

Retrieval-Augmented LLM Agents: Learning to Learn from Experience

Thomas Palmeira Ferraz Romain Deffayet Vassilina Nikoulina
Hervé Déjean Stéphane Clinchant
NAVER LABS Europe, France

{thomas.palmeira,vassilina.nikoulina} @ naverlabs.com

Code: 🌀 [naver/RAGents](#) Models: 😊 [naver/ragents](#)

Abstract

While large language models (LLMs) have advanced the development of general-purpose agents, robust generalization to unseen tasks remains challenging. Two common approaches are supervised fine-tuning and training-free memory-augmented generation using retrieved experience; yet both have limitations: fine-tuning often fails to extrapolate to new tasks, while experience retrieval often underperforms compared to supervised baselines. In this work, we combine these approaches and study how retrieval-augmented LLM agents can learn to use retrieved trajectories in-context. First, we establish a strong LoRA fine-tuning baseline that outperforms several state-of-the-art agent training pipelines. Second, we analyze key design choices for experience retrieval, including storage, querying, and trajectory selection. We then integrate experience retrieval directly into the fine-tuning process, finding that this substantially improves generalization to unseen tasks. Finally, we show that these gains often persist with imperfect experience and, even when agents reuse their own failed attempts without test-time parameter updates. Overall, our results establish simple episodic retrieval as a strong foundation for agent memory and **retrieval-aware fine-tuning as a practical and effective framework for building agents that learn to learn from experience.**

1 Introduction

Large language models (LLMs) have recently enabled general-purpose agents that can interact with textual environments, execute multi-step plans, and learn to solve families of tasks with minimal task-specific engineering (Schick et al., 2023; Patil et al., 2024; Wang et al., 2024a; Jimenez et al., 2024; Yang et al., 2024b; Park et al., 2023). In addition to sophisticated prompting, reflection and refinement pipelines (Yao et al., 2023; Shinn et al., 2023), adaptation through parameter-efficient fine-tuning (Hu et al., 2022; Han et al., 2024) is a natural

Method	ALFWorld
Prompting	
Zero-shot	29.9
ReAct (Yao et al., 2023)	17.1 ^a
ITP _I (Liu et al., 2026b)	35.7
Training-Free Memory-Augmented	
Mem0 (Chhikara et al., 2025)	33.6 ^b
A-MEM (Xu et al., 2025)	34.7 ^c
AgeMem-noRL (Yu et al., 2026)	37.9
Memory Bank (Zhong et al., 2024)	40.3 ^d
Reflexion (Shinn et al., 2023)	42.7 ^e
EXPRAG baseline (ours)	83.6
Supervised Fine-tuned	
NAT (Wang et al., 2024b) + ReAct	66.4 ^f
IWM (Zhang et al., 2026c)	70.3
Self-Reflection (Zhang et al., 2026c)	71.1
ETO (Song et al., 2024) + ReAct	79.9 ^f
SFT + ReAct (Yao et al., 2023)	80.7 ^f
SAND (Xia et al., 2025)	85.0
ITP _R (Liu et al., 2026b)	85.1
Rule-based Expert	89.6
LoRA baseline (ours)	94.1

Table 1: **Complex solutions can underperform a strong baseline.** Success rate results for **Qwen 2.5-7B** on official held-out split of ALFWorld (valid-unseen) for training-free and supervised fine-tuned methods. This cross-paper comparison should be interpreted qualitatively. Unless noted, values come from original works; superscripts denote third-party reports used only when original papers do not report that model result: ^aLiu et al. (2026b), ^bXia et al. (2026), ^cYu et al. (2026), ^eFeng et al. (2025), ^dZhang et al. (2025a), ^fFei et al. (2025).

step for building agents (Chen et al., 2023; Qiao et al., 2024). This approach often yields strong performance on tasks seen during training. However, we argue that **truly useful agents should be able to perform a broad range of tasks, including previously unseen ones**, within the environment in which they were trained to operate. Yet standard held-out benchmarks often evaluate tasks that remain very similar to those seen during training.

The in-context learning capabilities of modern LLMs make retrieval-augmented generation (RAG) an attractive inference-time adaptation mechanism,

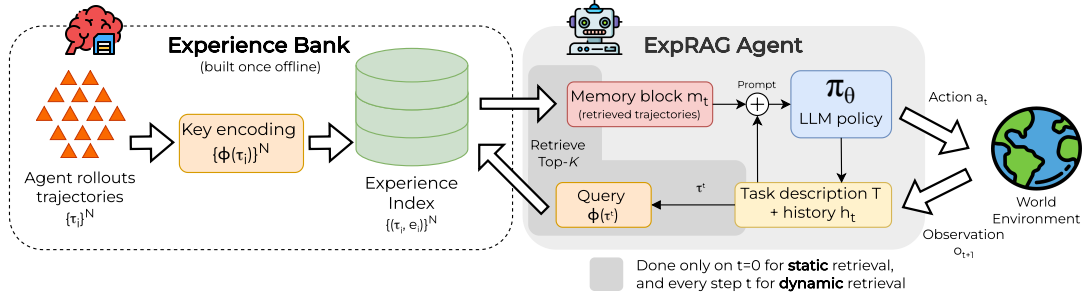


Figure 1: **EXPRAG Agent overview.** EXPRAG augments an LLM agent with retrieval over past experience. Offline, past trajectories are encoded into an experience index. At inference, task and history are used to retrieve top- K relevant trajectories, which are assembled into a memory block and provided to LLM policy to predict the next action. Retrieval is done once at $t = 0$ in the **static** setting, or refreshed every step in the **dynamic** setting.

by providing relevant external context (Izcard et al., 2023; Ram et al., 2023). For agents, a natural source of external context is prior experience, whether drawn from the agent’s own rollouts, other agents, or expert demonstrations. Retrieving relevant experience and feeding it in-context to the model has been proposed in prior work such as RAP (Kagaya et al., 2024), and contemporaneous work by Wei et al. (2025) coined the term EXP-PRAG, which we adopt in this paper.

This paper studies how to train retrieval-augmented LLM agents that can learn from retrieved experience in-context, enabling better generalization to unseen tasks within the same environment. We focus on a practical setting: moderately-sized open models adapted with supervised LoRA fine-tuning (Hu et al., 2022), operating in text-based environments with multi-turn decision making.

Contribution ①: Strong fine-tuning and retrieval-augmented baselines. Our first contribution is to provide effective recipes for both fine-tuning and retrieval-augmented inference. We noticed that the performance of fine-tuned models varies greatly in existing literature, and therefore propose our own LoRA baseline. In Table 1, we show that on the widely used ALFWorld benchmark, our LoRA baseline outperforms existing training pipelines, including more elaborate agentic methods. The same table further suggests that elaborate memory-augmented pipelines (without training) can provide limited improvement or even underperform compared with a simple experience-retrieval baseline (Best EXP-PRAG), consistent with Wei et al.’s (2025) observation that simple episodic retrieval is still competitive with self-evolving memory systems. Both results call for **bench-**

marking against strong simple baselines before attributing gains to complex memory architectures.

Contribution ②: A careful examination of EXP-PRAG. Second, although RAG has mature recipes for document retrieval, yet no analogous recipe is established for retrieving and using agent trajectories. In Section 4.3, we systematically study the agent-specific design choices: how to store experience, how to build a retrieval query, how much and when to retrieve, and how the source of indexed trajectories and the backbone model affect performance. These results provide a practical recipe for robust and effective adaptation through EXP-PRAG.

Contribution ③: Training agents to use retrieved experience (EXP-PRAG-LoRA). We propose to combine LoRA fine-tuning with EXP-PRAG, training the policy to use retrieved experience rather than introducing it only at inference time. In Sec. 4.4, EXP-PRAG-LoRA substantially improves generalization to unseen task groups while retaining strong performance on seen tasks, showing that memory gains depend not only on retrieving useful experience, but also on learning how to use it.

Contribution ④: Learning from imperfect and self-generated experience. Finally, we stress-test EXP-PRAG-LoRA beyond matched expert demonstrations using sparse, mismatched, stronger-LLM-generated, and self-generated experience at inference time. It often remains robust to imperfect but relevant memory and, in some cases, can even benefit from its own failed attempts without parameter updates.

Taken together, these results establish simple episodic retrieval as a strong foundation for agent memory, while showing that effective memory de-

depends not only on what experience is stored and retrieved, but also on whether the agent has learned how to use it, a capability that can transfer from expert experience during training to unseen tasks and imperfect experience at inference.

2 Background and Related Work

LLM agents have been studied in interactive settings such as text environments, web navigation, and tool use. We position our work at the intersection of agent training and episodic retrieval: the agent retrieves past trajectories, rather than documents, and we study how that retrieval should be used both at inference time and during fine-tuning.

LLM-based agents. A common strategy for LLM agents is to define a fixed interaction protocol and adapt an instruction-tuned model on expert trajectories via supervised fine-tuning (SFT) / behavior cloning (Chen et al., 2023, 2024a). This usually improves in-domain scores and output-format reliability, but generalization to unseen tasks often remains limited due to distribution shift and compounding errors in sequential decision making (Song et al., 2024). To improve robustness, one line of work initializes from SFT and further learns through trial-and-error, often with reinforcement learning or preference-based updates (Song et al., 2024; Zhang et al., 2026c; Feng et al., 2025). However, we find that simple SFT with LoRA can already outperform more complex methods in our settings, so we instead explore an orthogonal direction: experience retrieval.

Retrieval-augmented generation (RAG). Classical RAG augments a parametric generator with retrieved passages from an external corpus (Lewis et al., 2020; Guu et al., 2020; Ram et al., 2023), typically for tasks such as open-domain QA and knowledge-intensive generation. Some systems integrate retrieval more tightly into the architecture, training the model to process retrieved context (e.g., RETRO, Borgeaud et al. 2022; Atlas, Izacard et al. 2023; and RAFT, Zhang et al. 2024). We borrow this retrieval-as-context view, but replace *semantic* memory with *episodic* memory: instead of retrieving documents to ground factual generation, the agent retrieves prior interaction trajectories to guide action selection in a new situation.

Memory for agents. Broader agent-memory systems extend retrieval with explicit writing, compression, reflection, and control mechanisms. Re-

cent surveys distinguish classical RAG from *agent memory*, which typically involves (i) what gets written (episodic traces, feedback, reflections, skills), (ii) how memory is structured or compressed (summaries, abstractions, procedural artifacts), and (iii) control over when and how memory is retrieved and updated (e.g., step-wise loops, learned or heuristic read/write policies) (Hu et al., 2025; Zhang et al., 2025c; Wu et al., 2025; Yu et al., 2026). Representative examples include systems that manage long-term context or learn read/write policies (e.g., MemGPT, Packer et al. 2023), and methods that store reflective summaries or compact experience units rather than full trajectories, such as Reflexion (Shinn et al., 2023), A-MEM (Xu et al., 2025), and Memento (Zhou et al., 2025). Unlike these approaches, we do not propose a new memory controller, online memory update rule, or reflection mechanism. Our memory is intentionally simpler: a fixed, read-only bank of full trajectories retrieved into context. Closest to our setting, contemporaneous work by Wei et al. (2025) compares inference-only episodic retrieval with self-evolving read-write memory, using strong proprietary models. By contrast, we study weaker open-source models under both inference-time and fine-tuning retrieval augmentation pipelines.

Learning from experience. Beyond storing past interactions, a complementary line of work studies how agents improve from them. Classical meta-learning trains models to adapt rapidly from new examples or trajectories (Hospedales et al., 2022; Finn et al., 2017), while recent LLM-agent methods reuse past trials either through external memory or parameter updates. Reflexion and ExpeL transform prior interactions into reusable verbal feedback or distilled experience (Zhao et al., 2024; Shinn et al., 2023), whereas methods such as ETO, NAT, and Early Experience use successful or failed trajectories as supervision for further training (Song et al., 2024; Wang et al., 2025a; Zhang et al., 2026c). Our setting is complementary: we keep experience as raw retrieved trajectories and train the policy itself to use them in context, then test whether this ability transfers to sparse, teacher-generated, and self-generated experience without test-time parameter updates.

Position of this work. Our setting differs from prior memory-based agents along four experiment-relevant axes: we retrieve full trajectories rather than summaries or reflections, keep memory fixed

rather than updating it online, study retrieval during training as well as at inference time, and distinguish gains from retrieval and from parameter updates. Accordingly, our goal is not to propose a rich read–write memory architecture, but to establish strong foundations for fine-tuning augmented with episodic retrieval: (i) we propose strong baselines and identify their main drivers of success, (ii) we investigate whether retrieval-augmented fine-tuning teaches agents a reusable ability to learn from retrieved experience on unseen tasks, including under imperfect experience, and (iii) we characterize when this adaptation succeeds or fails as the availability, source, and quality of experience change.

3 EXPRAG: Experience Retrieval-Augmented Generation

Purely parametric learning (e.g. LLM fine-tuning with LoRA) can fit the training-tasks distribution well, but may fail to generalize to new tasks. Retrieval adds an adaptive, inference-time mechanism: the agent can condition its generation on relevant past experience and apply it in-context without updating parameters. In that case, our goal is to **build agents that learn to reliably use retrieved experience when it is available, rather than simply memorize cases seen during training**. In this section we motivate the use of retrieval for in-context learning and formalize our approach for experience RAG (which we refer to as EXPRAG).

LLM-based Agents. We consider an agent interacting with a textual environment over at most T decision steps for a task description $\mathcal{T}$ provided as text. At each step t , the agent observes o_t , outputs an action a_t , and receives the next observation o_{t+1} from the environment or a terminal flag when the task is completed. We denote the full trajectory by $\tau = (\mathcal{T}, o_1, a_1, \dots, o_{T-1}, a_{T-1}, o_T)$, and the trajectory history available at step t by $h_t = (\mathcal{T}, o_1, a_1, \dots, o_t)$. An LLM policy π_θ defines a conditional distribution over actions given history, i.e., $a_t \sim \pi_\theta(\cdot \mid h_t)$. Our goal is to learn π_θ that generalizes to unseen tasks within the same environment, where interaction dynamics are fixed but required action sequences may differ. This includes tasks requiring action types not observed during training (e.g., task is to heat an object but agent was trained to clean an object).

Encoding Agent Trajectories as Multi-turn LLM Chats. Given a trajectory τ , we serialize the interaction as a multi-turn chat $chat(\tau)$ with the base model’s native chat template, mapping observations to user turns and actions to assistant turns. At step t , the policy conditions on $chat(h_t)$ and generates an action as a natural-language token sequence $a_t = (a_t^1, \dots, a_t^{n_t})$, autoregressively $a_t^i \sim \pi_\theta(\cdot \mid chat(h_t), a_t^{<i})$. This serialization follows multi-turn agent formulations (Wang and Ammanabrolu, 2025; Wang et al., 2025b; Jin et al., 2025), with supervised next-token prediction on assistant tokens as objective. It contrasts with more recently employed stepwise encodings (Zhang et al., 2026c; Wei et al., 2025; Feng et al., 2025), which serialize (h_t, a_t) as independent samples and re-encode history at every step. In our early explorations, we observe similar task performance but substantially faster training for multi-turn chat due to KV-cache reuse across turns.

Indexing and Retrieval. We maintain an index $I = \{(\tau_i, e_i)\}_{i=1}^N$ of textual trajectories and their key embeddings, where $e_i = \phi(\tau_i)$ is computed by a trajectory encoder $\phi(\cdot)$. Trajectories are stored as raw chat-formatted data without further filtering or aggregation. At decision step t , we build a textual query q_t from the current agent context and retrieve the top- K trajectories by nearest-neighbor search:

$$(\tau^1, \dots, \tau^K) = \arg \max_{1 \leq i \leq N} \text{topK } \phi(q_t) \cdot e_i.$$

Experience-Conditioned Generation. We then format retrieved trajectories into a memory block $m_t = \text{system}(\tau^1, \dots, \tau^K)$, inserted in the system prompt. When available, we distinguish successful and unsuccessful trajectories in the memory block prompt. The action a_t^i is generated autoregressively conditioned on both memory m_t and dialogue history $chat(h_t)$:

$$a_t^i \sim \pi_\theta(\cdot \mid m_t, chat(h_t), a_t^{<i}).$$

For EXPRAG, we assume that, for out-of-distribution tasks unseen during training, a small set of in-domain reference trajectories can be collected and used as in-context examples. These trajectories may come either from expert demonstrations (e.g., rule-based or stronger models) or from previously recorded agent rollouts generated with a strong prompt. We provide further details on index construction and retrieval in Appendix C.

4 Experiments

We evaluate retrieval-augmented agents on text-based interactive benchmarks with multi-step decision making. Our goals are to measure gains on unseen tasks and understand which design choices matter most.

4.1 Dataset and Benchmarks

Environments. We evaluate on two commonly used text-based embodied environments: **ALF-World** (Shridhar et al., 2021) and **Science-World** (Wang et al., 2022). ALFWorld involves household manipulation tasks with binary episode outcome (success/failure). ScienceWorld contains science-oriented tasks aligned with an elementary-school science curriculum and provides a dense episode score in $[-1, 1]$, with success at 1. For consistency, we report **success rate**. For ScienceWorld, we convert the episode outcome to a binary success/failure score.

Benchmarks primarily measure within-group generalization. Both benchmarks associated with the environments provide official held-out splits (ALFWorld: *valid-unseen*; ScienceWorld: *test*), where tasks are held out within predefined task groups (ALFWorld contains 6 “*task-types*” while ScienceWorld contains 10 “*topics*”). However, sampling splits within each group makes the test set still close to the training distribution and thus less realistic, as agents are evaluated on tasks very similar to the ones they were trained on. This is demonstrated by the performance obtained in preliminary runs, reported in Table 1, where standard LoRA fine-tuning already achieves very strong performance on ALFWorld, surpassing rule-based expert and nearly saturating the benchmark.

Building Out-of-Distribution Benchmarks. To evaluate stronger out-of-distribution generalization and the ability of the model to adapt at test time from past experiences with EXPRAG, we partition the task groups into **easy** (used for training trajectory collection and fine-tuning) and **hard** (held out for evaluation). In our experiments, we report results separately on easy and hard tasks to isolate generalization. Extra details on these splits are available in Appendix C.2.

4.2 Implementation Details

Data and Indexing. Across all experiments, we instantiate experience as scripted expert trajec-

ries generated by the environment-provided policy. This controlled choice allows us to study retrieval-augmented adaptation under a reliable memory source, avoiding an additional confound from noisy trajectories given that zero-shot backbones perform poorly. Importantly, these trajectories still contain failures and suboptimal segments, requiring the model to learn to identify which retrieved actions are relevant. We serialize trajectories using the chat template described in Section 3 (full templates in Appendix H). Unless otherwise stated, the retrieval index is built from the training split and includes both successful and unsuccessful episodes. For fine-tuning, we use only successful trajectories from the training split. We use successful trajectories from the corresponding validation split for model selection and report results on the corresponding held-out split; each split is further partitioned into Easy and Hard subsets.

System Prompt. For each environment, we use a minimal system prompt that specifies (i) the task setting, (ii) the action interface, and (iii) the response format (one action per turn). Concretely, at the beginning of each episode, we provide a static list of action templates valid in the entire environment (e.g., *go to [receptacle]*, *use [object]*), rather than step-specific instantiated valid actions. In contrast, some prior work provides per-step valid-action candidates during rollout (e.g., Feng et al., 2025; Zhang et al., 2026c); for example, if only *table 1* and *cabinet 4* are currently relevant, candidates such as *go to table 1*, *open cabinet 4*, (...) are shown. We avoid this additional guidance to reduce action-space leakage and evaluate stronger decision-making from context alone. Full prompt templates are provided in Appendix H.

Training Setup. In Section 4.4, we study adaptation with low-rank adapters (LoRA) (Hu et al., 2022). We supervise models on agent trajectories formatted as multi-turn chats, as described in Sec. 3. We compute cross-entropy loss on assistant tokens only. Key hyperparameters and implementation details are provided in Appendix C.

4.3 Retrieval-Augmented Inference without Training

We first evaluate EXPRAG only at inference time: the policy remains a frozen instruction-tuned model, and retrieval is enabled by prepending retrieved trajectories as in-context memory. Thus, in this section, all conditions use the same instruction-

tuned checkpoint and differ only in which retrieved trajectories are added to the prompt.

Ablations. We vary: (i) top- k ($K \in \{1, 2, 4\}$), (ii) index composition (**all/easy/hard** task-types available for retrieval), and (iii) retrieval mode. **Static retrieval** selects trajectories once at episode start. **Dynamic retrieval** re-queries at every step using the partial interaction history, which requires clearing the KV cache and re-encoding the prompt with updated retrieved examples up to the current observation.

For this experiment, static retrieval uses task descriptions as queries/keys; dynamic retrieval uses partial trajectories as queries and full trajectories as keys (JSON). This serialization affects retrieval embeddings only, not policy-side display. We investigate the impact of this choice in Appendix D.4.

Table 2 shows inference-only EXPRAG results for Ministral 3-8B on ALFWorld and ScienceWorld; other backbones are in Appendix D.2.

4.3.1 Discussion

Inference-only Experience Retrieval improves in all scenarios. Adding retrieved trajectories at inference time strongly improves over the NO RAG baseline on both benchmarks. The best all-task score rises from 4.48% to 64.18% on ALFWorld and from 10.40% to 35.24% on ScienceWorld. Gains appear in all scenarios, including when the retrieval index is mismatched with the evaluation split, indicating that retrieved trajectories help not only by recalling near-matching solutions, but also by providing reusable action patterns and subgoal strategies.

Choosing top- K : more experiences vs. context rot. Increasing K generally improves performance, but the gains are much larger on ALFWorld than on ScienceWorld. The best all-task score on ALFWorld rises from 41.04% at $K=1$ to 64.18% at $K=4$, whereas ScienceWorld improves at $K=2$ (26.62% $\rightarrow$ 34.67%) and only marginally to 35.24% at $K=4$, suggesting earlier saturation. We also observed more sensitivity in mismatched index settings. These results suggest that larger k helps when they add complementary evidence, but can hurt when they introduce noisy or weakly transferable context. Since inference cost grows roughly linearly with K , we use $K=2$ in later experiments as a better cost-performance trade-off. We report inference cost in Appendix I.

Refreshed context relevance vs. instability in dynamic retrieval. Dynamic retrieval helps mainly when the retrieval index is aligned with the evaluation setting, but is less stable than static retrieval under index mismatch. The largest failures occur when hard tasks retrieve from the easy index, suggesting that these trajectories may provide local action cues but often lack the task-specific strategy needed for later subgoals. A likely reason is that dynamic retrieval replaces retrieved context at every step, creating context churn and making later decisions more sensitive to retrieval quality (similar to effects reported by Zhu et al. (2026); Yang et al. (2026)). Overall, re-retrieval helps only when the refreshed context remains sufficiently relevant.

Index composition controls which tasks benefit. Performance is generally weaker when the retrieval index is mismatched with the evaluation setting, especially when hard tasks retrieve from the easy index, although some cross-split gains remain. This suggests that easy-task trajectories can still provide reusable sub-skills or partial reasoning cues. At larger K , ALFWorld often benefits more from the hard index, whereas ScienceWorld more often benefits from the easy index. A plausible explanation is that ALFWorld hard tasks largely recombine easy sub-tasks, so hard trajectories provide reusable multi-step plans. In ScienceWorld, however, hard trajectories may contain more failed, longer, or less reusable traces that an inference-only model cannot reliably filter.

Upshot. Retrieval-augmented inference is a strong training-free mechanism for adapting to unseen tasks; assuming sufficiently relevant retrieval, more experience generally helps up to the model’s effective context capacity, so a good practice is to use the highest top- K before performance saturates or degrades.

4.4 Retrieval-Augmented Fine-Tuning

Based on the results obtained with inference-only EXPRAG, we study fine-tuning with EXPRAG. Our central question is whether retrieval should be used only at inference time or also during fine-tuning on retrieval-augmented data. Recall that our aim to improve performance on unseen tasks within the same environment without degrading seen-task performance.

EXPRAG type	Top- K	Index	ALFWorld			ScienceWorld		
			Easy tasks	Hard tasks	All tasks	Easy tasks	Hard tasks	All tasks
No RAG	0	–	8.22	0.00	4.48	14.41	5.08	10.40
static	1	all	43.84	33.88	39.30	33.33	17.71	26.62
		easy	42.93	5.47	25.87	28.83	10.16	20.81
		hard	35.62	33.88	34.83	22.65	17.97	20.64
dynamic	1	all	46.92 $\uparrow$ 3.1	34.02 $\uparrow$ 0.1	41.04 $\uparrow$ 1.7	29.81 $\downarrow$ 3.5	17.71 0.0	24.61 $\downarrow$ 2.0
		easy	44.75 $\uparrow$ 1.8	4.37 $\downarrow$ 1.1	26.37 $\uparrow$ 0.5	32.65 $\uparrow$ 3.8	7.03 $\downarrow$ 3.1	21.65 $\uparrow$ 0.8
		hard	38.82 $\uparrow$ 3.2	39.89 $\uparrow$ 6.0	39.30 $\uparrow$ 4.5	26.18 $\uparrow$ 3.5	19.53 $\uparrow$ 1.6	23.32 $\uparrow$ 2.7
static	2	all	53.42	48.09	50.99	44.31	21.88	34.67
		easy	47.27	11.48	30.97	43.53	12.24	30.09
		hard	39.73	46.99	43.04	24.12	15.36	20.36
dynamic	2	all	57.53 $\uparrow$ 4.1	48.36 $\uparrow$ 0.3	53.36 $\uparrow$ 2.4	43.53 $\downarrow$ 0.8	21.88 0.0	34.23 $\downarrow$ 0.4
		easy	57.53 $\uparrow$ 10.3	6.56 $\downarrow$ 4.9	34.33 $\uparrow$ 3.4	43.73 $\uparrow$ 0.2	9.64 $\downarrow$ 2.6	29.08 $\downarrow$ 1.0
		hard	39.27 $\downarrow$ 0.5	48.09 $\uparrow$ 1.1	43.28 $\uparrow$ 0.2	25.10 $\uparrow$ 1.0	22.92 $\uparrow$ 7.6	24.16 $\uparrow$ 3.8
static	4	all	63.47	65.03	64.18	42.06	22.27	33.56
		easy	63.01	7.11	37.56	20.39	17.71	19.24
		hard	40.19	62.84	50.50	41.57	13.02	29.31
dynamic	4	all	70.55 $\uparrow$ 7.1	55.74 $\downarrow$ 9.3	63.81 $\downarrow$ 0.4	44.71 $\uparrow$ 2.7	22.66 $\uparrow$ 0.4	35.24 $\uparrow$ 1.7
		easy	71.69 $\uparrow$ 8.7	9.29 $\uparrow$ 2.2	43.28 $\uparrow$ 5.7	46.67 $\uparrow$ 26.3	10.94 $\downarrow$ 6.8	31.32 $\uparrow$ 12.1
		hard	42.93 $\uparrow$ 2.7	65.57 $\uparrow$ 2.7	53.24 $\uparrow$ 2.7	17.25 $\downarrow$ 24.3	23.96 $\uparrow$ 10.9	20.14 $\downarrow$ 9.2

Table 2: **Results for EXPRAG inference without training across ALFWorld and ScienceWorld for Ministral 3-8B.** Values are mean success rates over 3 seeds; full mean $\pm$ std for every entry is reported in Appendix D.1. Across entries, the median/max std is 1.58/6.20 points on ALFWorld and 1.83/4.45 on ScienceWorld. The **bold** values mark the best top- k setting for each (EXPRAG type, index) within a given dataset split. For dynamic retrieval rows, the superscript arrows report the per-cell difference against the corresponding static row with the same index and top- k : $\uparrow$ indicates improvement, $\downarrow$ indicates a decline, and black values denote ties.

4.4.1 Experimental Setup

Based on the findings from Section 4.3, we adopt a practical shared EXPRAG setting for the fine-tuning experiments: (i) trajectories stored in JSON format, (ii) static retrieval, and (iii) $K = 2$. In both environments, we fine-tune four backbone models (Ministral 3-8B, Gemma 3-4B, Qwen 2.5-7B, Qwen 2.5-7B-1M) on the train split of *easy* tasks. We then report inference results on the held-out split of both *easy* and *hard* tasks, in order to assess both in-distribution and out-of-distribution performance. We consider two methods:

- Ⓐ **LoRA**: we employ SFT as described in Sec. 4, and (optionally) apply EXPRAG during inference;
- Ⓑ **EXPRAG-LoRA**: retrieval-augmented fine-tuning where the EXPRAG memory block is added to each training context using the inference-time retrieval pipeline (Sec. 3). We hypothesize this encourages model to learn to use retrieved trajectories instead of memorizing training targets, increasing its reliance on retrieved context at inference.

Generalization Dynamics and Epoch Selection.

For LoRA and EXPRAG-LoRA, we observed roll-out success often continued improving after validation loss started increasing, particularly on held-out task groups. We therefore avoid validation-loss-

based early stopping and evaluate longer runs. We report this preliminary findings on LLM-agents training dynamics in Appendix E. Considering this, for our comparisons, we use fixed training budgets of 10 epochs for ALFWorld and 30 for ScienceWorld, chosen to balance performance, training cost, and robustness across setups. We encourage practitioners to consider even longer runs when maximizing out-of-distribution performance is the primary goal.

4.4.2 Retrieval-Augmented Fine-tuning Enables Robust Task Generalization

We report fine-tuning results in Table 3. We assess hard-task gains with paired task-level confidence analyses; the testing protocol and full results are reported in Appendix F.

In-distribution tasks. On easy tasks seen during training, all fine-tuning methods outperform training-free EXPRAG across models on both ALFWorld and ScienceWorld. Retrieval-augmented training (EXPRAG-LoRA) consistently improves performance on ScienceWorld, but sometimes yields a minor decrease on ALFWorld. Note that for in-distribution tasks, EXPRAG-LoRA does not require additional held-out trajectories, since we use the training set as experience index.

Method	ALFWorld		ScienceWorld	
	Easy tasks (in-d)	Hard tasks (oo-d)	Easy tasks (in-d)	Hard tasks (oo-d)
Ministral 3-8B				
EXPRAG	54.8	47.5 [†]	43.5	18.8
LoRA (no EXPRAG)	98.6	34.4	38.8	15.6
LoRA (with EXPRAG)	97.3	67.2 ^{***}	54.1	28.1 [†]
EXPRAG-LoRA	97.3	88.5 ^{***}	58.8	42.2 ^{**}
Gemma 3-4B				
EXPRAG	20.6	4.9	10.6	6.3
LoRA (no EXPRAG)	61.6	1.6	8.2	6.3
LoRA (with EXPRAG)	57.5	3.3	15.3	7.8
EXPRAG-LoRA	86.3	73.8 ^{***}	31.8	6.2
Qwen 2.5-7B				
EXPRAG	81.6	70.5 ^{***}	16.5	6.2
LoRA (no EXPRAG)	86.3	21.3	24.7	7.8
LoRA (with EXPRAG)	89.0	70.5 ^{***}	25.9	23.4 [*]
EXPRAG-LoRA	84.9	90.2 ^{***}	38.8	29.7 ^{**}
Qwen 2.5-7B-1M				
EXPRAG	67.1	54.1 ^{***}	20.0	15.6
LoRA (no EXPRAG)	98.6	23.0	43.5	14.1
LoRA (with EXPRAG)	82.2	68.9 ^{***}	34.1	12.5
EXPRAG-LoRA	97.3	91.8 ^{***}	50.6	29.7 [*]

Table 3: **Task generalization through retrieval-augmented training for different model backbones.** We report success rates of ALFWorld and ScienceWorld after fine-tuning on a subset of tasks (easy tasks). For hard-task columns, bold marks improvements over the corresponding LoRA baseline that are significant under a paired McNemar test ($p < 0.05$); superscripts denote [†] $p < 0.1$, ^{*} $p < 0.05$, ^{**} $p < 0.01$, and ^{***} $p < 0.001$. Full paired bootstrap confidence intervals and McNemar tests are reported in Appendix F. All retrieval settings use matched index.

Out-of-distribution task generalization. On hard tasks, which the agent has never encountered during training, the performance of LoRA fine-tuned models collapses, falling below training-free EXPRAG in most cases. These sharp easy-to-hard drops are consistent with SFT over-specializing the policy to the easy-task training distribution, such that introducing relevant experience only at inference may be insufficient. Applying EXPRAG at inference time on the LoRA-trained model often improves over bare LoRA, but the paired evidence is weaker in some settings. In contrast, EXPRAG-LoRA is reliably better than bare LoRA on ALFWorld hard for all backbones and on ScienceWorld hard for all backbones except Gemma 3-4B.

Upshot. Bringing retrieval augmentation already into training enables the model to natively handle retrieved trajectories in-context and generalize better to new out-of-distribution tasks.

This comes with higher training cost due to the longer context, but preserves in-distribution performance in the same high-success range. These findings are consistent across benchmarks and models.

4.4.3 Robustness to Imperfect Experience and Learning from Failures

Our experiments so far use a matched index of scripted expert trajectories, a controlled but strong assumption. We therefore stress-test whether the gains of fine-tuned EXPRAG-LoRA persist when available experience is sparse, mismatched, absent, or model-generated, and whether agents can reuse their own failed attempts. We evaluate these conditions on the same EXPRAG variants from Sec. 4.4.2.

We use **Matched Expert**, the full hard-task expert index, as our reference condition. We test more realistic variations: **Empty** removes retrieval, **Mismatched Expert** keeps only the easy-task expert index, and **Low-Coverage Expert** contains only five trajectories per hard task type. To test whether expert experience is necessary, **Matched Strong-LLM** replaces it with matched trajectories generated by stronger Qwen2.5-70B, generally performing worse than expert and containing more suboptimal solutions, representing a setting where a stronger teacher seeds experience for a smaller one. Finally, in **Self-Experience**, we go one step further and use the agent’s own first attempt (successful or failed)¹ of No RAG as index for a second attempt, testing whether it can improve from its own experience without parameter updates. We test this condition both alone and together with each non-empty experience source. Results for Ministral 3-8B are shown in Figure 2; full index-construction details and results for other models are provided in Appendix G.

Retrieval-aware fine-tuned model learns to use imperfect experience. EXPRAG-LoRA is generally the strongest variant when experience is imperfect but still useful. It remains robust with **Low-Coverage Expert** and **Matched Strong-LLM** indices, despite the latter containing more suboptimal trajectories than the matched expert bank. On ALFWorld, the same holds for **Self-Experience** and **Mismatched Expert**; only with **Empty** index does it fall on par with LoRA + EXPRAG.

¹For Ministral-3-8B, Self-Experience index contains 0/61 successful trajectories on ALFWorld and 2/64 on ScienceWorld; full index statistics are reported in Appendix G.1.

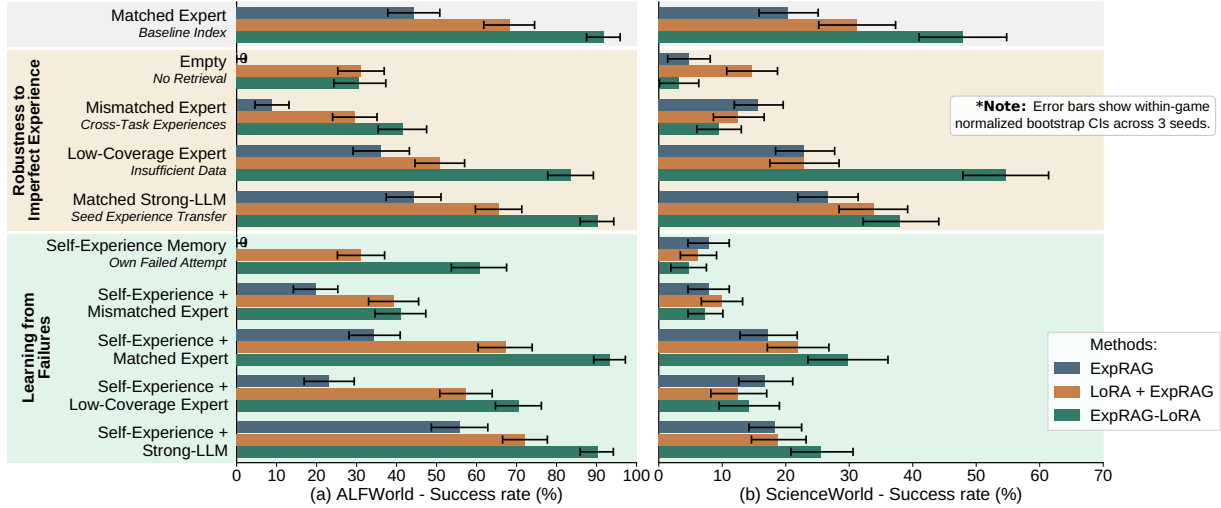


Figure 2: **Experience Robustness and Learning from failures.** Results on unseen hard tasks with Ministral 3-8B. Error bars show 95% within-game normalized bootstrap CIs across three seeds; full statistics are in Appendix G.

Together, these results are consistent with retrieval-aware fine-tuning helping the model identify and combine useful information from imperfect experience, rather than merely imitating retrieved demonstrations: for EXPRAG-LoRA, relevance appears more important than having a large or near-perfect expert bank. The ScienceWorld exceptions may reflect factors observed earlier: a larger easy-to-hard gap with less reusable behaviors, harder retrieval among similar trajectories, and lower-quality trajectories overall.

Towards Self-Improving Agents from Past Experience. EXPRAG-LoRA substantially benefits on ALFWorld when **Self-Experience** is the sole source of experience. This is particularly striking as all first attempts fail, suggesting that retrieval-aware fine-tuning can help the model revisit its mistakes and turn unsuccessful rollouts into useful experience. This ability also persists when the agent’s own experience is combined with **Matched Expert** or **Matched Strong-LLM** memories. The same behavior does not emerge on ScienceWorld, where **Self-Experience** hurts performance, which may reflect the lower diversity of available training experience in ScienceWorld, along with the issues discussed above. Learning from failures therefore also appears to depend on the meta-learning setting, particularly on the diversity of cases seen during training.

Upshot. Retrieval-aware fine-tuning enables agents to use experience beyond large, scripted expert banks: useful memory can be sparse, model-generated, or even come from the agent’s

own attempts. Its effectiveness still depends on the experience containing relevant, reusable information, but these results point toward in-context self-improvement without parameter updates.

5 Conclusion

We studied episodic retrieval for LLM agents, from training-free baselines to retrieval-aware fine-tuning. Across ALFWorld and ScienceWorld, EXPRAG yields large performance gains at inference-time, and our ablations establish a practical trajectory-retrieval recipe. Because standard held-out splits can remain close to the training distribution, we evaluate easy-to-hard task transfer: LoRA often collapses on unseen hard tasks, whereas EXPRAG-LoRA generally provides the strongest hard-task generalization while preserving strong seen-task performance. This advantage often persists with imperfect but useful experience; on ALFWorld, the agent can even reuse a failed attempt to improve a second rollout without test-time parameter updates. Overall, simple read-only episodic retrieval provides a strong foundation for agent memory. We argue that new, more complex memory architectures should be compared against strong fine-tuning and retrieval baselines under task-level distribution shifts and memory-quality stress tests before attributing gains to their added complexity.

Limitations

Our study has some limitations. First, our main experiments use scripted expert trajectories as a

controlled memory source. Section 4.4.3 stress-test this assumption with sparse, mismatched, stronger-LLM-generated, and self-generated experience, but does not study long-term accumulation of heterogeneous experience: Self-Experience reuses only the immediately preceding attempt. Future work should evaluate continually evolving memories over longer interaction horizons. Second, performance remains dependent on the relevance and coverage of available experience: missing or poorly matched trajectories can degrade generalization, and learning from self-generated failures does not emerge consistently across environments. This suggests the need for more principled strategies for experience collection, selection, and retrieval. Finally, our approach uses a fixed, read-only episodic memory. This isolates the policy-side ability to use retrieved experience, but leaves open how these findings extend to read-write memories with consolidation mechanisms (e.g., summarization, abstraction, and selective retention) and longer-horizon adaptation.

Acknowledgements

We thank our colleagues Nadezhda Chirkova and Anastasiia Ivanova for their helpful discussions and valuable feedback on this work and manuscript. Thomas’ research is partially supported by the French Ministry of Higher Education, Research and Innovation under CIFRE PhD Convention “*Reconcile Efficiency and Interpretability in Structured Planning and Reasoning*” (no. 2025/0487).

Use of AI Assistants. AI tools were used during this research for manuscript language editing and polishing, assistance with editing software code, and creating and modifying scientific figures. All AI-generated outputs were reviewed and verified by the authors, who take full responsibility for the scientific content, experimental results, interpretations, and conclusions.

References

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru WANG, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, and 8 others. 2026. [A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence](#). *Transactions on Machine Learning Research*. Survey Certification.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggione, Chris Jones, Albin Cassirer, and 9 others. 2022. [Improving Language Models by Retrieving from Trillions of Tokens](#). In *International Conference on Machine Learning*, pages 2206–2240. PMLR.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. [Fire-Act: Toward Language Agent Fine-tuning](#). *Preprint*, arXiv:2310.05915.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024a. [Agent-FLAN: Designing data and methods of effective agent tuning for large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9354–9366, Bangkok, Thailand. Association for Computational Linguistics.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024b. [Self-play fine-tuning converts weak language models to strong language models](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 6621–6642. PMLR.

Zihao Cheng, Zeming Liu, Yingyu Shan, Xinyi Wang, Xiangrong Zhu, Yunpu Ma, Hongru Wang, Yuhang Guo, Wei Lin, and Yunhong Wang. 2026. [Mem²Evolve: Towards self-evolving agents via co-evolutionary capability expansion and experience distillation](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20784–20831, San Diego, California, United States. Association for Computational Linguistics.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. [Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory](#). In *ECAI 2025 - 28th European Conference on Artificial Intelligence*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pages 2993–3000. IOS Press.

Zhaoye Fei, Li Ji, Siyin Wang, Junhao Shi, Jingjing Gong, and Xipeng Qiu. 2025. [Unleashing Embodied Task Planning Ability in LLMs via Reinforcement Learning](#). *Preprint*, arXiv:2506.23127.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. [Group-in-Group Policy Optimization for LLM Agent Training](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 46375–46408. Curran Associates, Inc.

Thomas Palmeira Ferraz, Kartik Mehta, Yu-Hsiang Lin, Haw-Shiuan Chang, Shereen Oraby, Sijia Liu, Vivek

- Subramanian, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. 2024. [LLM self-correction with De-CRIM: Decompose, critique, and refine for enhanced following of instructions with multiple constraints](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7773–7812, Miami, Florida, USA. Association for Computational Linguistics.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. [Model-agnostic meta-learning for fast adaptation of deep networks](#). In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujia Yang, Nan Duan, and Weizhu Chen. 2024. [CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing](#). In *International Conference on Learning Representations*, volume 2024, pages 57734–57811.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. [The Llama 3 Herd of Models](#). Preprint, arXiv:2407.21783.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Papat, and Mingwei Chang. 2020. [Retrieval augmented language model pre-training](#). In *International Conference on Machine Learning*, pages 3929–3938. PMLR.
- Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. [Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey](#). *Transactions on Machine Learning Research*.
- Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2022. [Meta-Learning in Neural Networks: A Survey](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-Rank Adaptation of Large Language Models](#). In *International Conference on Learning Representations*.
- Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, and 28 others. 2025. [Memory in the age of ai agents](#). *arXiv preprint arXiv:2512.13564*.
- Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. [Atlas: Few-shot learning with retrieval augmented language models](#). *Journal of Machine Learning Research*, 24(251):1–43.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. [SWE-bench: Can Language Models Resolve Real-world Github Issues?](#) In *The Twelfth International Conference on Learning Representations*.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Serkan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. [Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning](#). In *Second Conference on Language Modeling*.
- Tomoyuki Kagaya, Thong Jing Yuan, Yuxuan Lou, Jayashree Karlekar, Sugiri Pranata, Akira Kinose, Koki Oguri, Felix Wick, and Yang You. 2024. [Rap: Retrieval-augmented planning with contextual memory for multimodal llm agents](#). Preprint, arXiv:2402.03610.
- Taeil Kim, Kangsan Kim, and Sung Ju Hwang. 2026. [Agent memory distillation: Empowering small llm agents with hierarchical teacher memory](#). Preprint, arXiv:2608.07169.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.
- Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, and 1 others. 2026a. [Ministral 3](#). Preprint, arXiv:2601.08584.
- Youwei Liu, Jian Wang, Hanlin Wang, Beichen Guo, and Wenjie Li. 2026b. [Imagine-then-Plan: Agent Learning from Adaptive Lookahead with World Models](#). Preprint, arXiv:2601.08955.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. [Self-Refine: Iterative Refinement with Self-Feedback](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.
- Meta AI. 2024. [Llama 3.2: Revolutionizing edge AI and vision with open, customizable models](#). Accessed: 2026-02-16.
- Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. [Memgpt: Towards llms as operating systems](#). *CoRR*, abs/2310.08560.

- Joon Sung Park, Joseph O’Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. [Generative Agents: Interactive Simulacra of Human Behavior](#). In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST ’23*, New York, NY, USA. Association for Computing Machinery.
- Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. [Gorilla: Large Language Model Connected with Massive APIs](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 126544–126565. Curran Associates, Inc.
- Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Jiang, Chengfei Lv, and Huajun Chen. 2024. [AutoAct: Automatic agent learning from scratch for QA via self-planning](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3003–3021, Bangkok, Thailand. Association for Computational Linguistics.
- Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. [In-Context Retrieval-Augmented Language Models](#). *Transactions of the Association for Computational Linguistics*, 11:1316–1331.
- Nils Reimers and Iryna Gurevych. 2019. [Sentence-BERT: Sentence embeddings using Siamese BERT-networks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. 2025. [Continual learning of large language models: A comprehensive survey](#). *ACM Comput. Surv.*, 58(5).
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. [Reflexion: Language Agents with Verbal Reinforcement Learning](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.
- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. [ALFWorld: Aligning Text and Embodied Environments for Interactive Learning](#). In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. [Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG](#). *arXiv preprint arXiv:2501.09136*.
- Jaehyeon Son, Soochan Lee, and Gunhee Kim. 2025. [When meta-learning meets online and continual learning: A survey](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(1):413–432.
- Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. [Trial and Error: Exploration-Based Trajectory Optimization of LLM Agents](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600, Bangkok, Thailand. Association for Computational Linguistics.
- Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. 2026. [Dynamic cheat-sheet: Test-time learning with adaptive memory](#). In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7080–7106, Rabat, Morocco. Association for Computational Linguistics.
- Guokai Tang and Feng Zhao. 2026. [SmartAD: Capacity-aligned agent distillation for small language models](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 27045–27057, San Diego, California, United States. Association for Computational Linguistics.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, and 1 others. 2025. [Gemma 3 Technical Report](#). *Preprint*, arXiv:2503.19786.
- torch tune maintainers and contributors. 2024. [torch tune: Pytorch’s finetuning library](#).
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. [Voyager: An Open-Ended Embodied Agent with Large Language Models](#). *Transactions on Machine Learning Research*.
- Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Herbert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharmashan Kumaran, and Matt Botvinick. 2016. [Learning to reinforcement learn](#). *arXiv preprint arXiv:1611.05763*.
- Renxi Wang, Xudong Han, Yixuan Zhang, Timothy Baldwin, and Haonan Li. 2025a. [NAT: Enhancing Agent Tuning with Negative Samples](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7385–7398, Albuquerque, New Mexico. Association for Computational Linguistics.

- Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. 2024b. [Learning From Failure: Integrating Negative Examples when Fine-tuning Large Language Models as Agents](#). *Preprint*, arXiv:2402.11651.
- Ruiyi Wang and Prithviraj Ammanabrolu. 2025. [A practitioner’s guide to multi-turn agentic reinforcement learning](#). *Preprint*, arXiv:2510.01132.
- Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. [ScienceWorld: Is your agent smarter than a 5th grader?](#) In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gotlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. 2025b. [RAGEN: Understanding Self-Evolution in LLM Agents via Multi-Turn Reinforcement Learning](#). *Preprint*, arXiv:2504.20073.
- Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. 2025. [Evo-Memory: Benchmarking LLM Agent Test-time Learning with Self-Evolving Memory](#). *arXiv preprint arXiv:2511.20857*.
- 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, and 3 others. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025. [From Human Memory to AI Memory: A Survey on Memory Mechanisms in the Era of LLMs](#). *arXiv preprint arXiv:2504.15965*.
- Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. 2026. [SkillIRL: Evolving Agents via Recursive Skill-Augmented Reinforcement Learning](#). *Preprint*, arXiv:2602.08234.
- Yu Xia, Yiran Jenny Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A. Rossi, Lina Yao, and Julian McAuley. 2025. [SAND: Boosting LLM Agents with Self-Taught Action Deliberation](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3062–3077, Suzhou, China. Association for Computational Linguistics.
- Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. [A-Mem: Agentic Memory for LLM Agents](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 17577–17604. Curran Associates, Inc.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024a. [Qwen2.5 Technical Report](#). *Preprint*, arXiv:2412.15115.
- An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, and 9 others. 2025. [Qwen2.5-1M Technical Report](#). *Preprint*, arXiv:2501.15383.
- John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024b. [SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 50528–50652. Curran Associates, Inc.
- Shiping Yang, Jie Wu, Wenbiao Ding, Ning Wu, Shining Liang, Ming Gong, Hongzhi Li, Hengyuan Zhang, Angel X. Chang, and Dongmei Zhang. 2026. [Quantifying and Improving the Robustness of Retrieval-Augmented Language Models Against Spurious Features in Grounding Data](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33479–33499, San Diego, California, United States. Association for Computational Linguistics.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. [ReAct: Synergizing Reasoning and Acting in Language Models](#). In *The Eleventh International Conference on Learning Representations*.
- Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. 2026. [Agentic Memory: Learning Unified Long-Term and Short-Term Memory Management for Large Language Model Agents](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21457–21483, San Diego, California, United States. Association for Computational Linguistics.
- Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. 2025a. [G-Memory:](#)

- [Tracing Hierarchical Memory for Multi-Agent Systems](#). In [Advances in Neural Information Processing Systems](#), volume 38, Main Conference, pages 12988–13018. Curran Associates, Inc.
- Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. 2026a. [Darwin gödel machine: Open-ended evolution of self-improving agents](#). In [The Fourteenth International Conference on Learning Representations](#).
- Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. 2026b. [Hyperagents](#). [Preprint](#), arXiv:2603.19461.
- Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, and 11 others. 2026c. [Agent Learning via Early Experience](#). In [Forty-third International Conference on Machine Learning](#).
- Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y Zou, and Kunle Olukotun. 2026d. [Agentic context engineering: Evolving contexts for self-improving language models](#). In [International Conference on Learning Representations](#), volume 2026, pages 86069–86100.
- Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. 2024. [RAFT: Adapting Language Model to Domain Specific RAG](#). In [First Conference on Language Modeling](#).
- Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. [Qwen3 embedding: Advancing text embedding and reranking through foundation models](#). [arXiv preprint arXiv:2506.05176](#).
- Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025c. [A Survey on the Memory Mechanism of Large Language Model-based Agents](#). [ACM Trans. Inf. Syst.](#), 43(6).
- Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. [Expel: LLM Agents Are Experiential Learners](#). In [Proceedings of the AAAI Conference on Artificial Intelligence](#), volume 38, pages 19632–19642.
- Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and Qianli Ma. 2026. [Lifelong learning of large language model based agents: A roadmap](#). [IEEE Transactions on Pattern Analysis and Machine Intelligence](#), 48(5):5552–5571.
- Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. [MemoryBank: Enhancing Large Language Models with Long-Term Memory](#). [Proceedings of the AAAI Conference on Artificial Intelligence](#), 38:19724–19731.
- Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. 2025. [Memento: Fine-tuning LLM Agents without Fine-tuning LLMs](#). [arXiv preprint arXiv:2508.16153](#).
- Qiming Zhu, Shunian Chen, Rui Yu, Zhehao Wu, and Benyou Wang. 2026. [From Lossy to Verified: A Provenance-Aware Tiered Memory for Agents](#). [Preprint](#), arXiv:2602.17913.

A Reproducibility Statement

To facilitate full reproducibility, we provide detailed documentation of our experimental setup and release the code and trained model checkpoints associated with this work. We release the source code required to reproduce our training, retrieval, and evaluation pipelines at github.com/naver/RAGents. We additionally release the trained model checkpoints at huggingface.co/collections/naver/ragents, including the checkpoints for each model trained on each dataset considered in our experiments. Dataset construction, splits, and statistics are described in Section 4.1 and Appendix C.2 (Tables 4 and 5). Experimental and implementation details are provided in Sections 4.2 and 4.4.1 and Appendix C, including retrieval and indexing settings (Appendix C.1), fine-tuning implementation and compute setup (Appendix C.4), exact model checkpoints (Appendix C.5), and hyperparameters (Appendix C.6; Tables 6 and 7). Exact prompts are provided in Appendix H. Variability and statistical reliability are reported in Appendix D.1 and Appendix F (Tables 18 and 19), including results across multiple seeds, paired bootstrap confidence intervals, and McNemar tests. Together, these resources provide a reproducible foundation for verifying our results and extending memory-augmented and retrieval-augmented training for LLM agents.

B Additional Related Work

Section 2 covers the work most directly related to our experiments: agent fine-tuning, episodic memory, and retrieval of prior trajectories. Here, we connect our setting to the broader literature on systems that learn from interaction. Meta-learning, continual learning, self-correction, memory-based adaptation, and self-improvement have often been studied separately, but these distinctions become less clear in LLM agents. In LLM agents, experience can affect the system in several ways: it can be reused at inference time, incorporated into training, or used to update memory or other components of the agent. We therefore organize this literature according to which part of the system is adapted.

Meta-learning and continual adaptation. Meta-learning asks how a model can be trained to adapt quickly from a small amount of new experience, either through a learned initialization (Finn et al., 2017) or by learning the adaptation procedure itself (Wang et al., 2016). Online and continual learning look at a different setting, where experience arrives sequentially. Continual learning is particularly concerned with preserving earlier capabilities as the data distribution changes (Shi et al., 2025). The two perspectives can also be combined: a system may adapt online while learning, over time, how to adapt more effectively (Son et al., 2025). For LLM agents, adaptation does not necessarily require parameter updates at all. It can instead happen through memory, retrieved context, or interaction with the environment (Zheng et al., 2026; ang Gao et al., 2026). We consider a more restricted case: training is retrieval-aware, but at test time the model adapts to a new task only by conditioning on retrieved context.

From self-correction to evolving memory. Another line of work asks how an agent can learn from its own previous outputs. Self-correction methods iteratively critique and revise the solution at hand, but typically do not retain the resulting lessons across inputs (Madaan et al., 2023; Ferraz et al., 2024; Gou et al., 2024). Some methods make these lessons persistent by storing reflections, feedback, or distilled insights for use in later attempts (Zhao et al., 2024; Shinn et al., 2023). More recent systems go further and treat persistence as part of the learning process itself: Dynamic Cheatsheet (Suzgun et al., 2026) accumulates reusable strategies at test time, ACE (Zhang et al., 2026d) treats context as an evolving playbook, and Evo-Memory (Wei et al., 2025) studies memories that continue to change as the agent encounters new tasks. Self-correction, episodic memory, and test-time learning can therefore be combined: feedback need not only correct the current solution, but can also be retained as experience and reused on later tasks.

Experience quality, provenance, and transfer. Experience can differ not only in relevance, but also in outcome, source, and compatibility with the learner. As discussed in Section 2, methods such as ETO, NAT, and Early Experience use failed or suboptimal interactions as supervision for subsequent policy optimization (Song et al., 2024; Wang et al., 2025a; Zhang et al., 2026c). A related teacher-student line trains smaller agents on trajectories produced by stronger models (Chen et al., 2023; Tang and Zhao,

2026), while Kim et al. (2026) instead converts successful teacher trajectories into structured external memory. By contrast, our stress tests keep the trained policy and inference pipeline fixed and examine whether a retrieval-trained agent can directly use raw trajectories from sparse expert banks, stronger teachers, or its own previous attempts, without additional test-time optimization or memory synthesis.

Toward self-evolving agents. Broader self-improvement methods differ in what they allow to change as experience accumulates. At the model level, SPIN iteratively updates an LLM using responses generated by previous versions of itself (Chen et al., 2024b). Agent-oriented systems extend this idea to policies and memory: RAGEN (Wang et al., 2025b) studies self-evolution through multi-turn reinforcement learning, while SkillRL (Xia et al., 2026) jointly evolves a policy and a retrieved skill library. Mem²Evolve (Cheng et al., 2026) broadens this further to reusable tools and expert assets, using accumulated experience to guide capability expansion while newly acquired capabilities generate new experience in turn. At the system level, the Darwin Godel Machine iteratively modifies and evaluates its own agent harness code, so that the mechanisms supporting future improvement can themselves change (Zhang et al., 2026a,b). Together, these approaches span a progression from adapting behavior through context, to evolving memory and skills, and ultimately to modifying model parameters or the agent architecture itself. Our work focuses on an earlier prerequisite in this progression: training a policy to reliably interpret and reuse raw episodic experience.

B.1 Terminology clarification: Retrieval-Augmented Agents vs. Agentic RAG

The similarity in terminology can obscure an important conceptual distinction. Our setting is not an instance of *Agentic RAG*. Following Hu et al. (2025), we distinguish *retrieval-augmented agents*, which retrieve *experience* to improve decisions in an environment, from *Agentic RAG* (Singh et al., 2025), where an agent orchestrates multi-step retrieval over external knowledge sources. Our work belongs to the former: it studies policy generalization through episodic context reuse, whereas Agentic RAG primarily targets knowledge retrieval and synthesis.

C Implementation details

This appendix centralizes the implementation choices that are shared across experiments.

C.1 Indexing and Retrieval experimental settings

In all conditions with retrieval ($K > 0$), we build a fixed retrieval index from expert trajectories in the environment training split (including failures unless stated otherwise). We embed item queries and index keys (in formatted text) using Qwen/Qwen3-Embedding-0.6B model² (Zhang et al., 2025b) and retrieve the top- K nearest trajectories by dot-product similarity in embedding space (Section 3). We do not experiment with embedding models other than this one, as we found the retrieved trajectories with this model to be satisfactory, given the simplicity of the retrieval task in this scenario, and therefore keep it fixed throughout the paper. We use the *Sentence Transformers* library³ (Reimers and Gurevych, 2019) to compute the embeddings.

Trajectories are stored as raw chat-formatted data. Unless otherwise stated, when retrieved trajectories are inserted into the policy prompt, we format them as *chat JSON*; this is the default used in the main-paper zero-shot and fine-tuning results. Appendix D.4 compares this default against *agentic JSON*, *compact JSON*, and *textual* alternatives.

C.2 Benchmark data statistics

In Section 4.1 we introduced our benchmark datasets and how we split them into hard/easy tasks. Here we provide more details on the data statistics. Table 4 reports how we split training and validation datasets into hard/easy tasks, and what tasks belong to each category. To evaluate models faster and avoid task imbalance, we subsample the ScienceWorld test set, and keep only 5 variations per task. We provide the

²Available at: <https://huggingface.co/Qwen/Qwen3-Embedding-0.6B>

³More information: <https://sbert.net/>

list of variations in Table 5. We use this subset in all tables except Table 1, where we compare with related work.

ALFWorld			ScienceWorld	
Split	Samples	Tasks (<i>task-types</i>)	Samples	Tasks
easy	train=1748, test=73	look_at_obj_in_light pick_clean_then_place_in_recep pick_and_place_simple	train=2335, test=1183	find-plant freeze inclined-plane-friction-unnamed-surfaces lifespan-longest-lived lifespan-longest-lived-then-shortest-lived inclined-plane-friction-named-surfaces boil change-the-state-of-matter-of inclined-plane-determine-angle measure-melting-point-known-substance measure-melting-point-unknown-substance use-thermometer find-non-living-thing melt find-animal lifespan-shortest-lived find-living-thing
hard	train=1805, test=61	pick_cool_then_place_in_recep pick_heat_then_place_in_recep pick_two_obj_and_place	train=1254, test=636	chemistry-mix-paint-secondary-color test-conductivity power-component-renewable-vs-nonrenewable-energy chemistry-mix-paint-tertiary-color identify-life-stages-1 identify-life-stages-2 test-conductivity-of-unknown-substances grow-fruit mendelian-genetics-known-plant power-component grow-plant mendelian-genetics-unknown-plant chemistry-mix

Table 4: Details about the hard/easy task splits: number of training and test samples, and task names assigned to each category. Samples from *training tasks* are used to build the index for the corresponding split (hard/easy), while easy training tasks are used for EXPRAG fine-tuning.

C.3 Memory block formatting

When inserting retrieved trajectories into the agent context, we prepend a memory block that separates successful from unsuccessful demonstrations. Concretely, we use the following template:

These are examples of successful trajectories: { . . . }. These are examples of unsuccessful trajectories: { . . . }.

Within this memory block, the retrieved trajectories themselves are represented in *chat JSON* format by default (Figure 3), unless a formatting ablation explicitly states otherwise.

C.4 Fine-tuning implementation

We implement supervised LoRA fine-tuning with TorchTune (TorchTune, 2024). Training trajectories are formatted as multi-turn chats using each backbone’s default chat template, and we compute the cross-entropy loss on assistant tokens only. We train all models on a single 80 GB NVIDIA A100 GPU.

Despite using TorchTune for training, we run inference with Hugging Face Transformers (Wolf et al., 2020), which we found to be more efficient.

Task	Task variations
boil	29, 27, 25, 23, 21
change-the-state-of-matter-of	29, 23, 28, 27, 26
chemistry-mix	26, 31, 24, 29, 25
chemistry-mix-paint-secondary-color	34, 27, 31, 33, 30
chemistry-mix-paint-tertiary-color	30, 33, 35, 28, 29
find-animal	248, 260, 294, 271, 299
find-living-thing	257, 256, 277, 266, 268
find-non-living-thing	280, 261, 233, 299, 253
find-plant	251, 231, 227, 254, 236
freeze	26, 23, 28, 21, 29
grow-fruit	123, 114, 108, 105, 96
grow-plant	95, 94, 100, 112, 116
identify-life-stages-1	12, 9, 11, 13, 10
identify-life-stages-2	6, 7, 8, 9
inclined-plane-determine-angle	151, 147, 150, 152, 148
inclined-plane-friction-named-surfaces	1356, 1327, 1322, 1366, 1360
inclined-plane-friction-unnamed-surfaces	146, 158, 145, 157, 156
lifespan-longest-lived	112, 106, 107, 111, 93
lifespan-longest-lived-then-shortest-lived	110, 114, 99, 101, 115
lifespan-shortest-lived	124, 94, 95, 121, 112
measure-melting-point-known-substance	355, 400, 422, 413, 403
measure-melting-point-unknown-substance	266, 263, 243, 281, 265
melt	27, 28, 29, 26, 23
mendelian-genetics-known-plant	94, 105, 100, 93, 91
mendelian-genetics-unknown-plant	413, 429, 406, 467, 442
power-component	19, 17, 18, 15, 16
power-component-renewable-vs-nonrenewable-energy	19, 16, 17, 15, 18
test-conductivity	858, 726, 839, 878, 844
test-conductivity-of-unknown-substances	538, 475, 494, 464, 532
use-thermometer	534, 476, 450, 473, 510

Table 5: Task variations used in our subsampled ScienceWorld test set.

C.5 Model Backbones

Models. In all experiments, we use only instruction-tuned base LLMs as agent policies. In the main text, we refer to each model by a short name. The corresponding official checkpoint names on the Hugging Face Hub⁴ are:

- **Minstral 3-8B** (Liu et al., 2026a): mistralai/Minstral-3-8B-Instruct-2512-BF16
- **Gemma 3-4B** (Team et al., 2025): google/gemma-3-4b-it
- **Qwen 2.5-7B-1M** (Yang et al., 2025): Qwen/Qwen2.5-7B-Instruct-1M
- **Qwen 2.5-7B** (Yang et al., 2024a): Qwen/Qwen2.5-7B-Instruct
- **Qwen 2.5-3B** (Yang et al., 2024a): Qwen/Qwen2.5-3B-Instruct
- **Llama 3.1-8B** (Grattafiori et al., 2024): meta-llama/Llama-3.1-8B-Instruct
- **Llama 3.2-3B** (Meta AI, 2024): meta-llama/Llama-3.2-3B-Instruct
- **Llama 3.2-1B** (Meta AI, 2024): meta-llama/Llama-3.2-1B-Instruct

We include both Qwen 2.5-7B and Qwen 2.5-7B-1M at matched parameter scales. Qwen2.5-7B-1M is not merely a longer-context setting of Qwen 2.5-7B: it is obtained via additional long-context adaptation (continued training with progressive context-length expansion and associated positional configuration changes), and is post-trained for long-input instruction following (Yang et al., 2025). Consequently, comparisons between Qwen 2.5-7B and Qwen 2.5-7B-1M reflect both increased context capacity and the effects of long-context-specific training and alignment.

⁴Base URL: https://huggingface.co/<model_name> ; replace <model_name> with the checkpoint official identifier to obtain the model page.

C.6 Hyperparameters

Table 6 reports the hyperparameters used for our method. The values below are the defaults used in all experiments unless stated otherwise in the main text or in the experimental setup; when a setting is specified elsewhere (e.g., for a particular benchmark or ablation), that specification overrides the corresponding entry in this table.

Hyperparameters	
optimizer	PagedAdamW8bit
learning rate	5e-5
weight decay	0.0
lr scheduler	constant
LoRA target modules	q_proj, v_proj, k_proj, output_proj
LoRA rank	8
LoRA α	16
LoRA dropout	0.1
dtype	bf16
decoding temperature	0.0 (greedy)
seed	2025

Table 6: All the hyperparameters used for our method. Values are shared across models unless specified otherwise in the text.

In addition to the shared optimization and decoding settings above, we set the maximum number of actions that the agent is allowed to perform, using `max_steps_per_task`. For ALFWorld, we follow the environment default (`max_steps_per_task = 50`). For ScienceWorld, we found that some tasks are not solvable within 50 steps. On the other hand, setting a uniformly high budget would make evaluation inefficient for easy tasks that the model cannot solve, where the agent may get stuck in loops. Based on the number of steps taken by the rule-based expert to solve each task for each task group, we therefore define a different `max_steps_per_task` value for each ScienceWorld task category. We report these values in Table 7. If the model fails to solve the task within the budget, we consider the episode as a failure.

Max Steps	ScienceWorld tasks
150	mendelian-genetics-known-plant; mendelian-genetics-unknown-plant
120	boil
90	freeze
80	grow-fruit
70	change-the-state-of-matter-of; inclined-plane-determine-angle; inclined-plane-friction-unnamed-surfaces; melt
50	chemistry-mix; chemistry-mix-paint-secondary-color; chemistry-mix-paint-tertiary-color; find-animal; find-living-thing; find-non-living-thing; find-plant; grow-plant; identify-life-stages-1; identify-life-stages-2; inclined-plane-friction-named-surfaces; lifespan-longest-lived; lifespan-longest-lived-then-shortest-lived; lifespan-shortest-lived; measure-melting-point-known-substance; measure-melting-point-unknown-substance; power-component; power-component-renewable-vs-nonrenewable-energy; test-conductivity; test-conductivity-of-unknown-substances; use-thermometer

Table 7: ScienceWorld task-dependent rollout budgets used in evaluation (`max_steps_per_task`).

EXPRAG type	Top- K	Index	ALFWorld			ScienceWorld		
			Easy tasks	Hard tasks	All tasks	Easy tasks	Hard tasks	All tasks
No RAG	0	–	8.22 $\pm$ 2.37	0.00 $\pm$ 0.00	4.48 $\pm$ 1.29	14.41 $\pm$ 3.24	5.08 $\pm$ 1.50	10.40 $\pm$ 1.78
static	1	all	43.84 $\pm$ 4.94	33.88 $\pm$ 6.20	39.30 $\pm$ 5.40	33.33 $\pm$ 1.80	17.71 $\pm$ 2.39	26.62 $\pm$ 2.05
		easy	42.93 $\pm$ 0.79	5.47 $\pm$ 0.95	25.87 $\pm$ 0.87	28.83 $\pm$ 0.68	10.16 $\pm$ 3.25	20.81 $\pm$ 1.45
		hard	35.62 $\pm$ 4.94	33.88 $\pm$ 4.73	34.83 $\pm$ 0.86	22.65 $\pm$ 1.48	17.97 $\pm$ 2.02	20.64 $\pm$ 1.27
dynamic	1	all	46.92 $\pm$ 0.69	34.02 $\pm$ 0.82	41.04 $\pm$ 0.61	29.81 $\pm$ 2.71	17.71 $\pm$ 2.39	24.61 $\pm$ 1.69
		easy	44.75 $\pm$ 3.45	4.37 $\pm$ 0.95	26.37 $\pm$ 2.28	32.65 $\pm$ 1.13	7.03 $\pm$ 0.90	21.65 $\pm$ 0.64
		hard	38.82 $\pm$ 5.19	39.89 $\pm$ 0.95	39.30 $\pm$ 3.11	26.18 $\pm$ 2.01	19.53 $\pm$ 3.72	23.32 $\pm$ 2.00
static	2	all	53.42 $\pm$ 2.37	48.09 $\pm$ 3.79	50.99 $\pm$ 1.56	44.31 $\pm$ 3.40	21.88 $\pm$ 3.13	34.67 $\pm$ 2.36
		easy	47.27 $\pm$ 0.97	11.48 $\pm$ 2.31	30.97 $\pm$ 1.58	43.53 $\pm$ 1.29	12.24 $\pm$ 2.30	30.09 $\pm$ 1.43
		hard	39.73 $\pm$ 1.37	46.99 $\pm$ 1.89	43.04 $\pm$ 1.56	24.12 $\pm$ 1.93	15.36 $\pm$ 1.18	20.36 $\pm$ 1.39
dynamic	2	all	57.53 $\pm$ 1.94	48.36 $\pm$ 0.95	53.36 $\pm$ 0.96	43.53 $\pm$ 0.00	21.88 $\pm$ 2.21	34.23 $\pm$ 0.95
		easy	57.53 $\pm$ 3.87	6.56 $\pm$ 0.00	34.33 $\pm$ 2.11	43.73 $\pm$ 2.03	9.64 $\pm$ 2.08	29.08 $\pm$ 1.83
		hard	39.27 $\pm$ 2.09	48.09 $\pm$ 1.89	43.28 $\pm$ 1.98	25.10 $\pm$ 2.31	22.92 $\pm$ 3.07	24.16 $\pm$ 2.36
static	4	all	63.47 $\pm$ 1.58	65.03 $\pm$ 5.75	64.18 $\pm$ 1.97	42.06 $\pm$ 2.61	22.27 $\pm$ 1.50	33.56 $\pm$ 2.12
		easy	63.01 $\pm$ 1.37	7.11 $\pm$ 0.95	37.56 $\pm$ 0.86	20.39 $\pm$ 1.80	17.71 $\pm$ 0.90	19.24 $\pm$ 1.40
		hard	40.19 $\pm$ 0.79	62.84 $\pm$ 3.41	50.50 $\pm$ 1.88	41.57 $\pm$ 2.45	13.02 $\pm$ 3.93	29.31 $\pm$ 0.39
dynamic	4	all	70.55 $\pm$ 3.26	55.74 $\pm$ 1.89	63.81 $\pm$ 1.29	44.71 $\pm$ 1.66	22.66 $\pm$ 2.70	35.24 $\pm$ 1.69
		easy	71.69 $\pm$ 0.79	9.29 $\pm$ 0.95	43.28 $\pm$ 0.75	46.67 $\pm$ 4.45	10.94 $\pm$ 1.56	31.32 $\pm$ 2.71
		hard	42.93 $\pm$ 1.58	65.57 $\pm$ 1.64	53.24 $\pm$ 0.43	17.25 $\pm$ 1.80	23.96 $\pm$ 0.90	20.14 $\pm$ 1.17

Table 8: **Mean and standard deviation of success-rates for EXPRAG inference without training across ALFWorld and ScienceWorld for Ministral 3-8B.** Values are reported as mean $\pm$ std over 3 seeds. Across entries, the median/max std is 1.58/6.20 points on ALFWorld and 1.83/4.45 on ScienceWorld. The **bold** mean values mark the best top- k setting for each (EXPRAG type, index) within a given dataset split.

D Extra experiments on EXPRAG without Training

D.1 Main Results Stability and Reliability Across Seeds

Tables 8 and 9 show the variability of the main results for Ministral 3-8B across three seeds (2025, 2026, 2027). These are the same results as in Table 2, with the standard deviations reported. Although decoding is greedy, the seed still affects mainly: (i) sampled environment instance (e.g., object placement), (ii) retrieval tie-breaking⁵, and therefore (iii) the subsequent interaction history seen by dynamic retrieval. Overall, the results are reasonably stable: across all configurations, the median/max standard deviation is 1.58/6.20 points on ALFWorld and 1.83/4.45 on ScienceWorld. We do not observe a single universally unstable regime. Instead, variability concentrates in harder or more retrieval-sensitive settings, while all-task averages are more stable (median std 1.56 on ALFWorld and 1.69 on ScienceWorld). Dynamic retrieval is only slightly less stable than static retrieval on ScienceWorld (median/max 2.00/4.45 vs. 1.80/3.93), whereas on ALFWorld both have the same median variability (1.58). Importantly, the main best-performing configurations from Section 4.3 are not unusually unstable: static retrieval with the all index at $K=4$ on ALFWorld reaches 64.18 ± 1.97 , and dynamic retrieval with the all index at $K=4$ on ScienceWorld reaches 35.24 ± 1.69 . Thus, the seed analysis supports the reliability of the main conclusions: the qualitative ranking of retrieval settings is robust, and the larger deviations mostly appear in specialized split/index combinations rather than in the central all-task findings.

D.2 Validating Main Results across Different Models

Tables 15, 16, 10, 11, 12, and 13 report inference-only EXPRAG results for additional backbones. Overall, they support the same main conclusion as Table 2 for Ministral 3-8B: retrieval is already highly useful without any training, although the magnitude of the gain depends strongly on the backbone.

Retrieval consistently improves over zero-shot. Retrieval consistently improves over NO RAG at the best setting for every model. In Section 4.3, Ministral 3-8B improves from 4.48/10.40 to 64.18/35.24

⁵Ties happen when multiple trajectories are retrieved with same score, which is recurrent, particularly in the static setting with task description as query.

Grouping	ALFWorld		ScienceWorld	
	Median std	Max std	Median std	Max std
No RAG	1.29	2.37	1.78	3.24
static	1.58	6.20	1.80	3.93
dynamic	1.58	5.19	2.00	4.45
Top- k = 0	1.29	2.37	1.78	3.24
Top- k = 1	1.62	6.20	1.90	3.72
Top- k = 2	1.89	3.87	2.06	3.40
Top- k = 4	1.48	5.75	1.75	4.45

(a) **Grouped by retrieval setting.** The first block aggregates by retrieval type; the second aggregates by top- k .

Split	ALFWorld		ScienceWorld	
	Median std	Max std	Median std	Max std
Easy Tasks	1.94	5.19	1.93	4.45
Hard Tasks	1.64	6.20	2.21	3.93
All Tasks	1.56	5.40	1.69	2.71

(b) **Grouped by evaluation split.** Each row pools all configurations evaluated on that split.

Index	ALFWorld			ScienceWorld		
	Easy Tasks	Hard Tasks	All Tasks	Easy Tasks	Hard Tasks	All Tasks
all	2.16 / 4.94	2.84 / 6.20	1.43 / 5.40	2.21 / 3.40	2.39 / 3.13	1.87 / 2.36
easy	1.17 / 3.87	0.95 / 2.31	1.23 / 2.28	1.54 / 4.45	1.82 / 3.25	1.44 / 2.71
hard	1.83 / 5.19	1.89 / 4.73	1.72 / 3.11	1.97 / 2.45	2.54 / 3.93	1.33 / 2.36

(c) **Grouped by retrieval index.** Each cell reports median/max std over all configurations with that index and benchmark split.

Table 9: **Summary statistics for standard deviations in zero-shot Ministral 3-8B results.** The top row summarizes variability by retrieval setting and by evaluation split. The bottom table breaks the same standard deviations down jointly by retrieval index and benchmark split.

ALFWorld/ScienceWorld all-task success. In the models reported here in this appendix, the best retrieval setting improves from 7.46/7.38 to 47.76/27.52 for Llama 3.1 8B, from 0.75/2.01 to 12.69/19.46 for Llama 3.2 3B, from 0.75/2.01 to 20.90/11.41 for Gemma 3 4B, from 5.22/2.68 to 25.37/8.05 for Qwen 2.5 3B, from 29.85/2.01 to 83.58/12.75 for Qwen 2.5 7B, and from 5.22/3.36 to 81.34/27.52 for Qwen 2.5 7B 1M. This matches the main-paper pattern: retrieved trajectories provide strong training-free gains from a frozen policy alone. Paired confidence intervals and McNemar tests for the hard-task comparisons against No RAG are reported in Table 18 in Appendix F.1.

Larger top- k is useful, but more backbone-dependent. Larger top- k remains useful overall, but the pattern is more backbone-dependent. In Section 4.3, for Ministral 3-8B, ScienceWorld largely saturates after $K=2$, whereas the appendix backbones usually achieve their best all-task scores at $K=4$, especially on ALFWorld. At the same time, the gains are not monotonic in every split or index setting: smaller backbones such as Gemma 3 4B and Qwen 2.5 3B still show noticeable regressions when additional retrieved trajectories are weakly matched or noisy. This remains consistent with the main-text interpretation that larger memory helps when it adds complementary evidence, but can hurt when it introduces distractors.

Dynamic retrieval is the least stable factor across backbones. Dynamic retrieval can help substantially in selected settings, for example Qwen 2.5 3B with the all index at $K=2$ on ALFWorld all tasks (20.90% $\rightarrow$ 24.63%) or Qwen 2.5 7B 1M with the all index at $K=4$ on ScienceWorld all tasks (23.49% $\rightarrow$ 27.52%). However, it also produces some of the largest drops, such as Qwen 2.5 7B with the all index at $K=2$ on ALFWorld all tasks (82.84% $\rightarrow$ 73.13%). This mirrors the Ministral 3-8B results: re-retrieval can increase relevance, but it is much less predictable because it repeatedly refreshes the prompt with partial-trajectory matches.

Index specialization remains broadly coherent. Easy-index rows tend to help easy splits more, hard-index rows tend to help hard splits more, and the mixed all index is often the strongest overall compromise. The same benchmark asymmetry as in the main paper also appears here: ALFWorld benefits more reliably from stronger retrieval and larger memory, whereas ScienceWorld remains more sensitive to index mismatch and retrieval noise, especially for smaller models.

Model size changes the magnitude of these effects. Among the smaller backbones, Llama 3.2 3B (Table 16), Gemma 3 4B (Table 10), and Qwen 2.5 3B (Table 11) all benefit from retrieval but remain substantially weaker and noisier than Ministral 3-8B, Llama 3.1 8B, and Qwen 2.5 7B. Within the Qwen family, scaling from 3B to 7B sharply improves both baseline competence and retrieval gains, especially on ALFWorld, where the best all-task score rises from 25.37% to 83.58%.

EXPRAg type	Top- K	Index	ALFWorld			ScienceWorld		
			Easy tasks	Hard tasks	All tasks	Easy tasks	Hard tasks	All tasks
No RAG	0	–	1.37	0.00	0.75	0.00	4.69	2.01
static	1	all	20.55	3.28	12.69	4.71	1.56	3.36
		easy	17.81	0.00	9.70	7.06	4.69	6.04
		hard	8.22	3.28	5.97	2.35	0.00	1.34
dynamic	1	all	19.18 ↓ 1.4	3.28 0.0	11.94 ↓ 0.8	7.06 ↑ 2.4	3.12 ↑ 1.6	5.37 ↑ 2.0
		easy	17.81 0.0	0.00 0.0	9.70 0.0	7.06 0.0	1.56 ↓ 3.1	4.70 ↓ 1.3
		hard	9.59 ↑ 1.4	6.56 ↑ 3.3	8.21 ↑ 2.2	4.71 ↑ 2.4	3.12 ↑ 3.1	4.03 ↑ 2.7
static	2	all	16.44	6.56	11.94	6.67	3.12	5.15
		easy	21.92	4.92	14.18	10.59	1.56	6.71
		hard	4.11	4.92	4.48	4.71	4.69	4.70
dynamic	2	all	23.29 ↑ 6.8	4.10 ↓ 2.5	14.55 ↑ 2.6	9.41 ↑ 2.7	3.12 0.0	6.71 ↑ 1.6
		easy	26.03 ↑ 4.1	3.28 ↓ 1.6	15.67 ↑ 1.5	9.41 ↓ 1.2	1.56 0.0	6.04 ↓ 0.7
		hard	9.59 ↑ 5.5	8.20 ↑ 3.3	8.96 ↑ 4.5	3.53 ↓ 1.2	3.12 ↓ 1.6	3.36 ↓ 1.3
static	4	all	21.92	19.67	20.90	8.24	7.81	8.05
		easy	24.66	3.28	14.93	9.41	4.69	7.38
		hard	4.11	6.56	5.22	7.06	3.12	5.37
dynamic	4	all	17.81 ↓ 4.1	6.56 ↓ 13.1	12.69 ↓ 8.2	14.12 ↑ 5.9	7.81 0.0	11.41 ↑ 3.4
		easy	27.40 ↑ 2.7	3.28 0.0	16.42 ↑ 1.5	14.12 ↑ 4.7	3.12 ↓ 1.6	9.40 ↑ 2.0
		hard	9.59 ↑ 5.5	6.56 0.0	8.21 ↑ 3.0	5.88 ↓ 1.2	12.50 ↑ 9.4	8.72 ↑ 3.4

Table 10: **Results for EXPRAg inference without training across ALFWorld and ScienceWorld for Gemma 3 4B.** The **bold** values mark the best top- k setting for each (EXPRAg type, index) within a given dataset split. For dynamic retrieval rows, the superscript arrows report the per-cell difference against the corresponding static row with the same index and top- k : ↑ indicates improvement, ↓ indicates a decline, and black values denote ties.

D.3 Long-context Qwen Supports Larger Top- K Retrieval.

Tables 12, 13, and 14 together provide a more nuanced picture than a simple “longer context is better” summary. The two Qwen variants share similar scale and architecture, but their behavior differs enough that the interpretation depends strongly on the benchmark.

The most striking difference between Qwen 2.5 7B (Table 12) and Qwen 2.5 7B 1M (Table 13) appears already in the NO RAG baseline on ALFWorld: 29.85% all-task success for the standard 7B model versus only 5.22% for the 1M variant, with similarly large gaps on easy and hard tasks. This makes the ALFWorld comparison intrinsically confounded. Given the unusually strong zero-shot performance of Qwen 2.5 7B on this benchmark, we suspect that the standard 7B model may have been exposed during training to ALFWorld tasks or very close variants. We cannot verify this directly, but the two models, despite similar size and architecture, need not share the same pre-training or post-training data. For that reason, we do not treat Qwen 2.5 7B as a main backbone in the paper, and we interpret its ALFWorld advantage with caution.

This caution is visible in Table 14. On ALFWorld, the left half of the table is dominated by red values because the standard 7B model starts from a much stronger prior. Those absolute differences are therefore not directly comparable to the ScienceWorld side. What remains informative on ALFWorld is mostly the direction of the trend: retrieval often reduces the large initial gap, and some cells even flip from red to green when K becomes large, especially in the $K=4$ block. For example, static retrieval with the all index moves from a -24.63 all-task gap in NO RAG to -5.22 at $K=1$, -20.15 at $K=2$, and then -2.24 at $K=4$, while the hard-index $K=4$ rows contain several green entries on easy and all tasks. We therefore

EXPRAG type	Top-K	Index	ALFWorld			ScienceWorld		
			Easy tasks	Hard tasks	All tasks	Easy tasks	Hard tasks	All tasks
No RAG	0	–	5.48	4.92	5.22	1.18	4.69	2.68
static	1	all	20.55	11.48	16.42	5.88	3.12	4.70
		easy	24.66	3.28	14.93	7.06	1.56	4.70
		hard	8.22	18.03	12.69	2.35	3.12	2.68
dynamic		all	24.66 ^{↑ 4.1}	19.67 ^{↑ 8.2}	22.39 ^{↑ 6.0}	–	–	–
		easy	23.29 ^{↓ 1.4}	1.64 ^{↓ 1.6}	13.43 ^{↓ 1.5}	7.06 _{0.0}	0.00 ^{↓ 1.6}	4.03 ^{↓ 0.7}
		hard	9.59 ^{↑ 1.4}	22.95 ^{↑ 4.9}	15.67 ^{↑ 3.0}	2.35 _{0.0}	0.00 ^{↓ 3.1}	1.34 ^{↓ 1.3}
static	2	all	26.03	14.75	20.90	4.71	4.69	4.70
		easy	23.29	1.64	13.43	12.94	0.00	7.38
		hard	15.07	21.31	17.91	3.53	4.69	4.03
dynamic		all	30.14 ^{↑ 4.1}	18.03 ^{↑ 3.3}	24.63 ^{↑ 3.7}	5.88 ^{↑ 1.2}	6.25 ^{↑ 1.6}	6.04 ^{↑ 1.3}
static	4	all	31.51	18.03	25.37	7.06	3.12	5.37
		easy	24.66	3.28	14.93	4.71	0.00	2.68
		hard	10.96	21.31	15.67	3.53	4.69	4.03
dynamic		all	30.14 ^{↓ 1.4}	8.20 ^{↓ 9.8}	20.15 ^{↓ 5.2}	11.76 ^{↑ 4.7}	0.00 ^{↓ 3.1}	6.71 ^{↑ 1.3}
		easy	24.66 _{0.0}	6.56 ^{↑ 3.3}	16.42 ^{↑ 1.5}	14.12 ^{↑ 9.4}	0.00 _{0.0}	8.05 ^{↑ 5.4}
		hard	9.59 ^{↓ 1.4}	14.75 ^{↓ 6.6}	11.94 ^{↓ 3.7}	1.18 ^{↓ 2.4}	6.25 ^{↑ 1.6}	3.36 ^{↓ 0.7}

Table 11: **Results for EXPRAG inference without training across ALFWorld and ScienceWorld for Qwen 2.5 3B.** The **bold** values mark the best top- k setting for each (EXPRAG type, index) within a given dataset split. For dynamic retrieval rows, the superscript arrows report the per-cell difference against the corresponding static row with the same index and top- k : [↑] indicates improvement, [↓] indicates a decline, and black values denote ties.

EXPRAG type	Top-K	Index	ALFWorld			ScienceWorld		
			Easy tasks	Hard tasks	All tasks	Easy tasks	Hard tasks	All tasks
No RAG	0	–	35.62	22.95	29.85	1.18	3.12	2.01
static	1	all	73.97	59.02	67.16	17.65	4.69	12.08
		easy	71.23	31.15	52.99	16.47	3.12	10.74
		hard	64.38	62.30	63.43	7.06	7.81	7.38
dynamic		all	72.60 ^{↓ 1.4}	65.57 ^{↑ 6.6}	69.40 ^{↑ 2.2}	11.76 ^{↓ 5.9}	7.81 ^{↑ 3.1}	10.07 ^{↓ 2.0}
		easy	73.97 ^{↑ 2.7}	34.43 ^{↑ 3.3}	55.97 ^{↑ 3.0}	12.94 ^{↓ 3.5}	3.12 _{0.0}	8.72 ^{↓ 2.0}
		hard	64.38 _{0.0}	62.30 _{0.0}	63.43 _{0.0}	3.53 ^{↓ 3.5}	7.81 _{0.0}	5.37 ^{↓ 2.0}
static	2	all	83.56	81.97	82.84	12.94	10.94	12.08
		easy	71.23	54.10	63.43	16.47	0.00	9.40
		hard	73.97	68.85	71.64	4.71	6.25	5.37
dynamic		all	82.19 ^{↓ 1.4}	62.30 ^{↓ 19.7}	73.13 ^{↓ 9.7}	8.24 ^{↓ 4.7}	7.81 ^{↓ 3.1}	8.05 ^{↓ 4.0}
static	4	all	79.45	88.52	83.58	12.94	9.38	11.41
		easy	78.08	42.62	61.94	9.41	0.00	5.37
		hard	47.95	80.33	62.69	1.18	7.81	4.03
dynamic		all	76.71 ^{↓ 2.7}	77.05 ^{↓ 11.5}	76.87 ^{↓ 6.7}	14.12 ^{↑ 1.2}	10.94 ^{↑ 1.6}	12.75 ^{↑ 1.3}
		easy	87.67 ^{↑ 9.6}	44.26 ^{↑ 1.6}	67.91 ^{↑ 6.0}	12.94 ^{↑ 3.5}	0.00 _{0.0}	7.38 ^{↑ 2.0}
		hard	56.16 ^{↑ 8.2}	67.21 ^{↓ 13.1}	61.19 ^{↓ 1.5}	2.35 ^{↑ 1.2}	9.38 ^{↑ 1.6}	5.37 ^{↑ 1.3}

Table 12: **Results for EXPRAG inference without training across ALFWorld and ScienceWorld for Qwen 2.5 7B.** The **bold** values mark the best top- k setting for each (EXPRAG type, index) within a given dataset split. For dynamic retrieval rows, the superscript arrows report the per-cell difference against the corresponding static row with the same index and top- k : [↑] indicates improvement, [↓] indicates a decline, and black values denote ties.

read the ALFWorld comparison mainly as evidence that retrieval can partially close the initial gap, not as a clean estimate of which model is intrinsically better.

ScienceWorld is more comparable because both models are weak in the pure zero-shot setting (2.01% for Qwen 2.5-7B and 3.36% for Qwen 2.5-7B-1M on all tasks). In that benchmark, Table 14 becomes

EXPRAG type	Top- K	Index	ALFWorld			ScienceWorld		
			Easy tasks	Hard tasks	All tasks	Easy tasks	Hard tasks	All tasks
No RAG	0	–	6.85	3.28	5.22	3.53	3.12	3.36
static	1	all	64.38	59.02	61.94	15.29	6.25	11.41
		easy	56.16	14.75	37.31	17.65	0.00	10.07
		hard	39.73	59.02	48.51	7.06	6.25	6.71
dynamic	1	all	63.01 ↓ 1.4	57.38 ↓ 1.6	60.45 ↓ 1.5	12.94 ↓ 2.4	0.00 ↓ 6.2	7.38 ↓ 4.0
		easy	57.53 ↑ 1.4	16.39 ↑ 1.6	38.81 ↑ 1.5	12.94 ↓ 4.7	0.00 0.0	7.38 ↓ 2.7
		hard	45.21 ↑ 5.5	54.10 ↓ 4.9	49.25 ↑ 0.7	12.94 ↑ 5.9	6.25 0.0	10.07 ↑ 3.4
static	2	all	71.23	52.46	62.69	20.00	14.06	17.45
		easy	67.12	29.51	50.00	20.00	0.00	11.41
		hard	47.95	54.10	50.75	8.24	7.81	8.05
dynamic	2	all	63.01 ↓ 8.2	49.18 ↓ 3.3	56.72 ↓ 6.0	16.47 ↓ 3.5	14.06 0.0	15.44 ↓ 2.0
static	4	all	80.82	81.97	81.34	16.47	32.81	23.49
		easy	69.86	34.43	53.73	17.65	0.00	10.07
		hard	63.01	72.13	67.16	11.76	25.00	17.45
dynamic	4	all	75.34 ↓ 5.5	60.66 ↓ 21.3	68.66 ↓ 12.7	23.53 ↑ 7.1	32.81 0.0	27.52 ↑ 4.0
		easy	79.45 ↑ 9.6	21.31 ↓ 13.1	52.99 ↓ 0.7	9.41 ↓ 8.2	28.12 ↑ 28.1	17.45 ↑ 7.4
		hard	61.64 ↓ 1.4	62.30 ↓ 9.8	61.94 ↓ 5.2	25.88 ↑ 14.1	0.00 ↓ 25.0	14.77 ↓ 2.7

Table 13: **Results for EXPRAG inference without training across ALFWorld and ScienceWorld for Qwen 2.5 7B 1M.** The **bold** values mark the best top- k setting for each (EXPRAG type, index) within a given dataset split. For dynamic retrieval rows, the superscript arrows report the per-cell difference against the corresponding static row with the same index and top- k : ↑ indicates improvement, ↓ indicates a decline, and black values denote ties.

progressively greener as K increases, especially in the lower $K=4$ block. This is consistent with the individual model tables: the bold best cells in Table 13 are concentrated at $K=4$, whereas Table 12 often peaks earlier on ScienceWorld, with several bold entries already at $K=1$ or $K=2$. The all-task gap for the all index illustrates this pattern clearly: under static retrieval it moves from -0.67 at $K=1$ to $+5.37$ at $K=2$ and $+12.08$ at $K=4$; under dynamic retrieval it moves from -2.69 to $+7.39$ and then $+14.77$. The same pattern appears on harder ScienceWorld settings, where many of the largest positive differences are bright green in the comparison table, such as $+23.43$ for static all / hard tasks at $K=4$ and $+28.12$ for dynamic easy / hard tasks at $K=4$.

Long-context support is one important retrieval bottleneck, but not the only one. Taken together, these results are consistent with the view that context-handling capacity is an important bottleneck for scaling retrieval to larger top- K , especially when the model must integrate several retrieved trajectories on ScienceWorld. However, we would avoid claiming that this experiment proves it is the main bottleneck. The comparison is not fully controlled: the ALFWorld discrepancy strongly suggests that the two Qwen variants may differ not only in context length but also in training data or post-training recipe. A more defensible conclusion is therefore that long-context support is a plausible and practically important factor behind the improved large- K behavior of Qwen 2.5 7B 1M, while other factors remain entangled in this particular model pair.

Also, surprisingly, the 1M variant demonstrated weaker performance for dynamic retrieval compared to static retrieval on mismatched scenarios. One may expect that the Sparse Attention Mechanism from Qwen 2.5 7B 1M would help to reduce the instability of dynamic retrieval, but it does not seem to be the case.

D.4 Impact of trajectory formatting on retrieval performance

All the trajectories are stored as raw chat-formatted data in standard OpenAI message format (see example in Figure 6). However, when retrieved, we can prepend the trajectory examples in a different format to make them shorter, and hopefully easier for the model to process in-context. Figure 3 provides examples of different trajectory formatting. Please note that once the retrieved trajectory is formatted, it is appended

EXPRAg type	Top- <i>K</i>	Index	ALFWorld			ScienceWorld		
			Easy tasks	Hard tasks	All tasks	Easy tasks	Hard tasks	All tasks
No RAG	0	–	-28.77	-19.67	-24.63	+2.35	0.00	+1.35
static	1	all	-9.59	0.00	-5.22	-2.36	+1.56	-0.67
		easy	-15.07	-16.40	-15.68	+1.18	-3.12	-0.67
		hard	-24.65	-3.28	-14.92	0.00	-1.56	-0.67
dynamic	1	all	-9.59	-8.19	-8.95	+1.18	-7.81	-2.69
		easy	-16.44	-18.04	-17.16	0.00	-3.12	-1.34
		hard	-19.17	-8.20	-14.18	+9.41	-1.56	+4.70
static	2	all	-12.33	-29.51	-20.15	+7.06	+3.12	+5.37
		easy	-4.11	-24.59	-13.43	+3.53	0.00	+2.01
		hard	-26.02	-14.75	-20.89	+3.53	+1.56	+2.68
dynamic	2	all	-19.18	-13.12	-16.41	+8.23	+6.25	+7.39
static	4	all	+1.37	-6.55	-2.24	+3.53	+23.43	+12.08
		easy	-8.22	-8.19	-8.21	+8.24	0.00	+4.70
		hard	+15.06	-8.20	+4.47	+10.58	+17.19	+13.42
dynamic	4	all	-1.37	-16.39	-8.21	+9.41	+21.87	+14.77
		easy	-8.22	-22.95	-14.92	-3.53	+28.12	+10.07
		hard	+5.48	-4.91	+0.75	+23.53	-9.38	+9.40

Table 14: **Difference between Qwen 2.5 7B 1M and Qwen 2.5 7B in zero-shot performance.** Each cell reports **Qwen 2.5 7B 1M – Qwen 2.5 7B** for the corresponding retrieval configuration and evaluation split. Positive values indicate that the 1M-context model performs better; negative values indicate that the standard-context 7B model performs better.

EXPRAg type	Top- <i>K</i>	Index	ALFWorld			ScienceWorld		
			Easy tasks	Hard tasks	All tasks	Easy tasks	Hard tasks	All tasks
No RAG	0	–	10.96	3.28	7.46	5.88	9.38	7.38
static	1	all	31.51	13.11	23.13	23.53	10.94	18.12
		easy	34.25	4.92	20.90	23.53	1.56	14.09
		hard	15.07	24.59	19.40	10.59	14.06	12.08
dynamic	1	all	32.88 $\uparrow$ 1.4	31.15 $\uparrow$ 18.0	32.09 $\uparrow$ 9.0	–	–	–
		easy	38.36 $\uparrow$ 4.1	3.28 $\downarrow$ 1.6	22.39 $\uparrow$ 1.5	21.18 $\downarrow$ 2.4	3.12 $\uparrow$ 1.6	13.42 $\downarrow$ 0.7
		hard	17.81 $\uparrow$ 2.7	29.51 $\uparrow$ 4.9	23.13 $\uparrow$ 3.7	9.41 $\downarrow$ 1.2	14.06 0.0	11.41 $\downarrow$ 0.7
static	2	all	43.84	37.70	41.04	28.24	17.19	23.49
		easy	35.62	1.64	20.15	23.53	4.69	15.44
		hard	23.29	34.43	28.36	12.94	20.31	16.11
dynamic	2	all	45.21 $\uparrow$ 1.4	31.15 $\downarrow$ 6.6	38.81 $\downarrow$ 2.2	29.41 $\uparrow$ 1.2	21.88 $\uparrow$ 4.7	26.17 $\uparrow$ 2.7
static	4	all	54.79	39.34	47.76	35.29	14.06	26.17
		easy	54.79	3.28	31.34	29.41	0.00	16.78
		hard	20.55	40.98	29.85	10.59	17.19	13.42
dynamic	4	all	50.68 $\downarrow$ 4.1	44.26 $\uparrow$ 4.9	47.76 0.0	30.59 $\downarrow$ 4.7	23.44 $\uparrow$ 9.4	27.52 $\uparrow$ 1.4
		easy	45.21 $\downarrow$ 9.6	8.20 $\uparrow$ 4.9	28.36 $\downarrow$ 3.0	31.76 $\uparrow$ 2.4	0.00 0.0	18.12 $\uparrow$ 1.3
		hard	17.81 $\downarrow$ 2.7	37.70 $\downarrow$ 3.3	26.87 $\downarrow$ 3.0	16.47 $\uparrow$ 5.9	18.75 $\uparrow$ 1.6	17.45 $\uparrow$ 4.0

Table 15: **Results for EXPRAg inference without training across ALFWorld and ScienceWorld for Llama 3.1 8B.** The **bold** values mark the best top-*k* setting for each (EXPRAg type, index) within a given dataset split. For dynamic retrieval rows, the superscript arrows report the per-cell difference against the corresponding static row with the same index and top-*k*: $\uparrow$ indicates improvement, $\downarrow$ indicates a decline, and black values denote ties.

to the system prompt and treated as plain text by the model; it is therefore different from the chat template used by the model.

Table 17 compares the impact of trajectory formatting on final EXPRAg performance. Results indicate that the effect of trajectory formatting is strongly backbone-dependent. JSON-based formats are clearly preferable for the Qwen models, where textual formatting can degrade sharply at larger top-*K*, but this

EXPRAG type	Top- K	Index	ALFWorld			ScienceWorld		
			Easy tasks	Hard tasks	All tasks	Easy tasks	Hard tasks	All tasks
No RAG	0	–	1.37	0.00	0.75	3.53	0.00	2.01
static	1	all	8.22	3.28	5.97	15.29	10.94	13.42
		easy	9.59	0.00	5.22	14.12	3.12	9.40
		hard	4.11	3.28	3.73	8.24	9.38	8.72
dynamic	1	all	10.96 ^{↑ 2.7}	3.28 ^{0.0}	7.46 ^{↑ 1.5}	15.29 ^{0.0}	7.81 ^{↓ 3.1}	12.08 ^{↓ 1.3}
		easy	10.96 ^{↑ 1.4}	1.64 ^{↑ 1.6}	6.72 ^{↑ 1.5}	9.41 ^{↓ 4.7}	3.12 ^{0.0}	6.71 ^{↓ 2.7}
		hard	15.07 ^{↑ 11.0}	0.00 ^{↓ 3.3}	8.21 ^{↑ 4.5}	9.41 ^{↑ 1.2}	4.69 ^{↓ 4.7}	7.38 ^{↓ 1.3}
static	2	all	15.07	9.84	12.69	9.41	14.06	11.41
		easy	10.96	0.00	5.97	10.59	0.00	6.04
		hard	9.59	6.56	8.21	5.88	15.62	10.07
dynamic	2	all	6.85 ^{↓ 8.2}	1.64 ^{↓ 8.2}	4.48 ^{↓ 8.2}	14.12 ^{↑ 4.7}	17.19 ^{↑ 3.1}	15.44 ^{↑ 4.0}
static	4	all	12.33	4.92	8.96	22.35	15.62	19.46
		easy	–	–	–	16.47	0.00	9.40
		hard	6.85	3.28	5.22	9.41	12.50	10.74
dynamic	4	all	12.33 ^{0.0}	3.28 ^{↓ 1.6}	8.21 ^{↓ 0.8}	24.71 ^{↑ 2.4}	7.81 ^{↓ 7.8}	17.45 ^{↓ 2.0}
		easy	21.92	0.00	11.94	22.35 ^{↑ 5.9}	1.56 ^{↑ 1.6}	13.42 ^{↑ 4.0}
		hard	6.85 ^{0.0}	1.64 ^{↓ 1.6}	4.48 ^{↓ 0.7}	7.06 ^{↓ 2.4}	10.94 ^{↓ 1.6}	8.72 ^{↓ 2.0}

Table 16: **Results for EXPRAG inference without training across ALFWorld and ScienceWorld for Llama 3.2 3B.** The **bold** values mark the best top- k setting for each (EXPRAG type, index) within a given dataset split. For dynamic retrieval rows, the superscript arrows report the per-cell difference against the corresponding static row with the same index and top- k : [↑] indicates improvement, [↓] indicates a decline, and black values denote ties.

chat JSON
<pre>{ "role": "assistant", "content": "look" }, { "role": "user", "content": "You are in the middle of a room. Looking quickly around you, you see nothing." },</pre>
agentic JSON
<pre>{ "role": "action", "content": "look" }, { "role": "observation", "content": "You are in the middle of a room. Looking quickly around you, you see nothing." },</pre>
compact JSON
<pre>{ "action": "look" }, { "observation": "You are in the middle of a room. Looking quickly around you, you see nothing." },</pre>
textual
<pre>Assistant: look User: You are in the middle of a room. Looking quickly around you, you see nothing.</pre>

Figure 3: Different trajectory formats: *chat JSON*, *agentic JSON*, *compact JSON*, and *textual*.

pattern does not hold uniformly for Ministral 3-8B and Gemma 3 4B, for which textual formatting is

often competitive and sometimes best. Among the JSON variants, chat JSON is a strong and relatively robust default, especially at larger K , but it is not a universal optimum: compact and agentic JSON each outperform it in some settings. We therefore use chat JSON in the main experiments as a consistent default rather than because it dominates every model-format combination.

Format	ALFWorld				ScienceWorld			
	Ministral 3-8B	Gemma 3-4B	Qwen 2.5-7B	Qwen 2.5-7B-1M	Ministral 3-8B	Gemma 3-4B	Qwen 2.5-7B	Qwen 2.5-7B-1M
Zero-shot	4.5	0.8	29.9	5.2	10.4	2.0	2.6	3.4
<i>Top-1 trajectories</i>								
Chat JSON	39.3	12.7	67.2	61.9	26.2	3.4	12.1	11.4
Agentic JSON	39.6	12.7	61.9	64.2	27.5	0.7	4.0	8.7
Compact JSON	41.8	8.2	61.2	62.0	22.2	4.7	6.0	7.4
Textual	43.3	11.9	41.8	55.2	33.6	6.7	0.7	8.1
<i>Top-2 trajectories</i>								
Chat JSON	51.0	11.9	82.8	62.7	34.7	5.1	10.7	17.5
Agentic JSON	48.5	10.5	73.9	64.2	36.2	2.0	1.3	12.1
Compact JSON	49.3	14.9	79.1	66.4	30.9	5.4	9.4	16.8
Textual	52.2	12.7	14.9	45.5	32.9	16.1	1.3	5.4
<i>Top-4 trajectories</i>								
Chat JSON	64.2	20.9	83.6	81.3	34.5	8.1	7.4	23.5
Agentic JSON	56.0	19.4	81.3	74.6	30.9	3.4	0.7	14.1
Compact JSON	59.0	20.2	79.1	78.4	29.5	9.4	10.7	22.8
Textual	48.5	19.4	1.5	51.5	30.9	14.1	0.7	7.4
<i>Top-5 trajectories</i>								
Chat JSON	67.2	19.4	78.4	76.1	31.5	8.7	10.7	24.8
Agentic JSON	61.2	12.7	83.6	75.4	34.1	3.5	0.7	5.4
Compact JSON	59.7	14.2	70.9	81.3	38.3	7.4	8.7	24.2
Textual	44.0	18.7	0.0	75.4	29.5	10.7	1.3	3.4

Table 17: Impact of trajectory formatting on EXPRAG performance (all tasks, official valid unseen split)

E Generalization Dynamics: Agents keep improving when trained longer.

During preliminary experiments, we found that longer fine-tuning can yield substantially stronger downstream agent performance. Across ALFWorld and ScienceWorld, and for both LoRA and EXPRAG-LoRA, task success often continues to improve after validation loss has reached its minimum and started increasing, resulting in a weak correspondence between validation loss and rollout performance. This occurs both for unseen instances of task groups observed during training (e.g., train/easy $\rightarrow$ valid-unseen/easy) and for held-out task groups (e.g., train/easy $\rightarrow$ valid-unseen/hard), including task types not observed during fine-tuning.

Motivated by this observation, we train for up to 50 epochs and evaluate multiple checkpoints rather than selecting models solely by validation-loss minima. Out-of-distribution performance often continues improving well beyond 10 epochs and, in several settings, reaches its best values near the end of the 50-epoch runs despite rising validation loss. Below, we report these dynamics in detail.

E.1 Experimental setting.

We fine-tune Llama-3.1-8B-Instruct with LoRA adapters on **easy** tasks only (as defined in Section 4.1). Training follows the same supervised setup as the main fine-tuning experiments (Appendix C.4): trajectories are formatted as multi-turn chats and we compute the cross-entropy loss on assistant tokens only, with greedy decoding at inference time (temperature = 0; Table 6). We train for up to 50 epochs and evaluate a set of checkpoints throughout training (epochs 1 to 50), every 5 epochs.

Methods and evaluation. We report curves for (i) **LoRA (no retrieval)** and (ii) **EXPRAG-LoRA**, which performs retrieval-augmented fine-tuning (i.e., the EXPRAG memory block is included in the

training context) and uses the same retrieval pipeline at inference time. For each checkpoint, we report: (a) the **validation loss** (blue line; left y -axis), computed on held-in validation trajectories, and (b) **agent performance** (right y -axis) obtained by executing the agent in the environment, measured as success rate (green) and average episode score (orange). We evaluate both **in-distribution** generalization to unseen instances of the **easy** task groups (labeled ind / cross-scene) and **out-of-distribution** generalization to **hard** task groups not seen during training (labeled ood / cross-task). For EXPRAG-LoRA we use a matched index: easy evaluations retrieve from an index built on easy-task training trajectories, and hard evaluations retrieve from a hard-task training index, consistent with the protocol described in Section 4.4.

E.2 Results on ALFWorld.

Figure 4 shows that validation loss typically bottoms out within the first few epochs and then increases steadily, while both in-distribution performance (Figures 4a and 4b) and out-of-distribution performance (Figures 4c and 4d) can continue improving much later into training. This yields a weak (and sometimes negative) correspondence between validation loss and agent success, illustrating why early stopping based solely on validation loss can miss the best-performing checkpoints on unseen tasks. We also observe that the extended-training regime can be non-monotonic, with occasional late-training instabilities for some settings (e.g., sharp drops for certain checkpoints), further motivating reporting multiple checkpoints rather than selecting by loss alone.

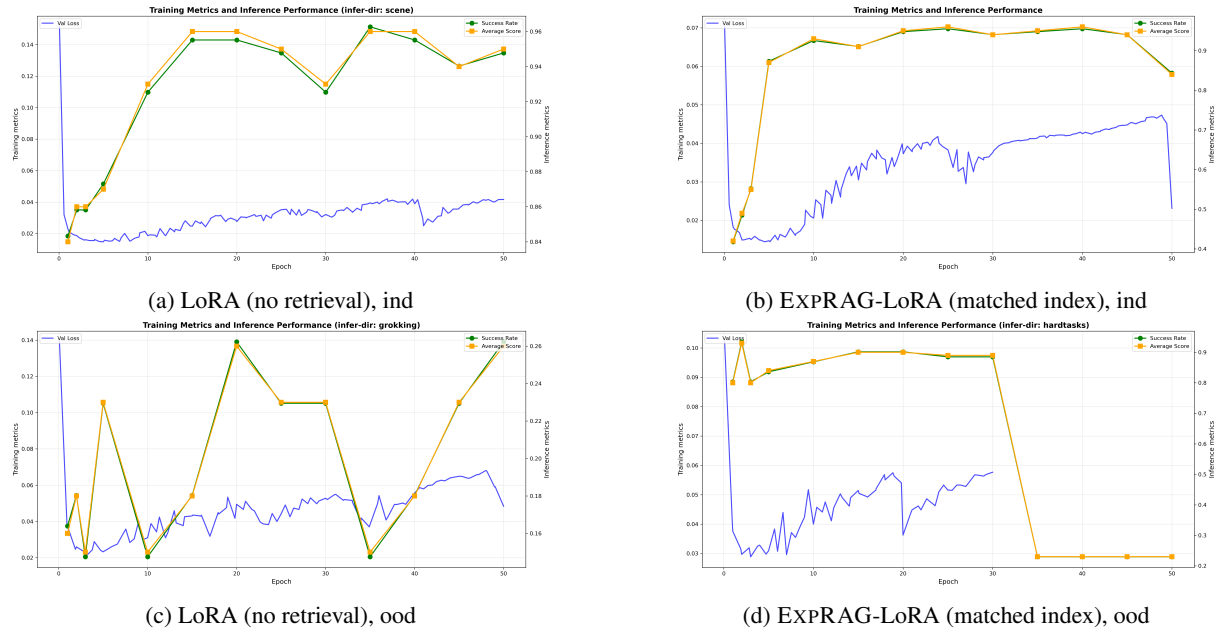


Figure 4: **Longer fine-tuning can improve generalization despite rising validation loss.** Comparison of validation loss and inference performance with respect to number of training epochs on ALFWorld for Llama-3.1-8B. Blue: validation cross-entropy (left axis; lower is better). Green/orange: rollout success rate and average episode score (right axis; higher is better) evaluated at multiple checkpoints during 50-epoch fine-tuning. Top: easy→easy (in-distribution). Bottom: easy→hard (out-of-distribution). For EXPRAG-LoRA, retrieval uses a matched index for each evaluation split.

E.3 Results on ScienceWorld.

We observe the same qualitative pattern in ScienceWorld (Figure 5). Despite validation loss increasing after its early minimum, success on both easy (Figures 5a and 5b) and hard tasks (Figures 5c and 5d) often improves at later epochs, with best checkpoints sometimes appearing near the end of training. Overall, these curves support the main-text conclusion that longer fine-tuning can materially improve generalization, and that validation loss alone is an unreliable proxy for downstream agent success in this setting.

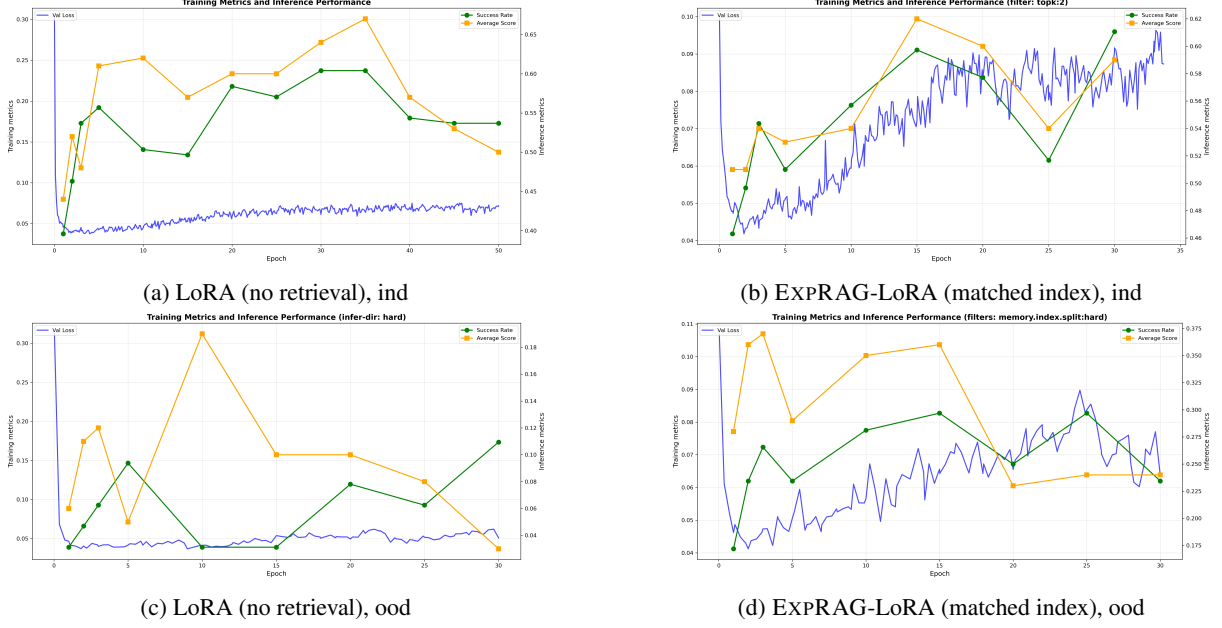


Figure 5: **Longer fine-tuning can improve generalization despite rising validation loss.** Comparison of validation loss and inference performance with respect to number of training epochs on ScienceWorld for Llama-3.1-8B. Blue: validation cross-entropy (left axis). Green/orange: rollout success rate and average episode score (right axis) across checkpoints during 50-epoch fine-tuning. Top: easy→easy (in-distribution). Bottom: easy→hard (out-of-distribution). For EXPRAG-LoRA, retrieval uses a matched index for each evaluation split.

F Statistical Reliability Analysis

We report two complementary forms of uncertainty throughout the paper. When multiple training or inference seeds are available, we report variation across seeds. For paired method comparisons on the same evaluation instances, we additionally use task-level paired tests, which account for the fact that two methods are evaluated on identical tasks rather than on independent samples.

Paired task-level analysis. For each pair of methods, we convert every evaluated task instance into a binary success/failure outcome for both methods. We then compute (i) a paired bootstrap confidence interval over the difference in success rate, using 10,000 bootstrap resamples and a 95% interval, and (ii) a McNemar test over the discordant pairs, i.e., tasks solved by exactly one of the two methods. The bootstrap interval estimates the plausible range of the success-rate difference, while the McNemar test asks whether the number of method wins is reliably larger than the number of baseline wins.

The detailed paired analyses for inference-only and fine-tuning results are reported below.

F.1 Statistical Reliability of Inference-Only ExpRAG

Table 18 reports paired task-level analyses for inference-only EXPRAG against the strict zero-shot NO RAG baseline on hard-task evaluations.

F.2 Statistical Reliability of Fine-Tuning Results

In Table 3, significance markers are based on the paired McNemar test against the corresponding LoRA baseline: $^{\dagger}p < 0.1$, $^*p < 0.05$, $^{**}p < 0.01$, and $^{***}p < 0.001$. Bold entries in hard-task columns indicate improvements significant at $p < 0.05$.

Table 19 reports the paired task-level analyses for the out-of-distribution hard-task comparisons in Table 3. Overall, EXPRAG-LoRA reliably improves over LoRA on ALFWorld-hard for all analyzed backbones and on ScienceWorld-hard for all analyzed backbones except Gemma 3-4B. The evidence is weaker for inference-only EXPRAG over LoRA in several settings, and some gains over LoRA with inference-time EXPRAG are not significant even when EXPRAG-LoRA improves significantly over bare LoRA.

Backbone	Benchmark	Comparison	Δ pp.	95% bootstrap CI	McNemar p
Ministral 3-8B					
	ALFWorld-hard	EXPRAG > No RAG	+47.5	[+34.4, +60.7]	1.86×10^{-9}
	ScienceWorld-hard	EXPRAG > No RAG	+15.6	[+6.2, +25.0]	0.0032
Gemma 3-4B					
	ALFWorld-hard	EXPRAG > No RAG	+4.9	[+0.0, +11.5]	0.125
	ScienceWorld-hard	EXPRAG > No RAG	+1.6	[−4.7, +7.8]	0.5000
Qwen 2.5-7B					
	ALFWorld-hard	EXPRAG > No RAG	+45.9	[+31.1, +60.7]	3.83×10^{-7}
	ScienceWorld-hard	EXPRAG > No RAG	+4.7	[−1.6, +12.5]	0.1875
Qwen 2.5-7B-1M					
	ALFWorld-hard	EXPRAG > No RAG	+59.0	[+45.9, +72.1]	1.46×10^{-11}
	ScienceWorld-hard	EXPRAG > No RAG	+12.5	[+3.1, +23.4]	0.0193

Table 18: **Paired statistical reliability analysis for inference-only ExprAG versus zero-shot.** Differences are in success-rate percentage points. Bootstrap intervals are paired task-level 95% confidence intervals from 10,000 resamples; p -values are from McNemar tests over paired binary success/failure outcomes.

Backbone	Benchmark	Comparison	Δ pp.	95% bootstrap CI	McNemar p
Ministral 3-8B					
	ALFWorld-hard	EXPRAG-LoRA > LoRA	+54.1	[+39.3, +68.9]	1.8×10^{-8}
	ALFWorld-hard	EXPRAG-LoRA > LoRA (with ExprAG)	+21.3	[+9.8, +34.4]	0.0012
	ALFWorld-hard	LoRA (with ExprAG) > LoRA	+32.8	[+18.0, +47.5]	9.0×10^{-5}
	ALFWorld-hard	EXPRAG > LoRA	+13.1	[−4.9, +31.1]	0.0977
	ScienceWorld-hard	EXPRAG-LoRA > LoRA	+26.6	[+9.4, +42.2]	0.0023
	ScienceWorld-hard	EXPRAG-LoRA > LoRA (with ExprAG)	+14.1	[+0.0, +28.1]	0.0466
	ScienceWorld-hard	LoRA (with ExprAG) > LoRA	+12.5	[+0.0, +26.6]	0.0577
	ScienceWorld-hard	EXPRAG > LoRA	+3.1	[−9.4, +15.6]	0.4018
Gemma 3-4B					
	ALFWorld-hard	EXPRAG-LoRA > LoRA	+72.1	[+60.7, +83.6]	5.68×10^{-14}
	ALFWorld-hard	EXPRAG-LoRA > LoRA (with ExprAG)	+70.5	[+59.0, +82.0]	1.14×10^{-13}
	ALFWorld-hard	LoRA (with ExprAG) > LoRA	+1.6	[−3.3, +6.6]	0.5000
	ALFWorld-hard	EXPRAG > LoRA	+3.3	[−3.3, +9.8]	0.3125
	ScienceWorld-hard	EXPRAG-LoRA > LoRA	+0.0	[−7.8, +7.8]	0.6562
	ScienceWorld-hard	EXPRAG-LoRA > LoRA (with ExprAG)	−1.6	[−9.4, +6.2]	0.7734
	ScienceWorld-hard	LoRA (with ExprAG) > LoRA	+1.6	[−4.7, +7.8]	0.5000
	ScienceWorld-hard	EXPRAG > LoRA	+0.0	[−7.8, +7.8]	0.6562
Qwen 2.5-7B					
	ALFWorld-hard	EXPRAG-LoRA > LoRA	+68.9	[+55.7, +80.3]	2.56×10^{-12}
	ALFWorld-hard	EXPRAG-LoRA > LoRA (with ExprAG)	+19.7	[+6.6, +32.8]	0.0059
	ALFWorld-hard	LoRA (with ExprAG) > LoRA	+49.2	[+32.8, +65.6]	1.41×10^{-6}
	ALFWorld-hard	EXPRAG > LoRA	+49.2	[+32.8, +65.6]	1.41×10^{-6}
	ScienceWorld-hard	EXPRAG-LoRA > LoRA	+21.9	[+7.8, +35.9]	0.0022
	ScienceWorld-hard	EXPRAG-LoRA > LoRA (with ExprAG)	+6.2	[−7.8, +20.3]	0.2517
	ScienceWorld-hard	LoRA (with ExprAG) > LoRA	+15.6	[+3.1, +28.1]	0.0207
	ScienceWorld-hard	EXPRAG > LoRA	−1.6	[−9.4, +6.2]	0.7734
Qwen 2.5-7B-1M					
	ALFWorld-hard	EXPRAG-LoRA > LoRA	+68.9	[+54.1, +82.0]	6.56×10^{-11}
	ALFWorld-hard	EXPRAG-LoRA > LoRA (with ExprAG)	+23.0	[+13.1, +34.4]	6.10×10^{-5}
	ALFWorld-hard	LoRA (with ExprAG) > LoRA	+45.9	[+27.9, +62.3]	7.55×10^{-6}
	ALFWorld-hard	EXPRAG > LoRA	+39.3	[+21.3, +55.7]	5.81×10^{-5}
	ScienceWorld-hard	EXPRAG-LoRA > LoRA	+15.6	[+1.6, +29.7]	0.0262
	ScienceWorld-hard	EXPRAG-LoRA > LoRA (with ExprAG)	+17.2	[+4.7, +29.7]	0.0096
	ScienceWorld-hard	LoRA (with ExprAG) > LoRA	−1.6	[−12.5, +9.4]	0.7095
	ScienceWorld-hard	EXPRAG > LoRA	+1.6	[−9.4, +12.5]	0.5000

Table 19: **Paired statistical reliability analysis for hard-task fine-tuning results.** Differences are in success-rate percentage points. Bootstrap intervals are paired task-level 95% confidence intervals from 10,000 resamples; p -values are from McNemar tests over paired binary success/failure outcomes.

Index condition	Experience source	Construction	Purpose
Matched Expert	Scripted expert	Full expert index from the hard-task training split: 1,805 trajectories for ALFWorld and 1,254 for ScienceWorld.	Reference condition with task-relevant expert experience.
Empty	None	No trajectories are available for retrieval.	Measures performance when no external experience is available.
Mismatched Expert	Scripted expert	Uses only trajectories from the easy-task training split while evaluating on unseen hard tasks: 1,748 trajectories for ALFWorld and 2,335 for ScienceWorld.	Tests robustness when experience is available but comes from different task groups.
Low-Coverage Expert	Scripted expert	Subsamples the matched hard-task index to five trajectories per hard task type: 15 trajectories for ALFWorld and 65 for ScienceWorld.	Tests whether a large matched experience bank is necessary.
Matched Strong-LLM	Qwen2.5-70B	Replaces scripted-expert trajectories with trajectories generated by Qwen2.5-70B on matched hard tasks. These trajectories are generally more suboptimal than those from the scripted expert.	Tests whether useful experience must come from a scripted expert or can instead be seeded by a stronger model.
Self-Experience	Evaluated agent	Uses the agent’s own first-attempt trajectory on the hard evaluation distribution, successful or failed, as experience for a second attempt. Model parameters are unchanged between attempts.	Tests whether the agent can improve in-context from its own past experience without test-time parameter updates.
Self-Experience + Source	Agent + external source	Combines first-attempt experience with each non-empty external source: Mismatched Expert, Low-Coverage Expert, Matched Expert, or Matched Strong-LLM.	Tests whether self-generated experience remains useful when other sources of experience are simultaneously available.

Table 20: Experience-index conditions used in the robustness experiments in Sec. 4.4.3.

Index condition	Trajectories	Success rate (%)
Matched Expert	1,805	62.2 (1,123/1,805)
Mismatched Expert ^a	1,748	97.5 (1,704/1,748)
Low-Coverage Expert	15	86.7 (13/15)
Matched Strong-LLM	1,799	68.6 (1,234/1,799)

Table 21: Composition of the external ALFWorld experience indices. Success rate is reported as a percentage, followed by successful trajectories over total trajectories. **Notes:** ^a Calculated from the train/easy split, although evaluation is on unseen hard tasks.

G Robustness to Imperfect and Self-Generated Experience

This appendix provides additional details for the experience-robustness experiments in Sec. 4.4.3, including the construction of each experience source, the Self-Experience evaluation protocol, and the complete results for Ministral 3-8B.

Unless stated otherwise, these experiments reuse the same checkpoints and inference configuration as Sec. 4.4.2, described in detail in Appendix C.1. No additional fine-tuning is performed for any robustness condition: only the experience available in the retrieval index changes.

G.1 Experience Index Construction

Table 20 defines the experience sources used in the robustness experiments. Except for **Empty**, retrieval and inference are unchanged; conditions vary only in the source, relevance, coverage, or origin of the available experience.

G.2 Self-Experience Protocol

Self-Experience uses the zero-shot trajectory produced by the agent’s first attempt on each hard evaluation instance as memory for a second attempt. The trajectory is stored regardless of whether the first attempt

Index condition	Trajectories	Success rate (%)	Score = -1 (%)
Matched Expert	1,254	99.0 (1,241/1,254)	0.0 (0/1,254)
Mismatched Expert ^a	2,335	93.7 (2,188/2,335)	0.2 (5/2,335)
Low-Coverage Expert	65	98.5 (64/65)	0.0 (0/65)
Matched Strong-LLM ^b	165	73.3 (121/165)	2.4 (4/165)
Self-Experience ^c	64	3.1 (2/64)	45.3 (29/64)

Table 22: Composition of the ScienceWorld experience indices. Success rate and the proportion with score = -1 are reported as percentages, followed by count over total trajectories. **Notes:** ^a Calculated from the train/easy split, although evaluation is on unseen hard tasks. ^b The Qwen index is subsampled to 15 trajectories per hard task type (165 total). ^c Calculated from zero-shot trajectories in valid_unseen/hard.

succeeds or fails. We then perform the second attempt with these zero-shot trajectories available through the same EXPRAG retrieval mechanism used elsewhere in the paper. Crucially, model parameters remain fixed between the two attempts; adaptation occurs exclusively through the retrieved context.

The first-attempt trajectories use exactly the same representation as the other experience sources. Following Appendix C.3, successful and unsuccessful trajectories are identified as such in the memory block, allowing the policy to distinguish demonstrations of successful behavior from failed experience.

On ALFWorld hard tasks, all first attempts fail. Thus, the **Self-Experience**-only condition is especially strict: the second attempt has access to self-generated experience, but none of that experience contains a successful solution.

G.3 Full Results

Tables 23, 24, and 25 report results under each experience condition. As in Fig. 2, we evaluate three policies: inference-only EXPRAG, LoRA with EXPRAG introduced only at inference time, and retrieval-aware EXPRAG-LoRA. All methods within a row receive experience from the same source; thus, differences reflect how the policy uses the available experience rather than differences in retrieval configuration.

Method	Experience source	Mean (%)	Within-game 95% CI	Retention (%)
ExpPRAG	Matched Expert	44.3	[37.8, 50.8]	100.0 [100.0, 100.0]
	Empty	0.0	[-2.2, 2.2]	0.0 [0.0, 0.0]
	Mismatched Expert	8.7	[4.6, 13.1]	19.8 [10.5, 31.2]
	Low-Coverage Expert	36.1	[29.1, 43.2]	81.5 [64.1, 102.7]
	Matched Strong-LLM	44.3	[37.4, 51.1]	100.0 [82.7, 120.8]
	Self-Experience	0.0	[-2.2, 2.2]	0.0 [0.0, 0.0]
	Self-Experience + Mismatched Expert	19.7	[14.2, 25.3]	44.4 [31.3, 59.7]
	Self-Experience + Low-Coverage Expert	23.0	[16.9, 29.4]	51.9 [38.2, 68.2]
	Self-Experience + Matched Expert	34.4	[28.1, 40.9]	77.8 [63.4, 94.0]
LoRA + ExpPRAG	Self-Experience + Matched Strong-LLM	55.7	[48.7, 62.8]	125.9 [106.0, 150.8]
	Matched Expert	68.3	[61.8, 74.5]	100.0 [100.0, 100.0]
	Empty	31.1	[25.3, 36.9]	45.6 [35.2, 56.8]
	Mismatched Expert	29.5	[24.0, 35.1]	43.2 [33.6, 53.7]
	Low-Coverage Expert	50.8	[44.6, 57.0]	74.4 [63.4, 86.2]
	Matched Strong-LLM	65.6	[59.7, 71.3]	96.0 [83.5, 109.9]
	Self-Experience	31.1	[25.2, 37.0]	45.6 [34.9, 57.3]
	Self-Experience + Mismatched Expert	39.3	[33.0, 45.5]	57.6 [46.4, 69.7]
	Self-Experience + Low-Coverage Expert	57.4	[50.8, 63.9]	84.0 [71.5, 98.3]
ExpPRAG-LoRA	Self-Experience + Matched Expert	67.2	[60.4, 73.9]	98.4 [88.7, 108.9]
	Self-Experience + Matched Strong-LLM	72.1	[66.5, 77.7]	105.6 [93.2, 119.3]
	Matched Expert	91.8	[87.5, 95.9]	100.0 [100.0, 100.0]
	Empty	30.6	[24.3, 37.3]	33.3 [25.9, 41.1]
	Mismatched Expert	41.5	[35.4, 47.5]	45.2 [37.4, 53.0]
	Low-Coverage Expert	83.6	[77.8, 89.2]	91.1 [84.1, 98.2]
	Matched Strong-LLM	90.2	[85.9, 94.3]	98.2 [92.5, 104.3]
	Self-Experience	60.7	[53.7, 67.5]	66.1 [57.6, 74.5]
	Self-Experience + Mismatched Expert	41.0	[34.6, 47.3]	44.6 [36.8, 52.8]
	Self-Experience + Low-Coverage Expert	70.5	[64.7, 76.2]	76.8 [69.5, 84.0]
	Self-Experience + Matched Expert	93.4	[89.3, 97.2]	101.8 [98.8, 105.0]
	Self-Experience + Matched Strong-LLM	90.2	[85.9, 94.2]	98.2 [93.1, 103.7]

Table 23: Success rate (%) on ALFWorld unseen hard tasks. Within-game CIs are 95% normalized bootstrap CIs. Retention is relative to the Matched Expert mean for the same method; brackets report its joint paired bootstrap CI. All robustness conditions use the same trained checkpoints; only the available experience changes.

Method	Experience source	Mean (%)	Within-game 95% CI	Retention (%)
EXP-RAG	Matched Expert	20.3	[15.8, 25.1]	100.0 [100.0, 100.0]
	Empty	4.7	[1.4, 8.1]	23.1 [9.4, 41.7]
	Mismatched Expert	15.6	[11.9, 19.6]	76.9 [52.2, 111.8]
	Low-Coverage Expert	22.9	[18.4, 27.7]	112.8 [84.8, 151.5]
	Matched Strong-LLM (subsampling)	26.6	[21.9, 31.4]	130.8 [97.7, 177.4]
	Self-Experience	7.8	[4.6, 11.1]	38.5 [21.1, 60.7]
	Self-Experience + Mismatched Expert	7.8	[4.6, 11.1]	38.5 [21.1, 60.7]
	Self-Experience + Low-Coverage Expert	16.7	[12.6, 21.1]	82.1 [57.9, 114.3]
	Self-Experience + Matched Expert	17.2	[12.8, 21.8]	84.6 [63.6, 110.3]
	Self-Experience + Matched Strong-LLM	18.2	[14.2, 22.5]	89.7 [64.9, 123.3]
LoRA + EXP-RAG	Matched Expert	31.2	[25.2, 37.3]	100.0 [100.0, 100.0]
	Empty	14.6	[10.7, 18.7]	46.7 [30.9, 66.7]
	Mismatched Expert	12.5	[8.6, 16.6]	40.0 [26.3, 56.1]
	Low-Coverage Expert	22.9	[17.5, 28.4]	73.3 [54.7, 95.8]
	Matched Strong-LLM (subsampling)	33.9	[28.4, 39.2]	108.3 [84.0, 139.6]
	Self-Experience	6.2	[3.4, 9.1]	20.0 [9.8, 32.7]
	Self-Experience + Mismatched Expert	9.9	[6.7, 13.2]	31.7 [18.6, 48.1]
	Self-Experience + Low-Coverage Expert	12.5	[8.2, 17.0]	40.0 [25.0, 59.3]
	Self-Experience + Matched Expert	21.9	[17.1, 26.8]	70.0 [51.5, 93.3]
	Self-Experience + Matched Strong-LLM	18.8	[14.6, 23.2]	60.0 [42.9, 81.2]
EXP-RAG-LoRA	Matched Expert	47.9	[41.0, 54.8]	100.0 [100.0, 100.0]
	Empty	3.1	[0.1, 6.3]	6.5 [2.1, 12.4]
	Mismatched Expert	9.4	[6.0, 13.0]	19.6 [11.3, 29.6]
	Low-Coverage Expert	54.7	[47.9, 61.4]	114.1 [96.2, 136.2]
	Matched Strong-LLM (subsampling)	38.0	[32.2, 44.1]	79.3 [65.3, 96.1]
	Self-Experience	4.7	[1.9, 7.5]	9.8 [4.0, 17.3]
	Self-Experience + Mismatched Expert	7.3	[4.6, 10.1]	15.2 [8.0, 24.4]
	Self-Experience + Low-Coverage Expert	14.1	[9.5, 19.0]	29.3 [19.6, 40.9]
	Self-Experience + Matched Expert	29.7	[23.5, 36.1]	62.0 [50.0, 75.3]
	Self-Experience + Matched Strong-LLM	25.5	[20.8, 30.6]	53.3 [40.6, 68.2]

Table 24: Success rate (%) on ScienceWorld unseen hard tasks. Within-game CIs are 95% normalized bootstrap CIs. Retention is relative to the Matched Expert mean for the same method; brackets report its joint paired bootstrap CI. The matched strong-LLM bank is reported as Qwen Teacher (Subsampling) in the analysis output. Experimental settings otherwise match Table 23.

Method	Experience source	Mean score	Within-game 95% CI	Retention (%)
EXP-RAG	Matched Expert	0.399	[0.352, 0.447]	100.0 [100.0, 100.0]
	Empty	0.146	[0.110, 0.183]	36.5 [27.5, 47.2]
	Mismatched Expert	0.288	[0.253, 0.326]	72.2 [58.5, 88.2]
	Low-Coverage Expert	0.384	[0.338, 0.432]	96.1 [81.2, 113.5]
	Matched Strong-LLM (subsampled)	0.421	[0.374, 0.466]	105.3 [89.4, 123.8]
	Self-Experience	0.204	[0.169, 0.239]	51.0 [40.2, 63.3]
	Self-Experience + Mismatched Expert	0.179	[0.146, 0.214]	44.9 [34.5, 57.1]
	Self-Experience + Low-Coverage Expert	0.295	[0.252, 0.340]	73.9 [60.7, 89.5]
	Self-Experience + Matched Expert	0.311	[0.266, 0.358]	77.9 [65.8, 90.7]
	Self-Experience + Matched Strong-LLM	0.331	[0.289, 0.374]	82.8 [70.1, 97.7]
LoRA + EXP-RAG	Matched Expert	0.490	[0.438, 0.542]	100.0 [100.0, 100.0]
	Empty	0.228	[0.187, 0.269]	46.4 [36.9, 56.8]
	Mismatched Expert	0.252	[0.214, 0.291]	51.4 [41.7, 61.9]
	Low-Coverage Expert	0.407	[0.357, 0.458]	83.1 [72.1, 95.3]
	Matched Strong-LLM (subsampled)	0.520	[0.475, 0.564]	106.0 [92.8, 121.3]
	Self-Experience	0.215	[0.186, 0.243]	43.8 [35.8, 52.3]
	Self-Experience + Mismatched Expert	0.242	[0.211, 0.274]	49.4 [40.2, 59.6]
	Self-Experience + Low-Coverage Expert	0.277	[0.234, 0.321]	56.4 [46.1, 67.8]
	Self-Experience + Matched Expert	0.377	[0.332, 0.424]	77.0 [66.4, 88.5]
	Self-Experience + Matched Strong-LLM	0.337	[0.298, 0.377]	68.8 [57.8, 81.0]
EXP-RAG-LoRA	Matched Expert	0.662	[0.609, 0.713]	100.0 [100.0, 100.0]
	Empty	0.112	[0.075, 0.150]	16.9 [12.4, 21.8]
	Mismatched Expert	0.197	[0.158, 0.237]	29.7 [23.2, 37.1]
	Low-Coverage Expert	0.679	[0.628, 0.731]	102.6 [93.4, 112.8]
	Matched Strong-LLM (subsampled)	0.532	[0.480, 0.586]	80.4 [71.8, 89.7]
	Self-Experience	0.181	[0.147, 0.215]	27.3 [21.4, 34.2]
	Self-Experience + Mismatched Expert	0.195	[0.162, 0.228]	29.4 [22.9, 36.7]
	Self-Experience + Low-Coverage Expert	0.285	[0.239, 0.334]	43.1 [35.7, 51.4]
	Self-Experience + Matched Expert	0.442	[0.387, 0.498]	66.8 [58.2, 75.8]
	Self-Experience + Matched Strong-LLM	0.392	[0.346, 0.440]	59.2 [50.7, 68.7]

Table 25: Transformed score (0–1) on ScienceWorld unseen hard tasks. Before averaging, we replace the failure score -1 with 0, so failures contribute zero rather than a negative value and do not create an asymmetric penalty relative to the nonnegative task scores. Within-game CIs are 95% normalized bootstrap CIs. Retention is relative to the Matched Expert mean for the same method; brackets report its joint paired bootstrap CI. Experimental settings otherwise match Table 24.

Index condition	Method	ALFWorld Hard tasks (oo-d)	ScienceWorld Hard tasks (oo-d)
Empty <i>Gemma 3-4B</i>	EXP-RAG	4.9 → 0.0	6.3 → 4.7
	LoRA + EXP-RAG	3.3 → 1.6	7.8 → 6.3
	EXP-RAG-LoRA	73.8 → 4.9	4.7 → 0.0
Mismatched Expert <i>Gemma 3-4B</i>	EXP-RAG	4.9 → 4.9	6.3 → 1.6
	LoRA + EXP-RAG	3.3 → 4.9	7.8 → 7.8
	EXP-RAG-LoRA	73.8 → 9.8	4.7 → 3.1
Empty <i>Qwen 2.5-7B</i>	EXP-RAG	68.9 → 22.9	6.3 → 3.1
	LoRA + EXP-RAG	70.5 → 21.3	23.4 → 7.8
	EXP-RAG-LoRA	90.2 → 34.4	29.7 → 4.7
Mismatched Expert <i>Qwen 2.5-7B</i>	EXP-RAG	68.9 → 54.1	6.3 → 0.0
	LoRA + EXP-RAG	70.5 → 24.6	23.4 → 17.2
	EXP-RAG-LoRA	90.2 → 50.8	29.7 → 17.2
Empty <i>Qwen 2.5-7B-1M</i>	EXP-RAG	54.1 → 3.3	7.8 → 3.1
	LoRA + EXP-RAG	68.9 → 23.0	12.5 → 12.5
	EXP-RAG-LoRA	91.8 → 36.1	29.7 → 4.7
Mismatched Expert <i>Qwen 2.5-7B-1M</i>	EXP-RAG	54.1 → 29.5	7.8 → 0.0
	LoRA + EXP-RAG	68.9 → 21.3	12.5 → 6.3
	EXP-RAG-LoRA	91.8 → 60.7	29.7 → 3.1
Empty <i>Llama 3.1-8B</i>	EXP-RAG	34.4 → 3.3	20.3 → 9.4
	LoRA + EXP-RAG	44.3 → 21.3	21.9 → 7.8
	EXP-RAG-LoRA	80.3 → 14.8	21.9 → 3.1
Mismatched Expert <i>Llama 3.1-8B</i>	EXP-RAG	34.4 → 1.6	20.3 → 4.7
	LoRA + EXP-RAG	44.3 → 21.3	21.9 → 4.7
	EXP-RAG-LoRA	80.3 → 34.4	21.9 → 1.6
Empty <i>Llama 3.2-1B</i>	EXP-RAG	0.0 → 0.0	0.0 → 0.0
	LoRA + EXP-RAG	8.2 → 0.0	4.7 → 4.7
	EXP-RAG-LoRA	34.4 → 1.6	10.9 → 1.6
Mismatched Expert <i>Llama 3.2-1B</i>	EXP-RAG	0.0 → 0.0	0.0 → 0.0
	LoRA + EXP-RAG	4.3 → 21.3	4.7 → 0.0
	EXP-RAG-LoRA	80.3 → 34.4	10.9 → 4.7

Table 26: **Index stress test for other backbones.** Each cell reports the success rate (%) as $a \rightarrow b$, where a uses the matched hard-task expert index and b uses either **Empty** (no trajectories) or **Mismatched Expert** (easy-task training trajectories).

G.4 Index Stress Test in other backbones

To complement the complete Ministral 3-8B stress test above, Table 26 reports initial cross-backbone experiment that evaluates only two conditions: **Empty** and **Mismatched Expert**. Each cell gives the success rate with the matched hard-task expert index followed by the success rate under the stressed index. This experiment uses the original fine-tuning evaluation setup; its absolute values should therefore be interpreted independently of the updated Ministral results above.

G.5 Statistical Reliability

Figure 2 and Tables 23, 24, and 25 report 95% within-game normalized bootstrap CIs, using `game_id + seed` as the resampling unit. This normalization removes shared game difficulty and is intended for visual comparisons among the aligned conditions, rather than as standalone absolute uncertainty. The paired task-level testing procedure used for the main fine-tuning comparisons is described separately in Appendix F.

H Prompts

This appendix details the exact prompting interface used in our experiments. For each environment, we use a `minimal` system prompt that specifies: (i) the task setting, (ii) the action interface/grammar, and (iii) the response format (one action per turn).

Restricting turn-level action-space hints. We provide a static list of action templates valid for the whole environment (Figures 7 and 8), rather than per-step instantiated valid-action candidates, even though the environment can provide those candidates. This design removes turn-level action-space hints, requiring the agent to infer plausible next actions from past observations and actions alone. Importantly, this diverges from contemporaneous work that exposes these turn-level candidate actions during rollout, arguably introducing some ground-truth information leakage. In early explorations in our zero-shot setting, we found this stricter setup to reduce backbone performance by approximately 15%.

Figure 7 shows the full ALFWorld system prompt. It includes the complete command inventory and formatting constraints (including one-command output), plus guidance for handling invalid actions (e.g., “Nothing happens”). Figure 8 shows the analogous ScienceWorld prompt, with its environment-specific action grammar and explicit command-format constraints (e.g., pick up <OBJ> and no extra prefix text).

Finally, Figure 6 illustrates how interactions are serialized as chat data. Consistent with the method in Section 3, each trajectory is encoded as a multi-turn chat: environment observations/task context are mapped to user turns, and action strings are mapped to assistant turns. This contrasts with stepwise formatting used by contemporaneous work, where each (h_t, a_t) pair is treated as an independent sample with history re-encoding at every step. We use multi-turn serialization to preserve conversational structure and training/inference consistency with chat models while keeping supervision on assistant action tokens.

As a final note, when using retrieval, we append the retrieved trajectories to the system prompt as a memory block, as described in Section C.3.

Example of (partial) JSON trajectory (ALFWorld)

```
{
  "role": "user",
  "content": "-= Welcome to TextWorld, ALFRED! -=\n\nYou are in the middle of a room.
    Looking quickly around you, you see a bathtubbasin 1, a countertop 2,
    a countertop 1, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a garbagecan 1,
    a handtowelholder 1, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1,
    a towelholder 2, and a towelholder 1.\n\nYour task is to: put a candle in drawer."
},
{
  "role": "assistant",
  "content": "look"
},
{
  "role": "user",
  "content": "You are in the middle of a room. Looking quickly around you, you see nothing."
},
{
  "role": "assistant",
  "content": "go to toilet 1"
},
{
  "role": "user",
  "content": "You arrive at toilet 1. On the toilet 1, you see a spraybottle 3, a spraybottle 2,
    and a spraybottle 1."
},
{
  "role": "assistant",
  "content": "go to drawer 1"
}
....
```

Figure 6: Example of a partial JSON trajectory in ALFWorld.

System prompt for ALFWorld

Interact with a household to solve a task. Here is the list of commands:

Available commands:

- look: look around your current location
- inventory: check your current inventory
- go to (receptacle): move to a receptacle
- open (receptacle): open a receptacle
- close (receptacle): close a receptacle
- take (object) from (receptacle): take an object from a receptacle
- move (object) to (receptacle): place an object in or on a receptacle
- examine (something): examine a receptacle or an object
- use (object): use an object
- heat (object) with (receptacle): heat an object using a receptacle
- clean (object) with (receptacle): clean an object using a receptacle
- cool (object) with (receptacle): cool an object using a receptacle
- slice (object) with (object): slice an object using a sharp object

You can only perform these commands. The text command you generate must be strictly compliant with the command grammar.

Only provide one single command in your response. This is an example command: go to shelf 1

Each time you provide a command, the household will return the outcome of it, and you will have to provide another command.

Note that you cannot carry two items at the same time so when you take an item, you must remember it and possibly move it to a receptacle later.

Figure 7: System prompt used for ALFWorld experiments.

I Efficiency Cost of EXPRAG

Table 27 reports how the value of top- K affects execution time and context size for EXPRAG. We note that the average number of prompt tokens scales almost linearly with k . The average number of steps required to perform the task is decreasing with higher top- k , meaning that the agent is able to solve the tasks faster. Total processing time is therefore growing much slower. We stick with top-2 as the best efficiency-effectiveness tradeoff for the follow-up experiments.

System prompt for ScienceWorld				
Interact with the environment to solve a scientific experiment.				
In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, hallway.				
All containers in the environment have already been opened, you can directly get items from the containers.				
The available actions are:				
open OBJ: open a container				
close OBJ: close a container				
activate OBJ: activate a device				
deactivate OBJ: deactivate a device				
connect OBJ to OBJ: connect electrical components				
disconnect OBJ: disconnect electrical components				
use OBJ [on OBJ]: use a device/item				
look around: describe the current room				
examine OBJ: describe an object in detail				
look at OBJ: describe a container's contents				
read OBJ: read a note or book				
move OBJ to OBJ: move an object to a container				
pick up OBJ: move an object to the inventory				
pour OBJ into OBJ: pour a liquid into a container				
mix OBJ: chemically mix a container				
teleport to LOC: teleport to a specific room				
focus on OBJ: signal intent on a task object				
wait: task no action for 10 steps				
wait1: task no action for a step				
For the pick up action: the command is 'pick up <OBJ>'.				
Do not prefix the command with text.				
You can only perform these commands. The text command you generate must be strictly compliant with the command grammar. Only provide one single command in your response.				

Figure 8: System prompt used for ScienceWorld experiments.

	prompt tokens	steps/sample	total time (sec)	time/step
No RAG	489.6	48	2769.59	0.430194
EXPRAG top1	1851.9	34.4	2719.94	0.590649
EXPRAG top2	3096.7	29.8	2927.81	0.734339
EXPRAG top4	5597.7	25.7	3556.04	1.03403
EXPRAG top5	6866.4	25	3657.74	1.09284

Table 27: Model: Ministral 3-8B, dataset: ALFWorld. Efficiency of different EXPRAG variants compared to zero-shot: average number of prompt tokens processed at each step, average number of steps per episode, total generation time for 134 episodes, and average time per step.