing agent assistant that can fulfill user's high-level instructions. Given simplified html of the browsed webpage at each step, you plan operations in python-style pseudo code using provided functions.
You first think about the reasoning process as an internal monologue and then decide an action. The reasoning process and answer are enclosed within `<think> </think>` and `<answer> </answer>` tags, respectively, i.e. responding in the following format:

```
<think> ... </think>
<answer> ... </answer>
```

 # More details about the code action. Your action should be readable, simple. Generate **ONLY ONE ACTION** in one round. Predefined functions:

```
def do(action, argument, element):
    """A single browsing operation on the webpage.
    action in {"Click", "Type", "Search", "Hover",
               "Scroll Up", "Scroll Down", "Press Enter",
               "Select Dropdown Option"}
    argument: optional; only for "Type", "Search",
              "Select Dropdown Option".
    element : optional; specific element id in the html
              (required for "Click", "Type", "Search",
              "Select Dropdown Option", "Hover").
    """
def exit(message):
    """End the browsing process; pass an answer if asked."""
def go_backward():
    """Go back to the previous page."""
def go_forward():
    """Go forward to the next page."""
```

Examples (excerpt):

```
- <think> # Element: the 'REPORTS' section on the left sidebar </think>
  <answer> do(action="Click", element="7") </answer>
- <think> # Element: the 'Period' dropdown </think>
  <answer> do(action="Select Dropdown Option",
               argument="Month", element="20") </answer>
- <think> reasoning process here </think>
  <answer> do(action="Scroll Down") </answer>
```

- <think> reasoning process here </think>
  <answer> exit(message="The top-3 best-selling
  products in January 2023 are: 1") </answer>

REMEMBER:

- Generate **ONLY ONE ACTION** per round.
- Don't reference an element you cannot see in the html.
- Use `# Element` to describe the chosen element; use `# Note` to record useful information.
- If stuck in a loop, switch methods or actions.
- For `exit`, `go_backward`, `go_forward`, follow the function format strictly (no `do("Exit", ...)`).
- Never use the browser's address bar to navigate.

[user]
URL: <url> Tabs: <tabs> Previous action: <prev_action>
Objective: <task_intent>
Strategic Guidance (when π_{mem} emits `[GENERATE]`; otherwise omitted):
<generated_hint>
Accessibility tree: <axtree>
[assistant]
<think>...</think> <answer> do(...) </answer>

Agent prompt — WORKARENA

[system]
You are an agent completing a web task in a ServiceNow instance. At each step you will see the accessibility tree of the current page and must produce exactly one action.
 # Action space. You interact with the page via the browsergym `HighLevelActionSet` (`bid` subset). Each interactive element in the accessibility tree has a `bid` in square brackets, e.g. `[a123] button 'Submit'`. Use the `bid` string when calling actions. The 12 available actions are:

```
noop(wait_ms=1000)               # wait for the page to update
scroll(delta_x, delta_y)         # scroll the page (px)
fill(bid, value)                 # fill a textbox / contenteditable
select_option(bid, options)      # select option(s) in a combobox
click(bid, button='left')        # click an element
dblclick(bid, button='left')     # double-click an element
hover(bid)                       # hover an element
press(bid, key_comb)             # focus and press a key combo
focus(bid)                       # focus an element without clicking
clear(bid)                       # clear an input field
drag_and_drop(from_bid, to_bid)  # drag-and-drop between two bids
upload_file(bid, file)           # upload a file to a file-input bid
```

Do NOT use CSS selectors, playwright locators, or element text.
 # Output format. Respond with exactly one reasoning block and one action block:

```
<think>
Your reasoning about what to do next.
</think>
<answer>
action_name(arguments)
</answer>
```

Example:

```
<think>
I see [a324] textbox 'Short description'. I need to fill
it with the required value from the goal.
</think>
<answer>
fill("a324", "Server outage report")
</answer>
```

[user]
Goal: <task_intent>
Strategic Guidance (when π_{mem} emits `[GENERATE]`; otherwise omitted):
<generated_hint>

Accessibility tree: `<axtree>`
[assistant]
`<think>...</think> <answer>action(...)</answer>`

Agent prompt — ALFWORLD

[system]
You are an embodied agent in a TextWorld household environment. Each turn, output EXACTLY ONE short text command. Do NOT include any prefix like `Action:` – output JUST the command. The episode terminates automatically when the goal is satisfied.
 # Action space. The available text commands are:

```
go to <receptacle>              # navigate to a receptacle
open <receptacle>               # open a closed receptacle
close <receptacle>              # close an open receptacle
take <object> from <recep>      # pick up an object
put <object> in/on <recep>      # place a held object
heat <object> with <recep>      # use microwave / stove
cool <object> with <recep>      # use fridge
clean <object> with <recep>     # use sink / cloth
use <object>                    # use a tool (lamp, ...)
examine <object|receptacle>     # inspect target
look                            # describe current location
inventory                       # list held objects
```

Receptacles and objects are referenced by name plus an integer index (e.g. `fridge 1`, `apple 2`).
 # Output format. Output exactly one command per turn, no extra text.
When π_{mem} emits `[GENERATE]`, the following is appended to the system prompt:

```
A memory hint module suggests the following procedural script.
Use it as advisory; verify with `look` or `examine` before
committing:
<hint>
{hint}
</hint>
```

[user] (first turn)
Goal: `<goal>`
Initial observation: `<observation>`
What is your first action? Output JUST the command.
[assistant]
`<single text command>`

Agent prompt — LAB (OS / DB)

[system] **OS subset (Bash skills).**
You are a sysadmin operating a Linux shell to complete a task.
 # Action space. Two actions are available:

````
Act: bash             # run shell commands; observe stdout
```bash
<commands>
```

Act: finish           # signal task completion
````

 # Output format. Each turn, output exactly one of the two action blocks above.
When π_{mem} emits `[GENERATE]`, the following is appended:

```
A memory hint module suggests the following procedural script.
Treat it as advisory -- verify with bash before trusting any
specific command:
<hint>
{hint}
</hint>
```

[system] **DB subset (SQL skills).**

You are a SQL agent operating against a SQLite database (with MySQL-compatible shims for `md5`, `concat_ws`, `substring`, `if`).
 # Action space. Two actions are available:

```
Action: Operation     # run one SQL statement; observe rows
```sql
<one SQL statement>
```

Action: Answer   # provide the final answer
Final Answer: <answer>
```

 # Output format. Each turn, output exactly one of the two action blocks above. For SELECT-style tasks, list rows as comma-separated tuples (e.g. `(1001, 'Alice', '2023-01-05')`); for INSERT/UPDATE/DELETE, write a short string (e.g. `done`); the env scores by hashing the post-mutation table.
[user] (first turn, both subsets)
Task instruction: `<instruction>`
What is your first action?
[assistant]
`<single Act/Action block per the format above>`

C.13 Additional Case Studies

We complement case analysis (Figure 5) by examining one representative task per Venn region. The eight regions partition the test split into qualitatively distinct outcome patterns, summarized below. **Region 001** contains Mem-π-only successes, *Pattern 1* of the main text where generation reaches what retrieval cannot. **Region 101** contains tasks Base and Mem-π solve but RAG breaks, *Pattern 2* where abstention recovers from RAG noise. **Region 011** contains tasks where memory is needed and both RAG and Mem-π succeed. **Region 111** contains tasks easy enough for any approach. **Region 110** contains Mem-π regressions, where Base and RAG solve but Mem-π fails. **Region 010** contains RAG-only successes, where retrieval surfaces a verified procedure that generation re-orders incorrectly. **Region 100** contains tasks where any memory hurts and only the unassisted base agent succeeds. **Region 000** contains tasks no method solves, typically reflecting environment- or tool-level limitations. We present a representative case for each region below in the same layout as Figure 5. Long task queries and hints are abridged with ellipses, keeping only the contrastive sub-strings.

Region 001: Mem-π wins Pattern 1. Generation reaches what retrieval cannot. 19 tasks
Case A1 (Task 8). *Top search terms (Magento admin).*
Task: "List the top 3 search terms in my store."
RAG: ✗ "…locate the 'Top Search Terms' table…read the *first two rows* and report each 'Search Term' with its 'Uses' count."
Mem-π: ✓ "…take the *first three rows* from the 'Search Term' column and output just those term names…"
Why: The retrieved hint inherits its source's top-2 specification verbatim. Generation conditions on the explicit "3" in the query and rewrites the count.

Region 101: Mem-π wins Pattern 2. Abstention recovers from RAG noise. 10 tasks
Case A2 (Task 41). *Storage option search (shopping).*
Task: "…find the best storage option to fit all 40 [Switch game] cards."
RAG: ✗ "…search…for 'game card case' or 'cartridge holder', open listings that show 40+ slots…"
Mem-π: ✓ [ABSTAIN]
Why: The retrieved hint locks the search to a narrow product family that excludes broader organizers. Abstention restores the base agent's open search, which selects an organizer that better satisfies "best".

Region 011: Memory is needed, RAG and Mem−π both solve. 14 tasks
Case A3 (Task 2). *Review keyword count (Magento admin).*
Task: "Tell me the number of reviews that our store received... that mention term 'best'."
RAG: ✓ "... 'Marketing' > 'User Content' > 'All Reviews', click 'Reset Filter', enter 'best' in the 'Review' filter... click 'Search', then read the 'records found' counter..."
Mem−π: ✓ "... same Magento navigation, enter the keyword in the 'Review' filter row... read the 'records found' count..."
Why: The retrieved entry is exactly aligned with the query, so retrieval and generation converge on the same correct procedure.

Region 111: All three solve. 28 tasks
Case A4 (Task 5). *Comment-score query (Reddit forum).*
Task: "... count of comments that have received more downvotes than upvotes for the user who made the latest post on the Showerthoughts forum."
RAG: ✓ "... set sort to 'New', open the latest post, click the author's name, open 'Comments', count comments with a leading −..."
Mem−π: ✓ "... open Showerthoughts, sort 'New', open top post, click author, 'Comments', count comments with negative score..."
Why: The agent already follows the standard navigation. Both memory variants reinforce the natural plan rather than introducing new structure.

Region 110: Mem−π regresses, Base and RAG solve. 4 tasks
Case A5 (Task 88). *Theme preview (Magento admin).*
Task: "Preview the Magento Blank theme for my shop."
RAG: ✓ "... find the 'Magento Blank' row... click the 'View' link *in the Action column*..."
Mem−π: ✗ "... click the theme name in the grid... open the preview... *avoid clicking the Action column*..."
Why: Generation produces a plausible but inverted instruction, telling the agent to avoid the very element that solves the task. Retrieval, drawing on the source task's verified flow, preserves the correct "Action → View" click.

Region 010: RAG-only success. 6 tasks
Case A6 (Task 17). *Pending review count (Magento admin).*
Task: "What is the total count of Pending reviews amongst all the reviews?"
RAG: ✓ "... 'Marketing' > 'User Content' > 'Pending Reviews', read the 'X records found' label... if unsure about filters, click 'Reset Filter'..."
Mem−π: ✗ "... 'Pending Reviews', *click 'Reset Filter' to clear any saved filters*, then read the records-found label..."
Why: The two hints are nearly identical, but the generated version starts with an unconditional 'Reset Filter' click. In our trajectory this clears a server-side default that is needed for the page to populate, while the retrieved hint defers the reset to a later checkpoint. The ordering matters and retrieval preserves the source's correct sequence.

Region 100: Both memory methods hurt, only Base solves. 3 tasks
Case A7 (Task 36). *Spend aggregation across orders (shopping).*
Task: "How much I spent on food shopping... from mid Jan to the end Jan 2023."
RAG: ✗ "... 'My Orders'... scroll to the bottom and use the pagination links by their text... open each order from Jan 15–31 via the 'View Order' link... sum item subtotals..."
Mem−π: ✗ "... 'My Orders', set 'Show per page' to 50... for each order dated 1/15–1/31 click 'View Order'... record the 'Grand Total' and sum..."
Why: Both hints prescribe a multi-page enumeration plus arithmetic that exceeds the agent's effective context window in this trajectory. The unassisted base agent shortcut, summing only the few orders that appear after applying a date filter, happens to be sufficient. Memory hurts here by enforcing a thorough but unsustainable plan.

Region 000: No method solves. 81 tasks
Case A8 (Task 0). *Bestseller report (Magento admin).*
Task: "What are the top-3 best-selling product in Jan 2023."
RAG: ✗ "…'Reports > Products > Bestsellers'…'Period' = 'Month', 'From' 01/01/2023, 'To' 01/31/2023…if you see 0 records, refresh stats via the 'statistics'/'here' link…"
Mem-π: ✗ "…same report…if it says 'We couldn't find any records', click the 'statistics'/'here' link or go to 'Reports > Statistics'…"
Why: Both hints correctly describe the report path, but the underlying Magento instance does not auto-refresh the bestseller statistics within the agent's step budget. The failure is a tool-level limitation rather than a memory-quality issue. We include this region to show that adaptive memory does not manufacture progress on tasks beyond the agent's tool capabilities.

Summary of regions. The two highlighted regions tied to Mem-π's wins over RAG, regions 001 and 101, account for $19 + 10 = 29$ tasks where adaptive generative memory contributes through mechanisms unavailable to retrieval. The shared-success regions, 011 and 111, contain $14 + 28 = 42$ tasks and confirm that adaptive memory does not displace retrieval when retrieval is well aligned. The opposing regions, 110, 010, and 100, cost $4 + 6 + 3 = 13$ tasks where Mem-π fails and at least one of the two baselines succeeds. The dominant sub-mode is region 010 where retrieval surfaces a verified procedure that generation re-orders incorrectly. The all-failure region 000 includes tasks dominated by environment- or tool-level limitations rather than memory quality.

Consistency with Table 1. The eight regions partition all 165 test tasks in one representative evaluation run. The Base Agent solves $3 + 4 + 10 + 28 = 45$ tasks (27.3%), RAG solves $6 + 4 + 14 + 28 = 52$ (31.5%), and Mem-π solves $19 + 10 + 14 + 28 = 71$ (43.0%). These single-run rates are within 0.2 pp of the three-seed means of 27.1, 31.4 and 43.1 in Table 1.