Retrieval-Augmented LLM Agents: Learning to Learn from Experience

Anonymous ACL submission

Abstract

While large language models (LLMs) have advanced the development of general-purpose agents, achieving robust generalization to unseen tasks remains a significant challenge. Current approaches typically rely on either fine-tuning or 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 propose to combine these approaches and systematically study how to train retrieval-augmented LLM agents to effectively leverage retrieved trajectories in-context. First, we establish a robust supervised fine-tuning (SFT) recipe using LoRA that outperforms several state-of-the-art agent training pipelines. Second, we provide a detailed analysis of key design choices for experience retrieval, identifying optimal strategies for storage, querying, and trajectory selection. Finally, we propose a pipeline that integrates experience retrieval into the fine-tuning process. Our results demonstrate that this combined approach significantly improves generalization to unseen tasks, providing **a scalable and effective framework for building agents that learn to learn from experience.**

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., 2024c) + ReAct	66.4[f]
IWM (Zhang et al., 2025b)	70.3
Self-Reflection (Zhang et al., 2025b)	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: [a]Liu et al. (2026b), [b]Xia et al. (2026), [c]Yu et al. (2026), [e]Feng et al. (2025), [d]Zhang et al. (2025a), [f]Fei et al. (2025).

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., 2024b; Jimenez et al., 2024; Yang et al., 2024; 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 step for building agents (Chen et al., 2023; Qiao et al., 2024). This approach often yields strong performance on tasks seen during training, but we argue that **truly useful agents should be able to perform a broad range of tasks, including never seen ones, within the environment in which they were trained to operate**.

The in-context learning capabilities of modern LLMs make retrieval-augmented generation (RAG) an attractive inference-time adaptation mechanism, by providing relevant external context (Izacard et al., 2023; Ram et al., 2023). For agents, a natural source of external context is prior experience,

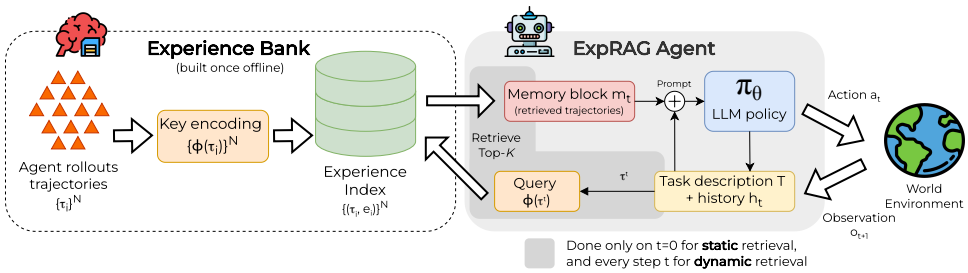


Figure 1: **ExpRAG Agent overview.** ExpRAG augments an LLM agent with retrieval over past experience. Offline, past trajectories are encoded into an experience index. At inference, task and history are used to retrieve top-K relevant trajectories, which are assembled into a memory block and provided 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.

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 ExpRAG, which we adopt in this paper.

This paper studies how to train retrieval-augmented LLM agents that can retrieve relevant experience and learn from it 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 ❶: 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 shows that elaborate memory-augmented pipelines (without training) can provide limited improvement or even underperform compared with a simple experience-retrieval baseline (Best ExpRAG), consistent with Wei et al.'s (2025) observation that simple episodic retrieval is still competitive with self-evolving memory systems. Both results call for **building stronger baselines before exploring complex engineered architectures**.

Contribution ❷: A careful examination of ExpRAG. Secondly, we did not find an off-the-shelf recipe for applying RAG to LLM-based agents. In Section 4.3, we conduct a detailed analysis of key design choices: how to store experience, how to build a retrieval query, how much and when to retrieve, what is the impact of the source of indexed trajectories and of the backbone model. These results show that robust and effective adaptation through ExpRAG is possible.

Contribution ❸: ExpRAG-LoRA. We propose to combine LoRA fine-tuning with ExpRAG, bringing the best of both worlds: strong in-distribution performance combined with out-of-distribution generalization. In Sec. 4.4, we therefore investigate how to best include ExpRAG in a parameter-efficient fine-tuning (PEFT) pipeline.

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, 2024). 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., 2025b; Feng et al., 2025). How-

ever, we find that simple PEFT already outperforms more complex methods, 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. Recent 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., 2025d; 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 both inference-only retrieval and retrieval-augmented fine-tuning, focusing on weaker open-source models.

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 richer read–write memory architecture, but to establish strong foundations for fine-tuning augmented with episodic retrieval: (i) we propose strong baselines and report their main drivers of success, (ii) we investigate whether retrieval-augmented fine-tuning generalizes to unseen episodes, i.e., whether it learns to learn, and (iii) we review the applicability of the method when no relevant experience is available at inference time.

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 task is completed. We denote the full trajectory by $\tau = (\mathcal{T}, o_1, a_1, \ldots, o_{T-1}, a_{T-1}, o_T)$, and the trajectory history available at step t by $h_t = (\mathcal{T}, o_1, a_1, \ldots, o_t)$. 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 <u>unseen</u> 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 $a_t = (a_t^1, \ldots, a_t^{n_t})$, 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., 2025; 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., 2025b; 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:

$$(\tau^1, \ldots, \tau^K) = \arg\operatorname*{topK}_{1 \le i \le N} \phi(q_t) \cdot e_i.$$

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

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 B.

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: **ALFWorld** (Shridhar et al., 2021) and **ScienceWorld** (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 episode outcome to 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 B.2.

4.2 Implementation Details

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

tions are relevant. We serialize trajectories using the chat template described in Section 3 (full templates in Appendix E). 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., 2025b); 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 E.

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 3. We compute cross-entropy loss on assistant tokens only. Key hyperparameters and implementation details are provided in Appendix B.

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) index composition (**all/easy/hard** task-types available for retrieval), (ii) top-k ($K \in \{1, 2, 4\}$), 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 C.4.

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

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

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		all	46.92 ↑ 3.1	34.02 ↑ 0.1	41.04 ↑ 1.7	29.81 ↓ 3.5	17.71 0.0	24.61 ↓ 2.0
		easy	44.75 ↑ 1.8	4.37 ↓ 1.1	26.37 ↑ 0.5	32.65 ↑ 3.8	7.03 ↓ 3.1	21.65 ↑ 0.8
		hard	38.82 ↑ 3.2	39.89 ↑ 6.0	39.30 ↑ 4.5	**26.18** ↑ 3.5	19.53 ↑ 1.6	23.32 ↑ 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		all	57.53 ↑ 4.1	48.36 ↑ 0.3	53.36 ↑ 2.4	43.53 ↓ 0.8	21.88 0.0	34.23 ↓ 0.4
		easy	57.53 ↑ 10.3	6.56 ↓ 4.9	34.33 ↑ 3.4	43.73 ↑ 0.2	9.64 ↓ 2.6	29.08 ↓ 1.0
		hard	39.27 ↓ 0.5	48.09 ↑ 1.1	43.28 ↑ 0.2	25.10 ↑ 1.0	22.92 ↑ 7.6	**24.16** ↑ 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		all	**70.55** ↑ 7.1	**55.74** ↓ 9.3	**63.81** ↓ 0.4	**44.71** ↑ 2.7	**22.66** ↑ 0.4	**35.24** ↑ 1.7
		easy	**71.69** ↑ 8.7	**9.29** ↑ 2.2	**43.28** ↑ 5.7	**46.67** ↑ 26.3	**10.94** ↓ 6.8	**31.32** ↑ 12.1
		hard	**42.93** ↑ 2.7	**65.57** ↑ 2.7	**53.24** ↑ 2.7	17.25 ↓ 24.3	**23.96** ↑ 10.9	20.14 ↓ 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 ± std for every entry is reported in Appendix C.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: ↑ indicates improvement, ↓ indicates a decline, and black values denote ties.

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. (2025b)). 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.

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 is not to reach higher performance on tasks seen during training, but on new unseen tasks within the training environment.

4.4.1 Preliminary: Generalization Dynamics in LLM Agent Fine-Tuning

During early experiments, we found that training for substantially more epochs than is typical in LLM agent adaptation pipelines can yield much stronger performance. Across both ALFWorld and ScienceWorld, and for both LoRA and ExpRAG-LoRA fine-tuning, out-of-distribution task success often continues to improve long after validation loss has started increasing (which one may see as overfitting), with the best unseen-task checkpoints frequently occurring well beyond conventional early-stopping points. Accordingly, we observe a weak correlation between validation loss and task success. This pattern appears not only for held-in settings (e.g., train/easy → valid-unseen/easy in ALFWorld), but also for held-out settings (e.g., train/easy → valid-unseen/hard), including task types not seen during training (e.g., cooling an object instead of placing it somewhere).

Motivated by this observation, we evaluate prolonged fine-tuning for up to 50 epochs and report

performance at multiple checkpoints, rather than selecting models solely based on validation-loss minima. Detailed results are provided in Appendix D. We find that out-of-distribution performance often continues improving well beyond 10 epochs and, in several settings, peaks at epoch 50 despite validation loss increasing from its early minimum. Overall, longer training frequently improves generalization, including on held-out task types.

Relation to prior work. This pattern is consistent with delayed generalization reported in the *grokking* literature, where generalization can emerge only after extended training beyond apparent overfitting (Wang et al., 2024a; Liu et al., 2022; Kumar et al., 2024). Proposed explanations for related phenomena include implicit regularization and optimization effects (Blanc et al., 2020; Damian et al., 2021). We therefore characterize our observations as emergent delayed downstream generalization in agent fine-tuning, while emphasizing that we do not claim a mechanistic explanation in this setting. Our contribution is empirical: prolonged training is an important yet under-emphasized factor for robust generalization in LLM-based agents. Deeper causal analysis is left to future work.

4.4.2 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.

Considering our interest in robust generalization, and following the generalization-dynamics analysis in 4.4.1, we adopt a fixed training budget per environment for the main comparisons: 9 epochs for ALFWorld and 29 epochs for ScienceWorld, which provides a practical balance between performance, training cost, and robustness across setups. However, we encourage practitioners to consider longer runs when maximizing out-of-distribution performance is the primary goal.

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**	4.7
Qwen 2.5-7B				
ExpRAG	81.6	81.9	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	7.8
LoRA (no ExpRAG)	**98.6**	23.0	43.5	12.5
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). The best method for each model and test set appears in bold. All retrieval settings use matched index.

4.4.3 Retrieval-augmented fine-tuning enables robust task generalization

We report fine-tuning results in Table 3.

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.

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. Applying ExpRAG at

inference time on the LoRA-trained model consistently improves over bare LoRA, but ExpRAG-LoRA is the strongest method in most settings (except for Gemma 3-4B on ScienceWorld hard).

Upshot. Using retrieval-augmentation already during training allows the model to natively handle retrieved trajectories in-context and generalize to new, out-of-distribution tasks. This comes at the cost of an increased training time due to the longer context, but keeps in-distribution performance in the same high-success range. These findings are consistent across benchmarks and models.

4.4.4 Robustness to Lack of Relevant Trajectories

Index Condition	Method	ALFWorld Hard tasks (oo-d)	ScienceWorld Hard tasks (oo-d)
Ministral 3-8B			
Empty	ExpRAG	47.5→0.0	18.8→3.1
	LoRA	67.2→31.2	28.1→**15.6**
	ExpRAG-LoRA	88.5→29.5	42.2→1.6
Mismatched	ExpRAG	47.5→9.8	18.8 →7.8
	LoRA	67.2→29.5	28.1→12.5
	ExpRAG-LoRA	88.5→**39.3**	42.2→6.3
Gemma 3-4B			
Empty	ExpRAG	4.9→0.0	6.3→4.7
	LoRA	3.3→1.6	7.8→6.3
	ExpRAG-LoRA	73.8→4.9	4.7→0.0
Mismatched	ExpRAG	4.9→4.9	6.3→1.6
	LoRA	3.3→4.9	7.8→**7.8**
	ExpRAG-LoRA	73.8→**9.8**	4.7→3.1
Qwen 2.5-7B			
Empty	ExpRAG	68.9→22.9	6.3→3.1
	LoRA	70.5→21.3	23.4→7.8
	ExpRAG-LoRA	90.2→34.4	29.7→4.7
Mismatched	ExpRAG	68.9→54.1	6.3→0.0
	LoRA	70.5→24.6	23.4→**17.2**
	ExpRAG-LoRA	90.2→**50.8**	29.7→**17.2**
Qwen 2.5-7B-1M			
Empty	ExpRAG	54.1→3.3	7.8→3.1
	LoRA	68.9→23.0	12.5→**12.5**
	ExpRAG-LoRA	91.8→36.1	29.7→4.7
Mismatched	ExpRAG	54.1→29.5	7.8→0.0
	LoRA	68.9→21.3	12.5→6.3
	ExpRAG-LoRA	91.8→**60.7**	29.7→3.1

Table 4: **Robustness to lack of relevant target trajectories on unseen (hard) tasks across different model backbones.** Each cell reports the success-rate drop induced by the index stress test as ($a \rightarrow b$), where (a) uses trajectories from the matched unseen hard-task index and (b) replaces that index with either **Empty** (no trajectories) or **Mismatched** (training-index trajectories reused at test time). **Bold** indicates the most robust method in each scenario.

We now study what happens when relevant target trajectories are unavailable for the new task that the agent must solve. In Table 4, we report the drop in performance induced by removing access to matched retrieval trajectories when generalizing to out-of-distribution tasks.

Leaving the index empty. With an empty index, ExpRAG-LoRA drops the most and often underperforms standard LoRA. This is expected, since training with retrieval-augmented context but removing it at inference creates a distribution shift.

Keeping the training index. Another stress test is to keep the training index (in our case, easy-task trajectories) at inference time instead of using a target index (built from hard tasks). Under this index mismatch, ExpRAG-LoRA degrades less than with an empty index because the input context remains closer to the training distribution, and cross-task trajectories can still provide useful cues. On ALFWorld, mismatched-index ExpRAG-LoRA remains the strongest method on out-of-distribution tasks. On ScienceWorld, where tasks require more specific behaviors, performance of all methods still largely collapses.

Upshot. Our results indicate that training on retrieval-augmented data may be beneficial, even when no demonstrations can be gathered on future, out-of-distribution tasks. However, in this case, it is preferable to keep the training index rather than completely removing access to any demonstration.

5 Conclusion

We investigated how to integrate experience retrieval into the training and inference pipeline of LLM agents, with the goal of improving robustness and generalization to unseen tasks within a fixed environment. Across ALFWorld and ScienceWorld, we show that inference-only experience retrieval (ExpRAG) already provides large gains over a no-retrieval baseline, and that the largest improvements on unseen tasks come from training with retrieval enabled. In particular, when training on a subset of easy tasks, retrieval-augmented fine-tuning (ExpRAG-LoRA) consistently generalizes to held-out hard tasks at inference time, avoiding the out-of-distribution collapse observed with standard LoRA fine-tuning. Overall, our results suggest that a simple, read-only episodic RAG pipeline constitutes a strong and competitive baseline for agent memory: before resorting to increasingly complex memory architectures or heavily optimized RL-based training pipelines, the community should benchmark against robust retrieval-augmented baselines to make progress traceable to design choices and comparisons meaningful.

Limitations

Our study has some limitations. First, throughout our experiments, the retrieval index is built from scripted expert trajectories generated by the environment-provided policy. Although these trajectories may include failures, such failures do not necessarily reflect the error patterns of LLM agents, and therefore do not measure how well retrieval transfers to realistic, self-induced mistakes. We adopt this setup as a controlled simplification to isolate the effect of retrieval-augmented adaptation, given that the underlying backbones perform poorly in the zero-shot setting. Future work should evaluate robustness under noisier memory sources, such as LLM-generated rollouts of varying quality, and explore self-evolving settings in which agents repeatedly attempt related tasks while learning in-context from their own past successes and failures. Second, performance remains dependent on trajectory availability and coverage: when task-relevant trajectories are missing, generalization to hard tasks can degrade sharply, even if most models still outperform zero-shot baselines. This suggests the need for more principled strategies for data collection, memory construction, and retrieval. Finally, our approach uses a fixed, read-only episodic memory. Exploring read–write memories with consolidation mechanisms (e.g., summarization, abstraction, and selective retention) is a promising direction to improve scalability and long-horizon adaptation.

References

Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. 2020. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In Proceedings of Thirty Third Conference on Learning Theory, volume 125 of Proceedings of Machine Learning Research, pages 483–513. PMLR.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Karen Simonyan, Chris Jones, Geoffrey Irving, and Oriol Vinyals. 2022. Improving language models by retrieving from trillions of tokens. arXiv preprint arXiv:2112.04426.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. FireAct: 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. 2024. 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.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. Preprint, arXiv:2504.19413.

Alex Damian, Tengyu Ma, and Jason Lee. 2021. Label noise sgd provably prefers flat global minimizers. In Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS '21, Red Hook, NY, USA. Curran Associates Inc.

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 The Thirty-ninth Annual Conference on Neural Information Processing Systems.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, 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.

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, Sercan 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.

Tanishq Kumar, Blake Bordelon, Samuel J. Gershman, and Cengiz Pehlevan. 2024. Grokking as the transition from lazy to rich training dynamics. Preprint, arXiv:2310.06110.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and Sebastian Riedel. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems (NeurIPS).

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.

Ziming Liu, Ouail Kitouni, Niklas Nolte, Eric J. Michaud, Max Tegmark, and Mike Williams. 2022. Towards understanding grokking: an effective theory of representation learning. In Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

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.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2024. Qwen2.5 Technical Report. Preprint, arXiv:2412.15115.

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.

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.

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.

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.

torchtune maintainers and contributors. 2024. torchtune: Pytorch's finetuning library.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. 2024a. Grokking of Implicit Reasoning in Transformers: A Mechanistic Journey to the Edge of Generalization. In Advances in Neural Information Processing Systems, volume 37, pages 95238–95265. Curran Associates, Inc.

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

Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. 2024c. 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 Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. 2025. 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. SkillRL: 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 The Thirty-ninth Annual Conference on Neural Information Processing Systems.

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. 2025a. 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. 2024. 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. 2025b. Quantifying and Improving the Robustness of Retrieval-Augmented Language Models Against Spurious Features in Grounding Data. Preprint, arXiv:2503.05587.

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. Preprint, arXiv:2601.01885.

Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng Yan. 2025a. G-Memory: Tracing Hierarchical Memory for Multi-Agent Systems. In The Thirty-ninth Annual Conference on Neural Information Processing Systems.

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. 2025b. Agent Learning via Early Experience. arXiv preprint arXiv:2510.0855.

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. 2025c. 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. 2025d. A Survey on the Memory Mechanism of Large Language Model-based Agents. ACM Trans. Inf. Syst., 43(6).

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 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: the goal is policy generalization and robustness through episodic context reuse, whereas the latter targets knowledge acquisition and synthesis for question answering.

B Implementation details

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

B.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`[1] model (Zhang et al., 2025c) 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* library[2] (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 C.4 compares this default against *agentic JSON*, *compact JSON*, and *textual* alternatives.

[1]Available at:https://huggingface.co/Qwen/Qwen3-Embedding-0.6B

[2]More information: https://sbert.net/

B.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 5 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 list of variations in Table 6. We use this subset in all tables except Table 1, where we compare with related work.

B.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 2), unless a formatting ablation explicitly states otherwise.

B.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.

B.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 Hub[3] are:

- **Ministral 3-8B** (Liu et al., 2026a): `mistralai/Ministral-3-8B-Instruct-2512-BF16`
- **Gemma 3-4B** (Team et al., 2025): `google/gemma-3-4b-it`

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

	ALFWorld		ScienceWorld	
Split	Samples	Tasks (*task-types*)	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 5: 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.

- **Qwen 2.5-7B-1M** (Yang et al., 2025a): `Qwen/Qwen2.5-7B-Instruct-1M`
- **Qwen 2.5-7B** (Qwen et al., 2024): `Qwen/Qwen2.5-7B-Instruct`
- **Qwen 2.5-3B** (Qwen et al., 2024): `Qwen/Qwen2.5-3B-Instruct`

We include both Qwen2.5 and Qwen2.5-1M at matched parameter scales. Qwen2.5-1M is not merely a longer-context setting of Qwen2.5: 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., 2025a). Consequently, comparisons between Qwen2.5 and Qwen2.5-1M reflect both increased context capacity and the effects of long-context-specific training and alignment.

B.6 Hyperparameters

Table 7 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.

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

Task	Task variations
boil	29, 27, 25, 23, 21
changethestateofmatterof	29, 23, 28, 27, 26
chemistrymix	26, 31, 24, 29, 25
chemistrymixpaintsecondarycolor	34, 27, 31, 33, 30
chemistrymixpainttertiarycolor	30, 33, 35, 28, 29
findanimal	248, 260, 294, 271, 299
findlivingthing	257, 256, 277, 266, 268
findnonlivingthing	280, 261, 233, 299, 253
findplant	251, 231, 227, 254, 236
freeze	26, 23, 28, 21, 29
growfruit	123, 114, 108, 105, 96
growplant	95, 94, 100, 112, 116
identifylifestages1	12, 9, 11, 13, 10
identifylifestages2	6, 7, 8, 9
inclinedplanedetermineangle	151, 147, 150, 152, 148
inclinedplanefrictionnamedsurfaces	1356, 1327, 1322, 1366, 1360
inclinedplanefrictionunnamedsurfaces	146, 158, 145, 157, 156
lifespanlongestlived	112, 106, 107, 111, 93
lifespanlongestlivedthenshortestlived	110, 114, 99, 101, 115
lifespanshortestlived	124, 94, 95, 121, 112
measuremeltingpointknownsubstance	355, 400, 422, 413, 403
measuremeltingpointunknownsubstance	266, 263, 243, 281, 265
melt	27, 28, 29, 26, 23
mendeliangeneticsknownplant	94, 105, 100, 93, 91
mendeliangeneticsunknownplant	413, 429, 406, 467, 442
powercomponent	19, 17, 18, 15, 16
powercomponentrenewablevsnonrenewableenergy	19, 16, 17, 15, 18
testconductivity	858, 726, 839, 878, 844
testconductivityofunknownsubstances	538, 475, 494, 464, 532
usethermometer	534, 476, 450, 473, 510

Table 6: Task variations used in our subsampled ScienceWorld test set.

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 8. If the model fails to solve the task within the budget, we consider the episode as a failure.

C Extra experiments on ExpRAG without Training

C.1 Main Results Stability and Reliability Across Seeds

Tables 9 and 10 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-breaking[4], 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

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

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

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

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.

C.2 Validating Main Results across Different Models

Tables 11, 12, 13, and 14 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 ALFWorld/ScienceWorld all-task success. In the models reported here in this appendix, the best retrieval setting improves 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 22.95/2.01 to 88.52/12.75 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.

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 (81.97% $\rightarrow$ 62.30%). 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-

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

Table 9: **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 ± 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.

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, Gemma 3 4B (Table 11) and Qwen 2.5 3B (Table 12) both benefit from retrieval but remain substantially weaker and noisier than Ministral 3-8B 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 88.52%.

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

Tables 13, 14, and 15 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 13) and Qwen 2.5 7B 1M (Table 14) appears already in the No RAG baseline on ALFWorld: 22.95% 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 15. 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

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 10: **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.

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		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		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		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 11: **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.

large, especially in the K=4 block. For example, static retrieval with the `all` index moves from a −17.73 all-task gap in No RAG to +2.92 at K=1, −19.28 at K=2, and then −7.18 at K=4, while the hard-index K=4 rows contain several green entries on easy and hard tasks. We therefore 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.

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 12: **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	29.85	22.95	1.18	3.12	2.01
static	1	all	73.97	67.16	59.02	**17.65**	4.69	**12.08**
		easy	71.23	52.99	31.15	**16.47**	**3.12**	**10.74**
		hard	64.38	63.43	62.30	**7.06**	**7.81**	**7.38**
dynamic		all	72.60 ↓ 1.4	69.40 ↑ 2.2	65.57 ↑ 6.6	11.76 ↓ 5.9	7.81 ↑ 3.1	10.07 ↓ 2.0
		easy	73.97 ↑ 2.7	55.97 ↑ 3.0	34.43 ↑ 3.3	**12.94** ↓ 3.5	**3.12** 0.0	**8.72** ↓ 2.0
		hard	**64.38** 0.0	**63.43** 0.0	62.30 0.0	**3.53** ↓ 3.5	7.81 0.0	**5.37** ↓ 2.0
static	2	all	**83.56**	82.84	81.97	12.94	**10.94**	**12.08**
		easy	71.23	**63.43**	**54.10**	**16.47**	0.00	9.40
		hard	**73.97**	**71.64**	68.85	4.71	6.25	5.37
dynamic		all	**82.19** ↓ 1.4	73.13 ↓ 9.7	62.30 ↓ 19.7	8.24 ↓ 4.7	7.81 ↓ 3.1	8.05 ↓ 4.0
static	4	all	79.45	**83.58**	**88.52**	12.94	9.38	11.41
		easy	**78.08**	61.94	42.62	9.41	0.00	5.37
		hard	47.95	62.69	**80.33**	1.18	**7.81**	4.03
dynamic		all	76.71 ↓ 2.7	**76.87** ↓ 6.7	**77.05** ↓ 11.5	**14.12** ↑ 1.2	**10.94** ↑ 1.6	**12.75** ↑ 1.3
		easy	**87.67** ↑ 9.6	**67.91** ↑ 6.0	**44.26** ↑ 1.6	**12.94** ↑ 3.5	0.00 0.0	7.38 ↑ 2.0
		hard	56.16 ↑ 8.2	61.19 ↓ 1.5	**67.21** ↓ 13.1	2.35 ↑ 1.2	**9.38** ↑ 1.6	**5.37** ↑ 1.3

Table 13: **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.

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 15 becomes 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 14 are concentrated

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		all	63.01 $\downarrow$ 1.4	57.38 $\downarrow$ 1.6	60.45 $\downarrow$ 1.5	12.94 $\downarrow$ 2.4	0.00 $\downarrow$ 6.2	7.38 $\downarrow$ 4.0
		easy	57.53 $\uparrow$ 1.4	16.39 $\uparrow$ 1.6	38.81 $\uparrow$ 1.5	**12.94** $\downarrow$ 4.7	0.00 0.0	7.38 $\downarrow$ 2.7
		hard	45.21 $\uparrow$ 5.5	54.10 $\downarrow$ 4.9	49.25 $\uparrow$ 0.7	12.94 $\uparrow$ 5.9	**6.25** 0.0	10.07 $\uparrow$ 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	8.50
dynamic		all	63.01 $\downarrow$ 8.2	49.18 $\downarrow$ 3.3	56.72 $\downarrow$ 6.0	16.47 $\downarrow$ 3.5	14.06 0.0	15.44 $\downarrow$ 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		all	**75.34** $\downarrow$ 5.5	**60.66** $\downarrow$ 21.3	**68.66** $\downarrow$ 12.7	**23.53** $\uparrow$ 7.1	**32.81** 0.0	**27.52** $\uparrow$ 4.0
		easy	**79.45** $\uparrow$ 9.6	**21.31** $\downarrow$ 13.1	**52.99** $\downarrow$ 0.7	9.41 $\downarrow$ 8.2	**28.12** $\uparrow$ 28.1	**17.45** $\uparrow$ 7.4
		hard	**61.64** $\downarrow$ 1.4	**62.30** $\downarrow$ 9.8	**61.94** $\downarrow$ 5.2	**25.88** $\uparrow$ 14.1	0.00 $\downarrow$ 25.0	**14.77** $\downarrow$ 2.7

Table 14: **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: $\uparrow$ indicates improvement, $\downarrow$ indicates a decline, and black values denote ties.

at K=4, whereas Table 13 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.

C.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 5). 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 2 provides examples of different trajectory formatting. Please note that once the retrieved trajectory is formatted, it is appended 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 16 compares the impact of trajectory formatting on final ExpRAG performance. Results indicate that the effect of trajectory formatting is

ExpRAG type	Top-K	Index	ALFWorld			ScienceWorld		
			Easy tasks	Hard tasks	All tasks	Easy tasks	Hard tasks	All tasks
No RAG	0	–	-28.77	-26.57	-17.73	+2.35	0.00	+1.35
static	1	all	-9.59	-8.14	+2.92	-2.36	+1.56	-0.67
		easy	-15.07	-38.24	+6.16	+1.18	-3.12	-0.67
		hard	-24.65	-4.41	-13.79	0.00	-1.56	-0.67
dynamic		all	-9.59	-12.02	-5.12	+1.18	-7.81	-2.69
		easy	-16.44	-39.58	+4.38	0.00	-3.12	-1.34
		hard	-19.17	-9.33	-13.05	+9.41	-1.56	+4.70
static	2	all	-12.33	-30.38	-19.28	+7.06	+3.12	+5.37
		easy	-4.11	-33.92	-4.10	+3.53	0.00	+2.01
		hard	-26.02	-17.54	-18.10	+3.53	+1.56	+3.13
dynamic		all	-19.18	-23.95	-5.58	+8.23	+6.25	+7.39
static	4	all	+1.37	-1.61	-7.18	+3.53	+23.43	+12.08
		easy	-8.22	-27.51	+11.11	+8.24	0.00	+4.70
		hard	+15.06	+9.44	-13.17	+10.58	+17.19	+13.42
dynamic		all	-1.37	-16.21	-8.39	+9.41	+21.87	+14.77
		easy	-8.22	-46.60	+8.73	-3.53	+28.12	+10.07
		hard	+5.48	+1.11	-5.27	+23.53	-9.38	+9.40

Table 15: **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.

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 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.

D Agents keep improving when trained longer.

In this appendix, we provide the full training dynamics underlying the observation in Section 4.4.1: downstream agent performance can keep improving for many epochs even after the validation loss has reached its minimum and started increasing.

Experimental setting. We fine-tune 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 B.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 7). 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.

chat JSON

```
{
  "role": "assistant",
  "content": "look"
},
{
  "role": "user",
   "content": "You are in the middle of a room. Looking quickly around you, you see nothing."
},
```

agentic JSON

```
{
"role": "action",
 "content": "look"
},
{
 "role": "observation",
  "content": "You are in the middle of a room. Looking quickly around you, you see nothing."
},
```

compact JSON

```
{
 "action": "look"
},
{
 "observation": "You are in the middle of a room. Looking quickly around you, you see nothing."
},
```

textual

```
Assistant: look
User: You are in the middle of a room. Looking quickly around you, you see nothing.
```

Figure 2: Different trajectory formats: *chat JSON*, *agentic JSON*, *compact JSON*, and *textual*.

Results on ALFWorld. Figure 3 shows that validation loss typically bottoms out within the first few epochs and then increases steadily, while both in-distribution performance (Figures 3a and 3b) and out-of-distribution performance (Figures 3c and 3d) 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.

Results on ScienceWorld. We observe the same qualitative pattern in ScienceWorld (Figure 4). Despite validation loss increasing after its early minimum, success on both easy (Figures 4a and 4b) and hard tasks (Figures 4c and 4d) 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.

E 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 6 and 7), 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-

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
Top-1 trajectories								
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
Top-2 trajectories								
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
Top-4 trajectories								
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
Top-5 trajectories								
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

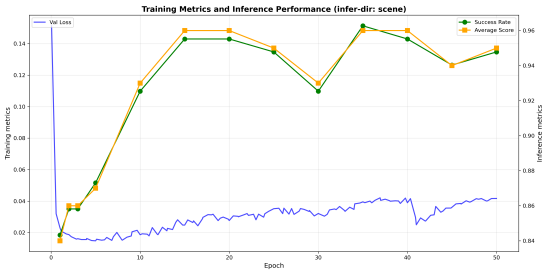
Table 16: Impact of trajectory formatting on ExpRAG performance (all tasks, official valid unseen split)

truth information leakage. In early explorations in our zero-shot setting, we found this stricter setup to reduce backbone performance by approximately 15%.

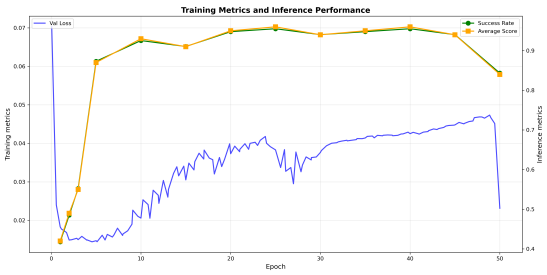
Figure 6 shows the full ALFWorld system prompt. It includes the complete command inventory and formatting constraints (including one-command output), plus guidance for handling invalid actions (e.g., "Nothing happens"). Figure 7 shows the analogous ScienceWorld prompt, with its environment-specific action grammar and explicit command-format constraints (e.g., `pick up <OBJ>` and no extra prefix text).

Finally, Figure 5 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.

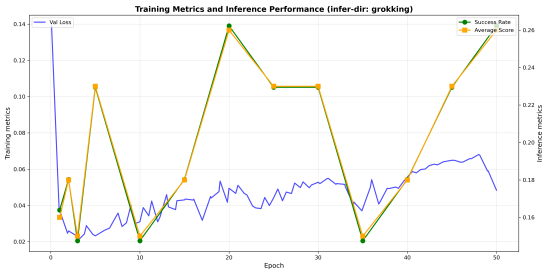
As a final note, when using retrieval, we append the retrieved trajectories to the system prompt as a memory block, as described in Section B.3.



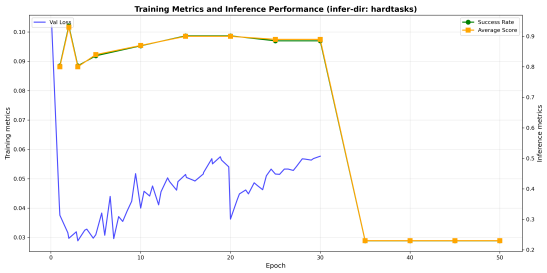
(a) LoRA (no retrieval), ind



(b) ExpRAG-LoRA (matched index), ind

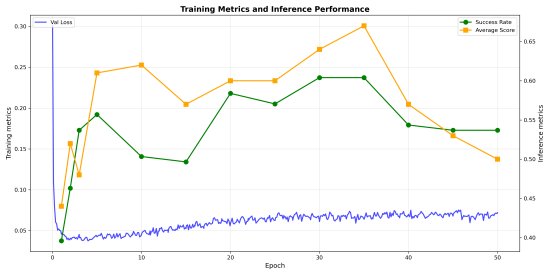


(c) LoRA (no retrieval), ood

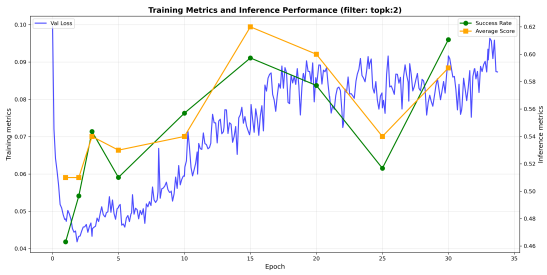


(d) ExpRAG-LoRA (matched index), ood

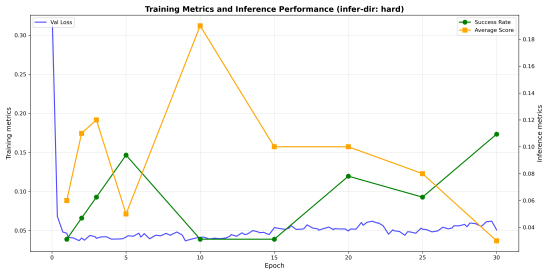
Figure 3: **Longer fine-tuning can improve generalization despite rising validation loss.** Comparison of validation loss and inference performance with respect to number oftraining epochs on ALFWorld. Blue: validation cross-entropy (left axis; lower is better). Green/orange: rollout success rate and 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.



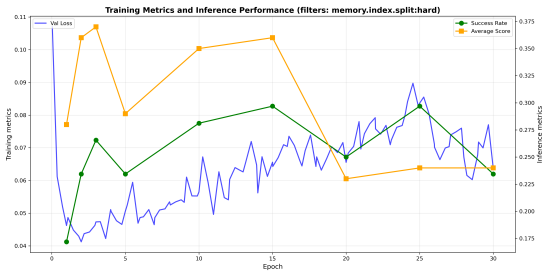
(a) LoRA (no retrieval), ind



(b) ExpRAG-LoRA (matched index), ind



(c) LoRA (no retrieval), ood



(d) ExpRAG-LoRA (matched index), ood

Figure 4: **Longer fine-tuning can improve generalization despite rising validation loss.** Comparison of validation loss and inference performance with respect to number oftraining epochs on ScienceWorld. 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.

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 5: Example of a partial JSON trajectory in ALFWorld.

System prompt for ALFWorld

```
Interact with a household to solve a task. Here are 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 6: System prompt used for ALFWorld experiments.

F Efficiency Cost of ExpRAG

Table 17 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 7: 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 17: Model: Ministral 3-8B, dataset: ALFWorld. Efficiency of different ExpRAGvariants 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.