Sample-Efficient Online Learning in LM Agents via Hindsight Trajectory Rewriting

Michael Y. Hu[1*] **Benjamin Van Durme**[2] **Jacob Andreas**[2] **Harsh Jhamtani**[2]
[1]New York University [2]Microsoft
michael.hu@nyu.edu, hjhamtani@microsoft.com

Abstract

Language model (LM) agents deployed in novel environments often exhibit poor sample efficiency when learning from sequential interactions. This significantly hinders the usefulness of such agents in environments where interaction is costly (for example, when they interact with humans or reset physical systems). While a number of existing LM agent architectures incorporate various mechanisms for experience storage and reflection, they make limited use of LMs' abilities to directly generate or reason about full counterfactual trajectories. We introduce ECHO (Experience Consolidation via Hindsight Optimization), a prompting framework that adapts hindsight experience replay from reinforcement learning for language model agents. ECHO generates optimized trajectories for alternative goals that could have been achieved during failed attempts, effectively creating synthetic positive examples from unsuccessful interactions. Our approach consists of two components: a hindsight rule that uses the language model itself to identify relevant subgoals and generate optimized trajectories, and an update rule that maintains compressed trajectory representations in memory. We evaluate ECHO on stateful versions of XMiniGrid, a text-based navigation and planning benchmark, and PeopleJoinQA, a collaborative information-gathering enterprise simulation. Across both domains, ECHO outperforms vanilla language agent baselines by up to 80%; in XMiniGrid, it also outperforms a number of sophisticated agent architectures including Reflexion and AWM, demonstrating faster adaptation to novel environments through more effective utilization of past experiences.

1 Introduction

While language models (LMs) have demonstrated remarkable generalization across tasks, their performance often degrades in unfamiliar or interactive environments, especially when learning from limited experience (Ziems et al., 2024; Kwa et al., 2025; Liang et al., 2023). In such settings, sample efficiency becomes critical, particularly when interactions are costly (e.g., with humans or physical systems). For example, a conversational assistant deployed for the first time in a new organization likely does not know where to look for specific pieces of information, or the best means of communicating with specific people. Thus, creating agents that can learn and adapt to their environments over time is of critical importance in improving their everyday usability.

Here we study the problem of building efficient mechanisms for *online learning* in LM agents. We consider the setting where an LM agent receives queries one at a time in a streaming fashion. Existing LM agent frameworks typically approach this setting through reflection (Shinn et al., 2023; Zhao et al., 2024), memory (Wang et al., 2025b), or experience replay mechanisms (Zheng et al., 2024), which allow agents to revisit past episodes and improve over time. However, these methods primarily focus on storing or synthesizing experiences, and thus fail to fully exploit the LM's ability to reason about

[*]Work done while interning at Microsoft. Code: https://github.com/michahu/echo

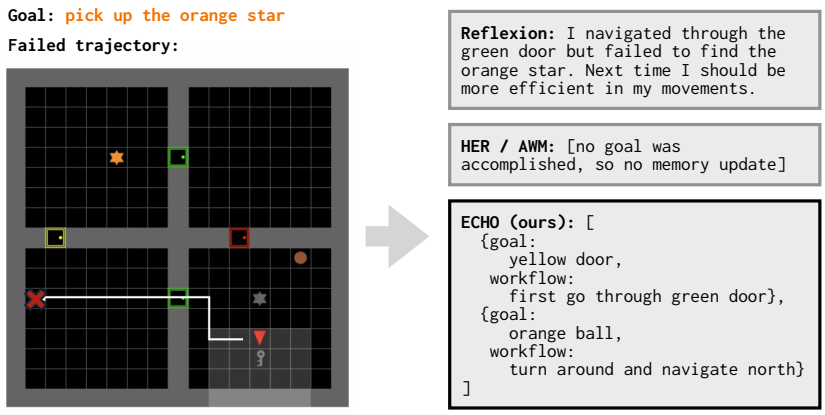

Figure 1: ECHO in the XMiniGrid environment. The agent fails in its first trajectory (left). Using this trajectory, ECHO identifies other objects the agent could have reached, and generates an optimized trajectory for these goals (right). In future iterations, the agent can then use these optimized trajectories to successfully complete unseen goals.

counterfactuals—what could have led to success in past failures. This gap suggests an opportunity to design LM agents that actively rewrite and optimize their past experiences, converting failures into synthetic successes that improve future decision-making.

In this work, we introduce ECHO (Experience Consolidation via Hindsight Optimization), a framework that adapts hindsight experience replay (HER) to LM agents, enabling them to generate and learn from counterfactual trajectories for more sample-efficient learning. Our approach builds on **hindsight experience replay** methods from the RL literature (Andrychowicz et al., 2017, HER). HER learns a goal-conditioned policy; during training, each attempt to reach a goal state s that fails in an end state s' is interpreted as a successful trajectory for reaching s'. For example, a trajectory in which an LM fails to slice an apple by attempting to grab a knife, dropping it, then grasping an apple may still be interpreted as a demonstration of a successful grasp. But HER and related methods are relatively limited in the set of trajectory modifications they can make—relabeling trajectories with goals, but not altering the structure demonstrated trajectories themselves.

ECHO is a significantly more expressive hindsight relabeling method for LMs. In ECHO, LMs can perform arbitrary re-writing of failed trajectories, including changing both their goals and their intermediate steps. In the running example, this procedure might not only relabel the failed slicing attempt as a successful grasp, but edit out the knife-grasping attempt that was relevant to the initial goal but not the relabeled one.

We test ECHO and various state-of-the-art agent architectures in stateful variants of a 2D GridWorld task (XMiniGrid, Nikulin et al. (2023)) and a question-answering task with multiple agents and tool calling (PeopleJoinQA, Jhamtani et al. (2025)). These environments require exploration to successfully solve all queries of the task. In XMiniGrid, the agent must explore to find various objects in different rooms, and in PeopleJoin, the agent has imprecise or incomplete information about which teammates have the information it needs to answer a question. We make these environments stateful by allowing agents to persist insights via a scratchpad memory. Next, we reset the environment to the same initial position or state and vary the queries or tasks posed to the agent. The agent can then infer information about the environment over time and improve its performance and efficiency.

On XMiniGrid-Stateful, ECHO outperforms the baseline reason-then-act (ReAct) LM agent (Yao et al., 2023) by 80% in average reward and the next best baseline by 42%. On PeopleJoinQA-Stateful, ECHO still outperforms the standard ReAct agent on both accuracy and efficiency, while being slightly worse than the best baseline in accuracy by 4.6% and tied in efficiency. We conclude that ECHO is a promising technique for improving the sample efficiency of language agents, especially in environments where rewards are sparse and the baseline language agent performs poorly.

2 Related Work

Language Model Agents: Language model (LM) agents are systems that use large language models as reasoning engines to interact with environments, make decisions, and execute actions over time (Yao et al., 2023; Schick et al., 2023). These agents typically operate through a perception–action loop, where they observe their environment, reason about the current state, and generate actions (Wang et al., 2023). Recent work has demonstrated LM agents' capabilities across diverse domains, from web navigation and tool use to multi-agent collaboration and code generation (Liu et al., 2024; Qian et al., 2023; Jhamtani et al., 2025). However, a key limitation of current LM agents is their reliance on static knowledge encoded during pre-training, making them less effective when deployed in novel environments that require exploration and adaptation (Zhou et al., 2024). This motivates the need for agents that can accumulate experience and improve their performance through interaction with their environment over time.

Offline Reasoning and Memory Following Sumers et al. (2024), we categorize memory systems for LM agents into two types: semantic and episodic. Semantic memory contains facts about the environment, and episodic memory stores past actions. In this work, we consider two baselines, Reflexion and Agent Workflow Memory (AWM), as exemplars of manipulating semantic and episodic memory (Shinn et al., 2023; Wang et al., 2025b). Reflexion instructs the language model to reflect on the previous trajectory and propose areas of improvement; we consider these high-level notes about the environment to be part of semantic memory. AWM instructs the model to generate a summary workflow of the trajectory, provided the trajectory is successful; we consider this to be episodic memory. Building on this work, the current paper develops an improved mechanism for constructing and updating episodic memories.

Experience Replay One reason why off-policy RL algorithms can be more efficient than on-policy ones is that they can store and learn from informative trajectories that are low probability under the current policy, or even trajectories that were never observed at all. Such *experience replay* techniques have proven especially valuable in situations with sparse rewards or limited data, as they extract maximal learning signal from a small number of positive examples (Schaul et al., 2016; Lu et al., 2023; Zhang et al., 2025). In particular, hindsight experience replay (HER) further improves sample efficiency by relabeling past trajectories with alternative goals that were actually achieved during execution (Andrychowicz et al., 2017). For instance, if an agent fails to reach a target location but successfully navigates to an intermediate point, HER treats this trajectory as a successful example for reaching that intermediate goal. Butt et al. (2024) apply HER to self-improve language models at writing code; our approach can be viewed as a generalization of HER in which not only goals, but arbitrary aspects of trajectories, can be edited in hindsight.

3 Approach

3.1 ECHO: Experience Consolidation via Hindsight Optimization

We consider an online setting wherein an LM agent processes a sequence of queries from time $t = 0$ to T *without* access to a ground-truth reward function or demonstrations.

Our key insight is that LMs have sufficient world knowledge to propose general edits to a trajectory, in addition to simply relabeling the trajectory for a particular goal. We take inspiration from HER and design a prompting framework that allows language agents to modify their past experiences. We call this framework Experience Consolidation via Hindsight Optimization, or ECHO. The basic idea behind ECHO is to take an existing trajectory and identify not just what goals that trajectory achieves, but all goals for which a successful trajectory can be *synthesized* given the initial rollout.

Listing 1: ECHO pseudocode

```
def ECHO(LM, trajectory, replay_buf={}):
    # hindsight rule
    summary = LM.summarize(trajectory)
    goals = LM.identify_goals(trajectory)
    for goal in goals:
        new_traj = LM.infer_traj(goal,
    trajectory)

        # update rule
        old_traj = replay_buf[goal]
        if old_traj and len(new_traj) <
    len(old_traj):
            replay_buf[goal] = new_traj
    return replay_buf
```

ECHO contains two parts: a hindsight rule and an update rule (Listing 1). During application of the hindsight rule, the LM first proposes goals that it can *infer* how to accomplish from a given trajectory. If no goals are proposed, then ECHO does nothing. Next, the LM generates an optimized trajectory or description from the goal and the original trajectory. The optimized trajectory or description is given in natural language; see Figure 1 or §3.3 for examples.

In the update rule, for each entity, we compare its newly generated descriptor to the descriptor's predecessor and save the shorter workflow. Our motivation here is related to Kolmogorov complexity, or minimum description length (see Grünwald (2007) for an overview); we want the replay buffer to contain the shortest possible description for achieving the goal.

ECHO runs the risk of appending a very short trajectory description early in the sequence of interactions, after which future trajectories will be ineffective. In our experience, this is very rare, because the LM has the option to abstain, or not propose any goals. As such, the goal-trajectory pairs that are added to the replay buffer are valid or near-valid trajectories. Nevertheless, a more precise update rule is a fruitful area for future work (§5.3).

3.2 Baselines

In Listings 2 and 3, we cast Reflexion (Shinn et al., 2023) and AWM (Wang et al., 2025b) into our hindsight-update framework as well. The important parts of these algorithms lie in the hindsight rule, and the update rule simply appends the hindsights edits to memory.

Listing 2: Reflexion pseudocode.

```
def reflexion(LM, trajectory, replay_buf
    =[]):
    # hindsight rule
    reflection = LM.reflect(trajectory)

    # update rule
    replay_buf.append(reflection)
    return replay_buf
```

Listing 3: AWM pseudocode.

```
def AWM(LM, trajectory, replay_buf=[]):
    # hindsight rule
    if LM.is_successful(trajectory):
        workflow = LM.infer_workflow(
    trajectory)

    # update rule
    replay_buf.append(workflow)
    return replay_buf
```

3.3 Qualitative Example

In this section, we show responses from Reflexion, AWM, and ECHO after the same trajectory, in which an agent failed to pick up a grey key.

Reflexion

In this episode, I successfully navigated towards the grey key but failed to pick it up despite multiple attempts. I need to ensure that I correctly execute the 'pick up' action when I am adjacent to the target item. Additionally, I should be more efficient in my movements and avoid unnecessary turns and steps. In future episodes, I will focus on better spatial awareness and precise action execution to achieve my goals more effectively.

AWM

{ "goal": "Pick up grey key.", "workflow": "" }

ECHO

{ "goal": "Pick up the grey star", "workflow": "Step 1: Navigate north from the starting location. Step 2: Move towards the grey star located to the northeast. Step 3: Pick up the grey star." }

Reflexion notes that the agent failed, but its feedback is generic; empirically, this kind of feedback does not change the model's performance within XMiniGrid-Stateful (§4.2). Since the trajectory failed, AWM correctly declines to generate a workflow. Conversely, ECHO successfully notices that the agent also observed the grey star while unsuccessfully navigating towards the grey key, producing a correct optimized trajectory explaining how to pick up the grey star.

4 Results

To test our algorithm, we create stateful versions of two environments where exploration is crucial: XMiniGrid (Nikulin et al., 2023) and PeopleJoinQA (Jhamtani et al., 2025). These corresponding versions, XMiniGrid-Stateful and PeopleJoinQA-Stateful, can be used as benchmarks for efficient online learning in language agents; we release these environments at `https://github.com/michahu/echo`. We ran all experiments using GPT-4o (Hurst et al., 2024); see Appendix B for hyperparameters.

4.1 Evaluation

We refer to the full sequence of states, actions, and rewards encountered as an episode, and a timestep as a single state-action-reward tuple within an episode. The offline algorithms we consider here operate between episodes.

Our evaluation metrics are **final average reward** (or accuracy) and **cumulative average reward**. The cumulative average reward at episode τ is the average of all rewards received up to that episode:

$$\text{Cumulative Average Reward at } \tau = \frac{1}{\tau + 1} \sum_{t=0}^{\tau} R_t$$

where R_t is the reward achieved in episode t. We use this metric to compare agents' sample efficiency, as sample-efficient agents will rapidly increase the cumulative average reward. To normalize for problem difficulty, we report rewards as improvements over a baseline ReAct agent (Yao et al., 2023). Thus, the best method will be the one that maximizes both the final average reward and the rate at which it improves upon the ReAct agent.

4.2 XMiniGrid-Stateful

XMiniGrid is a procedurally-generated GridWorld, where an agent navigates and perform tasks in a partially-observable 2D grid environment. XMiniGrid takes inspiration from XLand, a suite of procedurally-generated, partially-observable 3D games. To create XMiniGrid-Stateful, we prompt the agent to achieve one randomly sampled goal in the same environment per episode and reset the agent and other objects in the environment to the same starting locations between episodes. Thus, the agent can learn the starting locations of unseen objects over time.

In total, we test the language models on 10 unique, procedurally generated environments. Each environment has 4 objects distributed across 4 rooms. Partial observability makes the task challenging, akin to picking up objects in a dimly-lit house. For each environment, we sequentially ask the model to pick up a randomly sampled object 16 times, allowing duplicate queries. For each query, we give the model up to 64 steps to achieve it. Thus, the maximum number of queries required to run XMiniGrid-Stateful is $10 \times 16 \times 64 = 10,240$.

To make XMiniGrid compatible with language models, we convert its 2D observation space to an egocentric text description, which reads something like "You are two steps from a wall. You see a red door two steps to the right." Since even navigation in this partially observed environment is challenging for LMs, we restrict the randomly sampled goals to "pick up" goals. XMiniGrid-Stateful's evaluation metric is mean reward, or success rate of picking up the object, over the 64 steps.

ECHO strictly outperforms all other methods on XMiniGrid (Figure 2) In addition to AWM, we created a baseline called AWM++ which replaces AWM's update rule with our own (keeping the shorter workflow in memory when a goal collision occurs). Performance with this update rule improves slightly, but not enough to recapture the entire improvement from ECHO.

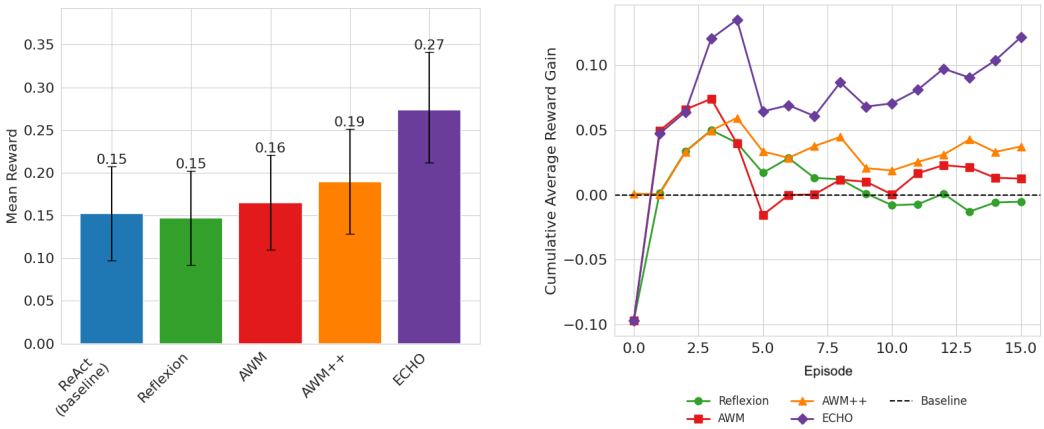

Figure 2: Results on the XMiniGrid-Stateful benchmark. Left: ECHO achieves the highest mean reward. Right: ECHO's cumulative reward is higher than the baseline ReACT agent's after 3 interactions, indicating that ECHO improves compared to a static baseline over time.

Trajectory Validity Analysis: While ECHO leverages the LM's world knowledge to synthesize counterfactual trajectories, these may not always be executable under true environment dynamics. To assess this limitation, we evaluated the validity of the hindsight-imputed workflows generated by ECHO. For each synthesized trajectory–goal pair, we then provided the full hindsight-imputed workflow to the LM agent as part of the system prompt. Across 40 sampled examples from XMiniGrid, the agent successfully reached these imputed goals 85% of the time (34 / 40). 4 of the remaining 6 failures arose from agent deviations from instructions during execution and 2 were due to infeasible steps in the synthesized workflows. This indicates that the counterfactual workflows generated by ECHO in XMiniGrid are largely correct and lead the agent to successful solutions.

4.3 PeopleJoinQA-Stateful

PeopleJoinQA is a question-answering environment where an agent must synthesize (join) information collected from simulated people to answer a question. To contact people and retrieve information, the agent must also use various tools, such as organization directory and document search functions. PeopleJoinQA is also partially observable because the agent does not know ahead of time which people possess the information it is seeking.

To create PeopleJoinQA-Stateful, we simply fix the organization, or the set of simulated people and knowledge, and ask the agent all the questions written for that organization. Between queries, the agent can then reflect on how information is distributed in the organization and improve. We chose 5 organizations within PeopleJoinQA, each with different numbers of people and possible queries. In total, there are 248 queries across the 5 organizations, and each query takes on average 7.98 messages between organization members to resolve, requiring a total of $1,980$ queries to run PeopleJoinQA-Stateful. (In our tests, XMiniGrid-Stateful is still slightly faster to run, due to the agent's observations being shorter).

In Figure 3, the most accurate method for PeopleJoinQA-Stateful is Reflexion, which is notable because in Reflexion the model provides feedback to itself generically. Thus, methods that manipulate episodic memory like AWM or ECHO may not be always be helpful in improving accuracy. However, AWM and ECHO both improve the efficiency of the agent's interactions, decreasing the average number of messages sent between the agent and other people in the organization by around 1.6 messages.

In fact, the most accurate or efficient method varies depending on the PeopleJoin organization, as shown in Figure 4 in the appendix; no method strictly dominates all other methods. For example, Figure 3 (right) shows that ECHO becomes the most efficient method after around 15 total queries in the organization answering questions about department stores. Thus, although the offline methods we consider in this work clearly outperform the baseline in the aggregate, understanding how to improve the robustness of these offline methods for all settings is an important area for future work.

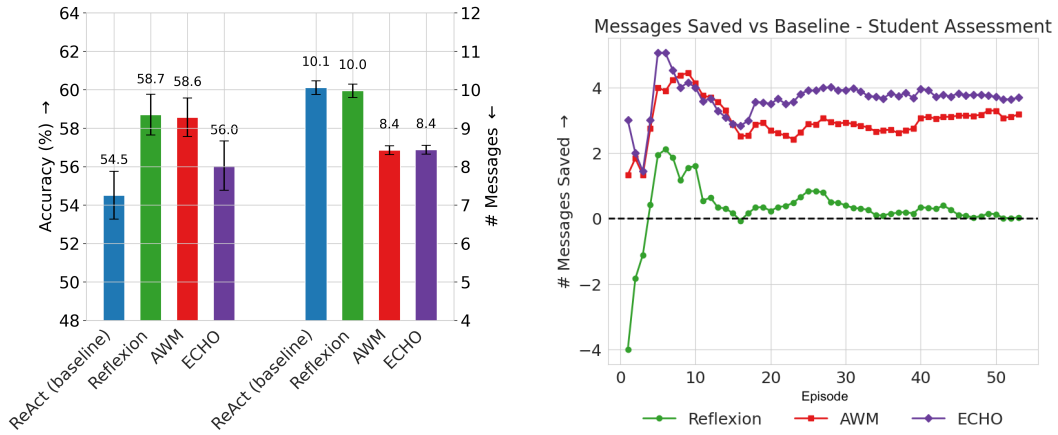

Figure 3: Results on the PeopleJoinQA-Stateful benchmark. Left: While Reflexion achieves slightly higher accuracy, ECHO and AWM are more efficient, completing the task in 1.6 fewer messages on average. Right: we plot the running average reward gain above ReAct. On average, ECHO outperforms the ReAct after the first query.

5 Discussion

5.1 Language Models as Incomplete World Models

After a single trajectory within an environment, the LM often will not have sufficient information to infer a full internal world model of its environment. However, we can use the LM's general knowledge to infer local improvements to its previous trajectory. Hindsight optimization uses the LM's pretrained world knowledge to fill in gaps and propose reasonable counterfactual information, even when the agent's direct experience is limited. By sidestepping the need for a complete world model, ECHO is particularly effective in partially observable environments where building world models would be infeasible.

5.2 Connections between RL and Prompting

ECHO continues the connection identified in ReACT (Yao et al., 2023) and Reflexion (Shinn et al., 2023) between reinforcement learning techniques and prompting strategies for language model agents. Tratitional RL methods like hindsight experience replay rely on numerical rewards and states, whereas language models can operate over experiences that can be described and modified through natural language. This enables a more flexible form of experience replay where the agent actively edits and improves past experiences based on its linguistic and commonsense understanding of the task. ECHO also bridges experience-based learning and symbolic reasoning, leveraging language models as both world models and policy generators.

5.3 Limitations and Future Work

In this work, we primarily considered natural language representations of semantic and episodic information. Recent work shows that code-like representations can be even more effective (Wang et al., 2025a), so future work should explore how ECHO's performance changes when outputting programmatic trajectory representations. Furthermore, our update heuristic of accepting shorter trajectories for the same task can likely be improved in favor of a more sophisticated that combines new and old information while maintaining the same bias towards compression. Future work could also explore augmenting ECHO with retrieval-based mechanisms that draw from a memory bank (Zheng et al. (2024); Moghe et al. (2024)), enabling more targeted reuse of relevant experiences.

6 Conclusion

In this work, we introduced ECHO, a framework that improves LM agent sample efficiency by adapting hindsight experience replay from off-policy RL. Our results demonstrate that agents can effectively learn from past experiences by using themselves as incomplete world models to edit and optimize previous trajectories. To evaluate ECHO and other agents in stateful environments, we introduced two new adaptations of existing benchmarks: XMiniGrid-Stateful and PeopleJoinQA-Stateful, both of which require exploration and multi-step reasoning. By using the language model to propose and refine its own experiences, ECHO provides a path towards more sample-efficient and adaptable LM agents, particularly in partially observable environments with sparse feedback.

References

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

N. Butt, B. Manczak, A. J. Wiggers, C. Rainone, D. W. Zhang, M. Defferrard, and T. Cohen. Codeit: Self-improving language models with prioritized hindsight replay. In *ICML*, 2024. URL https://openreview.net/forum?id=SXVn5IFsrs.

P. D. Grünwald. *The Minimum Description Length Principle*. MIT press, 2007.

A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

H. Jhamtani, J. Andreas, and B. Van Durme. LLM agents for coordinating multi-user information gathering. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17800–17826, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.916. URL https://aclanthology.org/2025.findings-acl.916/.

T. Kwa, B. West, J. Becker, A. Deng, K. Garcia, M. Hasin, S. Jawhar, M. Kinniment, N. Rush, S. V. Arx, R. Bloom, T. Broadley, H. Du, B. Goodrich, N. Jurkovic, L. H. Miles, S. Nix, T. Lin, N. Parikh, D. Rein, L. J. K. Sato, H. Wijk, D. M. Ziegler, E. Barnes, and L. Chan. Measuring ai ability to complete long tasks, 2025. URL https://arxiv.org/abs/2503.14499.

P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, B. Yuan, B. Yan, C. Zhang, C. Cosgrove, C. D. Manning, C. Re, D. Acosta-Navas, D. A. Hudson, E. Zelikman, E. Durmus, F. Ladhak, F. Rong, H. Ren, H. Yao, J. WANG, K. Santhanam, L. Orr, L. Zheng, M. Yuksekgonul, M. Suzgun, N. Kim, N. Guha, N. S. Chatterji, O. Khattab, P. Henderson, Q. Huang, R. A. Chi, S. M. Xie, S. Santurkar, S. Ganguli, T. Hashimoto, T. Icard, T. Zhang, V. Chaudhary, W. Wang, X. Li, Y. Mai, Y. Zhang, and Y. Koreeda. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=iO4LZibEqW. Featured Certification, Expert Certification, Outstanding Certification.

X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

C. Lu, P. J. Ball, Y. W. Teh, and J. Parker-Holder. Synthetic experience replay. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=6jNQ1AY1Uf.

N. Moghe, P. Xia, J. Andreas, J. Eisner, B. Van Durme, and H. Jhamtani. Interpreting user requests in the context of natural language standing instructions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4043–4060, 2024. URL https://aclanthology.org/2024.findings-naacl.255.pdf.

A. Nikulin, V. Kurenkov, I. Zisman, V. Sinii, A. Agarkov, and S. Kolesnikov. XLand-minigrid: Scalable meta-reinforcement learning environments in JAX. In *Intrinsically-Motivated and Open-Ended Learning Workshop, NeurIPS2023*, 2023. URL https://openreview.net/forum?id=xALDC4aHGz.

C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, J. Xu, D. Li, Z. Liu, and M. Sun. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023. URL https://arxiv.org/abs/2307.07924.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.05952.

T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

N. Shinn, F. Cassano, A. Gopinath, K. R. Narasimhan, and S. Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

T. Sumers, S. Yao, K. Narasimhan, and T. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=1i6ZCvflQJ. Survey Certification.

L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J.-R. Wen. A survey on large language model based autonomous agents, 2023.

Z. Z. Wang, A. Gandhi, G. Neubig, and D. Fried. Inducing programmatic skills for agentic tasks, 2025a. URL https://arxiv.org/abs/2504.06821.

Z. Z. Wang, J. Mao, D. Fried, and G. Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=NTAhi2JEEE.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

H. Zhang, J. Fu, J. Zhang, K. Fu, Q. Wang, F. Zhang, and G. Zhou. Rlep: Reinforcement learning with experience replay for llm reasoning, 2025. URL https://arxiv.org/abs/2507.07451.

A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang. Expel: Llm agents are experiential learners. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

L. Zheng, R. Wang, X. Wang, and B. An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Pc8AU1aF5e.

S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried, U. Alon, and G. Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

C. Ziems, W. Held, O. Shaikh, J. Chen, Z. Zhang, and D. Yang. Can large language models transform computational social science? *Computational Linguistics*, 50(1):237–291, Mar. 2024. doi: 10.1162/coli_a_00502. URL https://aclanthology.org/2024.cl-1.8/.

A Prompts

All of the offline proactive reasoning methods we study (Reflexion, AWM, ECHO, and their variants) are paired with a ReAct agent that performs actions.

A.1 XMiniGrid-Stateful

ReACT

You are an agent in a 2D gridworld. At each step you will receive a list of valid and invalid actions. Choose a valid action by its index. Complete the goal in #HORIZON# steps.

You will be prompted at each turn to first reason about your plan and then choose actions.

Reply concisely with following JSON format: {"thought": X, "choice": Y} where X is your reasoning and Y is the index of the desired choice. Ensure Y is a parseable integer!

To help the agent understand which actions are valid or invalid, at each step we provide the dynamic lists `valid_actions={1: "go forward", ...}, invalid_actions={4:"pick up", ...}`.

Reflexion

You are an agent in a 2D text-based environment. Reflect on your performance in the following episode and write some concise notes on how you can improve your performance in the next episodes. Reply with the following JSON format: {"reflection": X} where X is your reflection. Ensure X is a parsable string!

AWM

You are an agent in a 2D text-based environment. If the agent succeeds at accomplishing the given goal in the episode, convert the actions done in the following episode into abstract summary workflow. Discuss in high-level terms the steps a future agent should take to reach the goal. Include potential obstacles and landmarks in your workflow explanation.

Reply with the following JSON format: {"goal": "X", "workflow": Y} where X is the achieved goal and Y is your summary workflow. Ensure X and Y are parsable strings!

If the agent did not achieve the goal, then make Y an empty string.

ECHO has 3 LM calls, which we name according to our pseudocode, reproduced below:

```
def ECHO(LM, trajectory, replay_buf={}):
    # hindsight rule
    summary = LM.summarize(trajectory)
    goals = LM.identify_goals(trajectory)
    for goal in goals:
        new_traj = LM.infer_traj(goal, trajectory)

        # update rule
        old_traj = replay_buf[goal]
        if old_traj and len(new_traj) < len(old_traj):
            replay_buf[goal] = new_traj
    return replay_buf
```

ECHO: LM.summarize

You are an expert at analyzing agent behavior in 2D text-based environments. Create a concise, high-level summary of the agent's trajectory.

Instructions:

What to Include:
- Group low-level actions into high-level behaviors (e.g., "explored northern corridor" not individual moves)
- **All** objects discovered
- Completed objectives

What to Exclude:
- Individual movement steps, redundant actions, minor environmental details

Format: Chronological entries representing distinct phases or achievements

Output Format:
{ "0": "Agent spawned in [location] and observed [key objects/features]", "1": "Agent navigated to [destination] and discovered [important findings]", "2": "Agent interacted with [object/entity] resulting in [outcome]", ... }

ECHO: LM.identify_goals

You are an expert at analyzing 2D text-based environments to identify potential agent objectives. Given a trajectory summary, extract all possible goals an agent could pursue. The agent's goal will always be to pick up a specific object.

Task:
Identify all objects that could serve as pickup targets based on the environmental context shown in the summary.

Requirements:
- **Extract specific objects** mentioned in the trajectory - Avoid locations or non-portable objects
Output Format:
{ "possible_goals": ["Pick up the [object1]", "Pick up the [object2]", ...] }

ECHO: LM.infer_traj

You are an expert at creating action plans for agents in 2D text-based environments. Given a specific goal and a summary of a previous agent's actions, create a high-level workflow to achieve the goal.

Task: Design an abstract workflow for accomplishing the given goal using the environmental features from the trajectory summary.

Requirements:
- **Environment-specific actions only**: reference actual locations, objects, or features from the summary
- Use high-level abstractions (e.g., "navigate to the blue door")
- **Avoid generic phrases** like "move toward goal" or "find the object"
- Start from the agent's known starting location
- Focus on strategic phases, not individual actions

Output Format: { "goal": "[provided goal]", "workflow": "Step 1: [specific environment action]. Step 2: [specific environment action]. Step 3: [etc.]" }

A.2 PeopleJoinQA-Stateful

For the decision policy, we used the prompts from Jhamtani et al. (2025), available here: `https://github.com/microsoft/peoplejoin`.

Reflexion

You are a helpful and clever teacher of language agents. You have access to a prior interaction between a language agent and other agents in an organization, as well as your own reflection about the organization. Using the prior interaction and reflection, write a better reflection that will help a future language agent perform better in this organization.

Structure your reflection in the following json format: {'reflection': reflection}, where reflection is a string. The reflection should be concise and focused on giving instructions to future agents in this organization.

AWM

You are a helpful and clever teacher of language agents. Attached below is a prior interaction between a language agent and other agents in an organization. If you deem the interaction to successfully and accurately answer the initial question, return a summary of the interaction so future agents can easily reference what to do in similar situations. The summary should contain the query, a summary of events, and the final answer.

If the interaction was successful, return a json {'successful': true, 'summary': summary}, where summary is a string. If the interaction was not successful, return a json {'successful': false, 'summary': ''}.

For PeopleJoin, we found summarizing unnecessary. Furthermore, we found that it worked better to ask the model for one optimized trajectory instead of several. Because the number of timesteps in some PeopleJoin environments are long, we observed that adding many trajectories to memory at a time leads to the context length expanding significantly, even with our update rule.

ECHO

You are a helpful and clever teacher of language agents. Given a trajectory, write a simplified counterfactual workflow and final answer. If the trajectory is already efficient, you can simply summarize the events. If the correct final answer is unclear, then do not generate a workflow or final answer.

The counterfactual trajectory should include:
- the query
- a workflow for solving the query
- the final answer

Return a json {'query': query, 'workflow': workflow, 'final_answer': final_answer}. If either the correct workflow or final answer are unclear, then you should not generate a workflow or final answer. To abstain, return empty strings for 'workflow' and 'final_answer': {'query': query, 'workflow': '', 'final_answer': ''.

B Hyperparameters

The GPT-4o hyperparameters we used for the agent itself versus the offline proactive reasoning are slightly different. Below, we have labeled the agent as "ReACT" and the offline reasoning as "Offline."

Table 1: XMiniGrid-Stateful, ReACT

Hyperparameter	Value
Temperature	0
Max New Tokens	4000
API Version	05-13

Table 2: XMiniGrid-Stateful, Offline

Hyperparameter	Value
Temperature	0
Max New Tokens	4000
API Version	05-13

Table 3: PeopleJoin, ReACT

Hyperparameter	Value
Temperature	0
Max New Tokens	3800
API Version	11-20

Table 4: PeopleJoin, Offline

Hyperparameter	Value
Temperature	0.7
Max New Tokens	2000
API Version	11-20

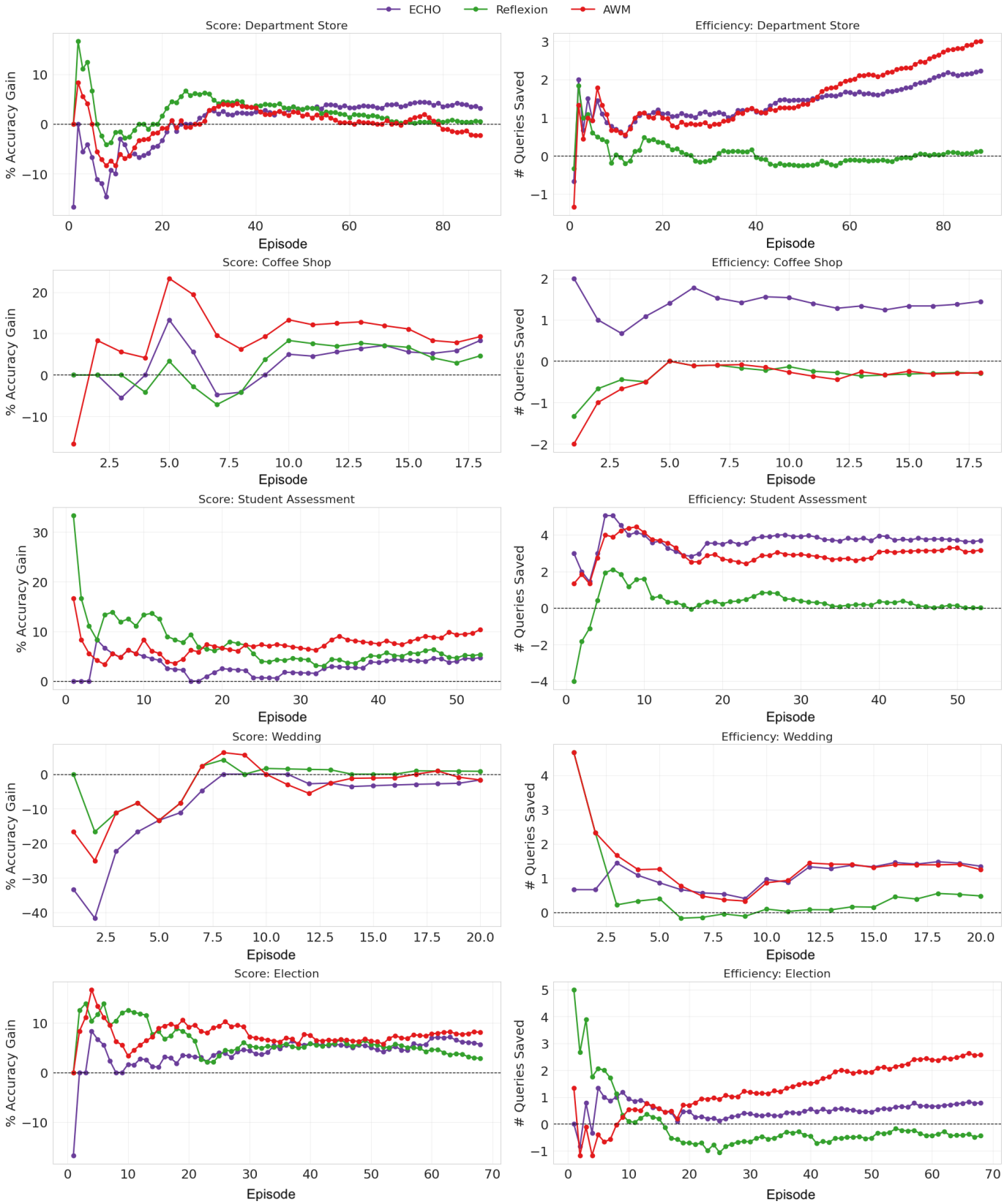

Figure 4: We chose 5 organizations from PeopleJoinQA, maximizing variation amongst number of people in the organization and total number of queries. No offline method consistently outperforms the baseline on both accuracy and efficiency for all organizations.