FROM PLAYBOOKS TO RULEBOOKS: EXTENDING AGENTIC CONTEXT ENGINEERING (ACE) FOR EPISODIC WORKFLOW LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using large language models often repeat workflow failures because prior execution experience is not converted into reusable guidance. This paper introduces a novel application of Agentic Context Engineering (ACE) to a single, stable agentic workflow task, in which deliberately simulated tool-level barrier scenarios are paired with human-crafted ground-truth resolutions to teach an agent how to recover from recurring operational exceptions. Through the Generator–Reflector–Curator training loop, ACE compares failed or incomplete trajectories with expected human-defined outcomes and distills corrective guidance into an *episodic rulebook*: a persistent, inspectable memory artifact containing generalizable exception-handling and recovery strategies for future executions. We evaluate this approach in a controlled, multi-step notification workflow containing happy-path, no-notification, and simulated barrier scenarios, without modifying model weights. The rulebook was learned from 30 training examples and evaluated on a disjoint, randomly selected set of 50 held-out examples from a 196-example test bed. The ACE-trained agent correctly identified and notified the required recipients in 41 of 50 cases (82%), compared with 24 of 50 cases (48%) for an untrained baseline using the same Generator, tools, mock data environment, and initial Context Playbook. The framework may extend to complex, repeatable industry workflows in which agents must coordinate multiple tools, apply business rules, and recover safely from incomplete or ambiguous information. By converting operational exceptions, human corrections, and approved recovery paths into inspectable guidance, ACE could support more consistent, auditable, and context-aware agent behavior within established governance and human-oversight controls.

1 INTRODUCTION

The ability to remember is central to meaningful interaction. Humans rely on memory to carry forward prior conversations, learn preferences over time, and build shared context that makes future exchanges more effective. AI agents face a similar requirement when they operate across repeated workflows: they must preserve relevant prior experience, adapt to user- or environment-specific patterns, and apply earlier lessons when similar situations arise. In practice, however, AI systems have often had to approximate these capabilities through ad hoc context engineering and custom state management. The resulting gap is not only a limitation of model intelligence; it is a memory design problem that determines whether an agent can improve across episodes without modifying its weights.

Large language models are increasingly expected to complete multi-step workflows that span tools, state transitions, failures, retries, and user correction. In such settings, model capability alone is insufficient; performance depends on how effectively the agent stores, retrieves, and updates memory across previous episodes. Recent surveys and taxonomies argue that agent memory should be distinguished not only by duration, but also by function, including working memory, factual or semantic memory, episodic memory derived from interaction histories, and procedural memory that preserves reusable strategies and skills (Jiang et al., 2026; Zhang et al., 2024; Wang et al., 2024b; Fang et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

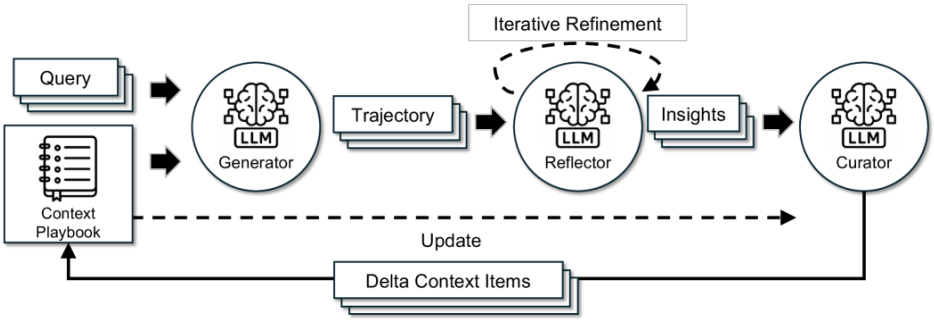


Figure 1: Core ACE workflow, adapted from prior work on Agentic Context Engineering (Zhang et al., 2025). The Generator proposes a reasoning trajectory, the Reflector compares it to ground truth, and the Curator distills corrective guidance into the Context Playbook.

Among these memory types, episodic memory is especially important for self-improving agents because it can preserve what happened in a prior run, what failed, how the user corrected the outcome, and which strategies proved robust under tool-call uncertainty (Pink et al., 2025; Park et al., 2023). Unlike semantic memory, which abstracts away instance-specific detail, episodic memory retains task-grounded traces that can later be reflected upon and generalized into actionable guidance (Shinn et al., 2023; Zhao et al., 2024). For AI applications, this implies that episodic memory cannot be static. It must be revised from human feedback, execution outcomes, and reflective analysis so that future episodes benefit from accumulated operational experience rather than repeating the same mistakes (Shinn et al., 2023; Zhao et al., 2024; Zhang et al., 2024).

2 Background and Motivation

2.1 Foundations in Agentic Context Engineering

As described in prior work (Zhang et al., 2025), Agentic Context Engineering (ACE) is an agentic training framework designed to optimize AI systems through self-reflective learning and iterative context refinement. At its core, ACE relies on incrementally improving the AI's internal "Context Playbook," which acts as a structured memory containing strategies, insights, and lessons drawn from human-provided ground-truth training samples. This learned Context Playbook is passed directly to the Generator LLM at execution time, providing a comprehensive, domain-experienced context that guides the agent in performing the specific agentic task.

ACE employs a modular loop: (1) a *Generator* LLM proposes a reasoning trajectory and actions; (2) a *Reflector* LLM compares generator outputs to ground-truth targets and surfaces improvements or barrier resolutions; and (3) a *Curator* LLM transforms these reflections into structured updates for the Context Playbook.

This grow-and-refine design preserves detailed, domain-specific knowledge and prevents context collapse, ensuring the playbook remains comprehensive yet scalable as the agent adapts to new data. The explicit separation of roles (Generator, Reflector, Curator) mirrors safe reinforcement learning practices, exploration, evaluation, and policy update, translated into context-space rather than weight-space.

3 PROPOSED FRAMEWORK: ACE FOR EPISODIC RULEBOOKS AND BARRIER RESOLUTION

This section describes our novel industry application of the Agentic Context Engineering (ACE) framework to a multi-branch, tool-mediated workflow with deliberately simulated barrier encounters. Rather than using ACE to refine a general procedural playbook for various task-answer questions, we use the ACE training loop to generate an *episodic rulebook* for a single, stable, repeatable agentic task with multiple tool-call steps to achieve its goal. In each training episode, the agent encounters controlled tool-level barriers,such as ambiguous territory mappings, unavailable representatives, missing assignments, or failed lookups,paired with ground-truth resolutions that define the expected recovery outcome. The Generator attempts the task under these conditions, the Reflector identifies where the trajectory diverges from the ground truth, and the Curator distills the successful resolution into reusable context for future task iterations.

The key novelty is therefore not simply that ACE improves through repeated examples, but that we intentionally construct a *barrier-rich tool environment* so ACE can learn reusable exception-handling rules from simulated operational failures and be able to generalize workflow resolutions to future unforeseen barriers of a similar nature. This converts episodic execution experience into a persistent, task-specific rulebook that strengthens future reasoning without modifying model weights.

Building on this design, we define an **episodic rulebook** as the persistent memory artifact produced by repeated ACE training episodes. Each entry captures a compact but actionable lesson from prior execution: the triggering context, the tool or reasoning barrier encountered, the recovery strategy that succeeded or failed, and the scope under which that lesson should generalize. This design is related to experiential-learning approaches that distill prior trajectories into reusable guidance (Zhao et al., 2024; Wang et al., 2024a) and to procedural-memory approaches that retain actionable strategies or skills for later tasks (Fang et al., 2025). The episodic rulebook differs from both raw execution logs and static procedural instructions. Raw traces preserve too much low-level detail to be reusable, while procedural context playbooks describe the human-authored expected path through a task but do not contain resolution strategies for encountered tool-call barriers. The episodic rulebook sits between these layers: it converts specific prior failures into curated, reusable guidance that helps the agent handle similar barriers in future runs.

4 METHODS

4.1 TRAINING USE CASE: FIRST PURCHASE OF A THERAPEUTIC; EMAIL NOTIFICATION WORKFLOW

The first-purchase notification task is a realistic, end-to-end agentic workflow in which an agent interprets a business event , a first-purchase alert for a novel therapeutic detected at a hospital , and executes the correct tool actions to notify the appropriate representative(s). The agent maps the purchasing account to the correct territory, identifies the responsible representative, determines whether medical follow-up is required, and sends the appropriate notification payload.

Each training sample preserves this same objective but varies the account, therapeutic context, territory mapping, representative assignment, and tool responses. Some samples follow the expected path cleanly; others introduce *simulated barrier scenarios* , tool-call failures such as ambiguous account-to-territory matches, unavailable representatives, or missing assignments , designed to introduce realistic operational friction. As shown in Figure 2, qualifying events trigger parallel Commercial and Medical notification tracks, each exposing two distinct barrier points. These variations allow ACE to learn not only the nominal tool sequence, but the reusable recovery rules needed when the environment behaves imperfectly.

4.2 DEFINING A TRAINING EPISODE

Each training episode consists of a system trigger, a series of tool-mediated actions, one or more branch points under uncertainty, and a terminal outcome. Each sample provides (1) an input trigger (“First Purchase of [therapeutic] detected at [hospital location]”), (2) a mock tool-call environment with embedded barrier scenarios, and (3) a human-defined ground-truth resolution that serves as the

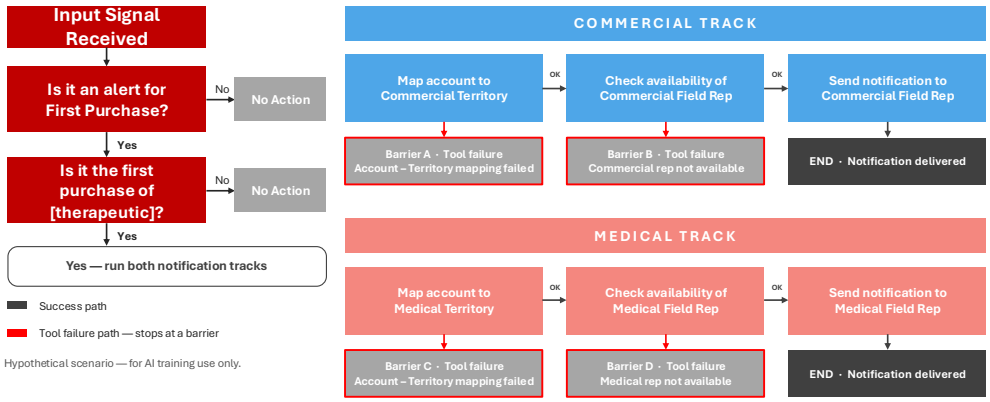


Figure 2: End-to-end decision flow for the first-purchase notification workflow, illustrating both the nominal execution path and simulated barrier scenarios. Upon receiving an input signal, the workflow first determines whether the event represents a first purchase of the target therapeutic; non-qualifying signals result in no action. Qualifying events trigger two parallel notification tracks, Commercial and Medical. Each track exposes two distinct barrier points: Barrier A/C, in which the account-to-territory mapping tool fails, and Barrier B/D, in which the assigned representative is reported as unavailable. These barriers are injected via mock data sources to simulate realistic operational friction during training. Successful completion of both tracks results in notification delivery to the correct representatives. This workflow serves as the controlled training environment for the ACE Generator–Reflector–Curator loop.

evaluation anchor for the Reflector. The objective is not merely to complete the present episode, but to transform it into reusable memory for subsequent ones.

4.3 ACE Training Workflow for the First Purchase Use Case

As illustrated in Figure 3, each training episode passes through three sequential stages, Generator, Reflector, and Curator, that together convert a single execution attempt into a durable rulebook update. The first-purchase workflow provides the ideal environment for this loop: it requires the agent to interpret an input signal, navigate branching Commercial and Medical notification tracks, invoke tools in the correct sequence, and recover gracefully from simulated barrier scenarios.

4.3.1 Step 1: Generator Stage—Producing a Reasoning Plan and Executing a ReAct Loop

As shown in Figure 3, the Generator LLM is the entry point of each training loop. It receives the first-purchase input signal along with the current rulebook context (empty on the first sample) and reasons through the task via a ReAct-style agentic loop (Yao et al., 2023), interweaving reasoning traces and tool calls step by step. The Generator is implemented using the LlamaIndex FunctionAgent, which manages tool invocations, validates signatures, and maintains the reasoning loop until a complete action sequence is formed. The available tools are:

- `CheckRepAvailability`
- `MapAccountToTerritory`
- `IdentifyNeighboringTerritory`

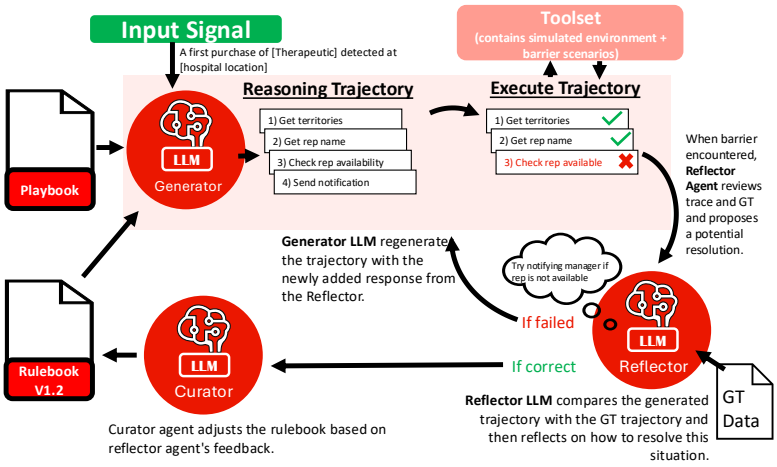


Figure 3: One complete ACE training-sample loop for the first-purchase workflow. The Generator LLM receives an input signal, produces a reasoning trajectory, and executes it against the toolset , retrieving territories, identifying the representative, checking availability, and sending the notification. When a barrier is encountered, the Reflector LLM compares the Generator's trajectory against ground-truth data and proposes a resolution (e.g., "Try notifying [alternate representative] if [primary] is not available"). If the trajectory is incorrect, the Curator LLM incorporates the Reflector's guidance into an updated rulebook (e.g., Rulebook v1.2); if correct, the loop concludes without an update. The Generator then reruns with the updated rulebook context, converting single-episode failures into durable, reusable exception-handling guidance.

- `GetRepForTerritory`
- `SendNotification`
- `CheckAccountHasMedicalHCP`

Each tool is backed by mock data sources constructed to surface realistic barrier conditions , ambiguous territory mappings, missing or conflicting account assignments, unavailable representatives, and unresolved lookups , mirroring the Barrier A through Barrier D scenarios shown in Figure 2. When a tool call returns an unexpected result, the Generator frequently stalls or proposes an incorrect next step, producing the divergent trajectory that the Reflector evaluates in Step 2.

4.3.2 Step 2: Reflector Stage — Comparing Generator Output to Ground Truth and Detecting Barriers

The Reflector LLM evaluates the Generator's trajectory against the human-annotated ground-truth target (Shinn et al., 2023; Zhao et al., 2024), identifying incorrect tool selections, missing steps, malformed arguments, logical errors, and recipient mismatches. When a barrier is detected, such as `check_availability` returning "representative unavailable," `map_account_to_territory` producing ambiguous results, or `retrieve_rep_assignment` returning "no match found" , the Reflector surfaces an explicit resolution recommendation. As depicted in Figure 3, representative resolutions include:

- "If availability lookup fails, attempt fallback logic to determine coverage."
- "If territory mapping is ambiguous, call `IdentifyNeighboringTerritory` before proceeding."

- "When notification cannot be sent due to a missing representative, escalate to the medical field representative only."

These reflections form the learning signal passed to the Curator in Step 3.

4.3.3 STEP 3: CURATOR STAGE—CONVERTING REFLECTIONS INTO DELTA CONTEXT ITEMS

The Curator LLM transforms the Reflector's recommendations into *Delta Context Items*: concise, generalizable lessons integrated into the Context Rulebook for future episodes. As shown in Figure 3, this produces an incremented rulebook version (e.g., Rulebook v1.2) that the Generator consults on its next run. The Curator filters noise and redundancy, ensuring each entry is actionable and clearly scoped. Representative examples include:

- "Always perform territory mapping before checking availability."
- "On availability lookup failure, retry using the fallback assignment resolution tool."
- "If representative assignment is ambiguous, call `ResolveRepresentativeAssignment` before determining the recipient."

This conversion of execution failures into curated guidance parallels prior systems that distill trajectories into reusable lessons or procedural skill libraries (Zhao et al., 2024; Wang et al., 2024a; Fang et al., 2025). After several iterations, the rulebook accumulates domain-specific strategies, barrier-resolution patterns, correct tool sequencing, and recipient-identification heuristics , enabling the Generator to handle novel barrier combinations without modifying model weights.

5 EVALUATION RESULTS

We evaluated whether the episodic rulebook learned through the ACE Generator–Reflector–Curator loop improved task performance relative to an untrained baseline. Both systems used the same Generator architecture, toolset, mock data environment, and initial human-authored context playbook. The ACE condition additionally used the rulebook produced from 30 training examples sampled from the 196-example test bed, whereas the baseline condition received no ACE training or learned rulebook updates. Evaluation was performed on a separate held-out set of 50 examples that was not used during training. These examples were randomly selected from the same test bed and comprised 9 happy-path scenarios, 19 scenarios in which no notification should be sent, and 22 barrier scenarios containing failed, missing, ambiguous, or otherwise non-nominal tool outputs.

An evaluation example was scored as correct when the system contacted and notified exactly the recipients required by the input signal and the state of the mock data environment. This outcome-based criterion therefore evaluated whether the agent correctly interpreted both the triggering event and the operational conditions exposed through its tool calls. For no-notification scenarios, success required the agent to correctly withhold all notifications.

As shown in Figure 4, across the 50 held-out examples, the ACE-trained Generator correctly completed 41 tasks, corresponding to an overall accuracy of 82%. Performance varied by scenario type: ACE succeeded on 7 of 9 happy-path examples (77.8%), all 19 no-notification examples (100%), and 15 of 22 barrier examples (68.2%). The untrained baseline completed 24 of 50 examples correctly, yielding an overall accuracy of 48%. Specifically, the baseline succeeded on 4 of 9 happy-path examples (44.4%), 15 of 19 no-notification examples (78.9%), and 5 of 22 barrier examples (22.7%). Figure 4 summarizes these results.

Relative to the baseline, ACE increased overall accuracy by 34 percentage points. The largest improvement occurred on barrier scenarios, where accuracy increased by 45.5 percentage points, from 22.7% to 68.2%. ACE also improved happy-path accuracy by 33.3 percentage points and no-notification accuracy by 21.1 percentage points. These results indicate that the learned rulebook contributed information beyond the initial human-authored context playbook, with the strongest observed benefit occurring when the workflow encountered non-nominal tool responses requiring exception-handling behavior.

	Happy Path Scenario	Irrelevant Trigger Input Scenario	Barrier Scenario	Total Accuracy
Validation Results	7/9 correct	19/19 correct	15/22 correct	**82%**
Baseline Results	4/9 correct	15/19 correct	5/22 correct	**48%**

 # Training Epochs: 30

Happy Path: The expected workflow with no barriers or unexpected tool call outputs in the mock training sample

Irrelevant Trigger Input: Trigger input is not related to a First Purchase of Therapeutic and thus should correctly be ignored.

Barrier Scenario: Training sample that contains mock data that will return an unexpected tool call result, such as a missing assignment information, failed returns, etc.

Figure 4: Held-out validation performance for the first-purchase notification workflow. Both conditions used the same Generator, tools, mock data environment, and initial human-authored context playbook; only the ACE condition received a generated context rulebook learned from 30 training examples. Accuracy was measured on 50 disjoint held-out examples as selection and notification of the correct recipients, including correctly sending no notification when required. ACE achieved 82% overall accuracy (41/50), compared with 48% (24/50) for the untrained baseline. Results are also reported by scenario type: happy path, no notification required, and tool-mediated barriers.

6 Discussion

The results provide preliminary evidence that ACE can derive reusable operational guidance from a comparatively small training set and apply that guidance to previously unseen workflow instances. After training on 30 examples, the resulting rulebook generalized to a disjoint 50-example evaluation set spanning nominal execution, non-actionable triggers, and tool-mediated barriers. The 34-percentage-point improvement over the untrained baseline, and particularly the 45.5-point gain on barrier scenarios, suggests that the learned rulebook contributed exception-handling guidance beyond the initial human-authored context playbook.

This finding supports the broader view that memory should be treated as a first-class design surface for tool-using agents (Pink et al., 2025; Jiang et al., 2026; Packer et al., 2023; Zhang et al., 2024; Wang et al., 2024b; Fang et al., 2025). Procedural playbooks define expected behavior, whereas episodic rulebooks preserve curated lessons from observed failures and approved recoveries. Related systems have shown that agents can benefit from verbal reflection, experiential knowledge extraction, skill libraries, and explicit long-term memory management (Shinn et al., 2023; Zhao et al., 2024; Park et al., 2023; Wang et al., 2024a; Packer et al., 2023). The contribution examined here is narrower and operationally focused: an inspectable rulebook learned for one stable workflow from simulated tool barriers and human-approved recovery outcomes. Because these artifacts are explicit rather than embedded in model weights, they can be inspected, revised, and governed independently of the underlying model.

The current study remains an initial proof of concept. It uses one workflow, one randomized held-out sample, and one training run; outcome accuracy also does not establish that the agent followed the correct reasoning trace or tool sequence. Future work should evaluate multiple random splits and runs, report uncertainty, ablate individual rulebook components, and measure trajectory quality, retrieval precision, rule staleness, and transfer across workflows. High-quality ground-truth out-

comes and representative barrier scenarios will remain essential because ambiguous supervision can produce overly specific or misleading rules.

7 CONCLUSION

This paper examined how Agentic Context Engineering (ACE) can be applied to a stable, repeatable agentic workflow with simulated barrier scenarios in tool-level calling. Using a multi-step first-purchase notification workflow, we trained the ACE Generator–Reflector–Curator loop to transform execution experience into an episodic rulebook that complements the workflow's initial procedural context playbook. The central novelty of the study was the deliberate construction of varied barrier scenarios within the tool environment,including ambiguous mappings, unavailable representatives, missing assignments, and failed lookups,and the pairing of each scenario with a human-defined ground-truth resolution. This design enabled ACE to compare incomplete or incorrect trajectories with approved outcomes and distill reusable exception-handling and recovery guidance that could generalize to future executions of the workflow.

AI-USE DISCLOSURE

This paper was prepared with the assistance of OpenAI GPT-5 to support literature searching, revision of the manuscript, improvement of readability, and reference formatting. Generative AI tools were not used to develop the theoretical or conceptual framework; formulate mathematical claims or proofs; propose or refine hypotheses; design the research methodology or experiments; implement the methods; generate, clean, or reformat datasets; conduct qualitative or thematic analysis; interpret results; or translate content. The authors independently defined the workflow, created the ground-truth resolutions, designed the barrier scenarios, implemented and ran the experiments, evaluated and scored the outputs, analyzed the results, and made all final scientific judgments. Every AI-assisted revision was reviewed by the authors. No AI-generated code, figures, data, experimental outputs, or citations were retained in the final manuscript, and no confidential, proprietary, personal, patient, or production data were entered into the generative AI tool. Both authors reviewed and approved this disclosure and take responsibility for the final content of this work.

REFERENCES

Ruiqi Fang, Yongqi Liang, Xin Wang, Jiaqi Wu, Shuofei Qiao, Peng Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025. doi: 10.48550/arXiv.2508.06433. URL https://arxiv.org/abs/2508.06433.

Dongping Jiang, Yongqi Li, Siyuan Wei, Jiayi Yang, Aditya Kishore, Andrew Zhao, Daniel Kang, Xiao Hu, Fangyi Chen, Qianwen Li, and Bo Li. Anatomy of agentic memory: Taxonomy and empirical analysis of evaluation and system limitations. *arXiv preprint arXiv:2602.19320*, 2026. doi: 10.48550/arXiv.2602.19320. URL https://arxiv.org/abs/2602.19320.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Sahil G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. doi: 10.48550/arXiv.2310.08560. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Marie Pink, Qinlan Wu, Vy A. Vo, Javier Turek, Jesse Mu, Alexander Huth, and Mariya Toneva. Position: Episodic memory is the missing piece for long-term LLM agents. *arXiv preprint arXiv:2502.06975*, 2025. doi: 10.48550/arXiv.2502.06975. URL https://arxiv.org/abs/2502.06975.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint*

arXiv:2303.11366, 2023. doi: 10.48550/arXiv.2303.11366. URL https://arxiv.org/abs/2303.11366.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. URL https://mlanthology.org/tmlr/2024/wang2024tmlr-voyager/.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 2024b. doi: 10.1007/s11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2210.03629. URL https://arxiv.org/abs/2210.03629.

Qinyuan Zhang, Chengbo Hu, Samyak Upasani, Banghua Ma, Fangyi Hong, Venkata Kamanuru, James Rainton, Chengshu Wu, Mengdi Ji, Hongyi Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025. doi: 10.48550/arXiv.2510.04618. URL https://arxiv.org/abs/2510.04618.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*, 2024. doi: 10.48550/arXiv.2404.13501. URL https://arxiv.org/abs/2404.13501.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. doi: 10.48550/arXiv.2308.10144. URL https://arxiv.org/abs/2308.10144.