

DolphinBench: Mapping the Pareto Frontier of Agent Memory

Soumil Rath, Deshraj Yadav, Taranjeet Singh
Mem0*

Abstract

Agents today often take real world actions which depend on long term memory and context recall over time. However, most current memory benchmarks are built for a conversational question-answer format, where the question itself signals that some fact must be retrieved, and often which one. Moreover, benchmarks rarely require anything beyond accuracy from submissions, allowing memory systems to make unreasonable cost/time tradeoffs to achieve higher scores. We present DolphinBench, a benchmark that evaluates memory directly through an agent’s task completion. DolphinBench includes three knowledge-work personas with roughly 500k tokens of user messages per persona and evaluates agents on tasks which depend on information from that history. We verify all 200 tasks per persona by running an agent with and without the relevant history, requiring success with it and failure without it. Finally, we require all evaluations to report total cost and latency alongside accuracy, which enables us to evaluate agent memory systems holistically. No existing memory benchmark combines all three. The dataset and evaluation code are available at <https://dolphinbench.ai>.

1 Introduction

The world is rapidly adopting agents in real world tasks, from software engineering [8] and web tasks [31] to customer service [28] and long running projects [26]. Such agents are increasingly required to keep track of information across project histories and past sessions that outgrow their context windows, which memory systems [4, 16, 20] are built to manage. It is important to evaluate this capability with memory benchmarks.

Memory benchmarks have largely not adapted to this setting. The dominant evaluation format is question answering [14, 27, 30], where a system ingests a long history and answers questions about it. Such evaluations are inherently biased though, as a direct question acts as a signal to the system that some fact must be retrieved, and frequently mentions which fact. A QA benchmark asks “What messaging platform does the team use?”, which already announces that a platform fact exists and is wanted; an action-based test of the same fact instead instructs the agent to post an update in the team’s channel, and never mentions platforms at all. So,

the evaluation effectively begins only after the most difficult step, recognizing that retrieval is required at all, has been completed on the system’s behalf. An agent acting under an instruction may get a weaker hint, but nothing in the format guarantees one. Evaluating memory for agents requires removing the question and using memory-dependent tasks as the accuracy signal.

Memory benchmarks commonly run into two further limitations, the first of which is in the range of metrics demanded from a valid solution. Since memory and information retrieval is an optimization problem where tradeoffs can be chosen, a full evaluation requires comparing accuracy, cost, and latency. A system can achieve high accuracy, for instance, by re-reading the entire history at each query with frontier models, incurring significant latency and compute costs that a production deployment could not sustain. We therefore require every result to report total cost and median task latency alongside accuracy.

Second, because most benchmarks are generated synthetically and at scale, they are prone to defects: incorrect answer keys, mislabeled supporting evidence, and questions that are unanswerable even given the complete history [12, 17]. Most benchmarks do not establish that each of their tests is actually solvable by a capable agent with perfect memory, which imposes an unknown ceiling on every reported score in case of such defects. We therefore verify every DolphinBench test using an agent with oracle memory: we provide the relevant history and require the agent to complete the task, then withhold that history and require it to fail. This checks that the task can be completed and that withholding the history affects success.

We introduce DolphinBench, a benchmark that evaluates agent memory through real world simulated actions. DolphinBench defines three knowledge-work personas: a startup CEO, an infrastructure engineer, and a product manager. We then simulate years of conversation (user input and tasks) for each persona with their agent. An agent processes this history and is then issued tasks whose correct actions depend on specific pieces of information within the session history.

Together, these choices give DolphinBench three properties that existing memory benchmarks do not combine: tests that evaluate memory through action rather than recall, results that report cost and latency alongside accuracy, and verification of every test through agent runs with and with-

*Correspondence: dolphinbench@mem0.ai

out the relevant history. Across the evaluated models and harnesses, the highest-scoring agent completes 70.67% of tasks. The memory systems rank differently across agents, and the most expensive agent is neither the most accurate nor the fastest.

2 Related Work

We identify three properties that are important for a memory benchmark for agents which mitigate the key issues faced by such benchmarks in practice.

- The test format must require action rather than pure text responses in a question-answer format.
- The final evaluation results must include cost and latency alongside accuracy.
- Each test should be solvable by the agent once it has perfect memory, and unsolvable without it (i.e. no ground truth errors, no hidden assumptions, and no tests that pass regardless of memory).

Table 1 shows how prior memory benchmarks compare to DolphinBench on each property. The following subsections develop on the justification and existing work for each principle in turn.

2.1 Task Format

Memory evaluation began as conversational QA [14, 30], later expanding into categories like temporal reasoning, knowledge updates, and abstention [7, 27] while keeping the QA format. The format carries the retrieval signal discussed in the introduction, and an action-style task removes it: the agent has to “know to remember”, which makes such tasks both harder and more representative of real world usage.

Some recent benchmarks have moved toward action-based testing. Mem2ActBench [21] scores agents on task completion in simulated environments; MemoryArena [6] evaluates interdependent agentic subtasks; STATE-Bench [11] tests stateful agent actions; MEMTRACK [5] tracks state across interleaved Slack, Linear, and Git timelines.

2.2 Metrics

Most benchmarks do not require cost and latency, making the partial reporting on these benchmarks inconsistent. For example, STATE-Bench reports dollar cost per task but not latency; MemoryArena reports latency but no aggregate cost; MemoryAgentBench reports memory construction and query execution times but not monetary cost. However, neither cost nor latency alone is enough, since inference choices like batch size and hardware trade them off. Thus, both these metrics should appear in any valid submission.

Accuracy alone is also easy to skew: the same benchmark can produce very different numbers under a modified judge prompt or judge model, with Penfield Labs [17] showing that one evaluated judge configuration accepted approximately 63% of deliberately incorrect, topically related answers. Cost and latency are harder to quietly improve, which makes them an anchor for comparisons across papers.

2.3 Validity

An independent audit of LoCoMo [17] reported incorrect answers for 99 of 1,540 non-adversarial questions (6.4%). Other popular benchmarks contain similar issues. LongMemEval contains ground-truth arithmetic errors and mislabeled evidence, surfaced as GitHub issues by the community [12].

Careful pipeline construction reduces error rates but does not eliminate them, so BEAM [24] adds human review, checking that every question is answerable. However, a test can pass that review and still be solvable without any memory, and then every system gets it right whether its memory works or not. Every DolphinBench test therefore has to fail without the relevant sessions as well as pass with them. We verify both automatically before a test ships, as described in the Test Construction section.

3 DolphinBench

DolphinBench is a benchmark built to evaluate agents on their memory which directly tests task success. The benchmark covers three personas: Morgan, a startup CEO; Alex, an infrastructure engineer; and Riley, a product manager. These were chosen to have a wide coverage of different tools/apps the agent would use, queries across different task types, and thus different memory access patterns as well.

Each persona has a simulated history spanning several years, with 3,400–5,128 user messages totalling approximately 500k tokens¹. The agents go through this simulated history, after which they are tested on tasks that require information from within those simulated sessions. The benchmark contains 600 tests, 200 per persona. Tool calls and apps (Notion, Gmail, GitHub, etc.) are both simulated to reproducibly test real world actions [28, 31].

Worked example. Morgan test 001:

History. In a January session, Morgan mentions a coffee preference: “*still doing blue bottle if i’m out early enough.*”

Instruction. “*I’m out early. Please order one small latte for pickup.*”

Answer & grading. The agent must use `place_order` to order from Blue Bottle. A deterministic check requires the

¹Measured with tiktoken’s `o200k_base`, over user-message content only.

Benchmark	Action-based	Cost+latency	Solvability-certified
<i>Non-action paradigms</i>			
MemoryBank [30]	✗	✗	✗
LoCoMo [14]	✗	✗	✗
LongMemEval [27]	✗	✗	✗
MemoryAgentBench [7]	✗	~ ^a	✗
MemoryBench [1]	✗	~ ^b	✗
MemBench [23]	✗	~ ^a	✗
BEAM [24]	✗	✗	✗
RealMem [3]	✗	~ ^f	✗
RealTalk [10]	✗	✗	✗
MemProbe [13]	✗	~ ^c	~ ^e
<i>Action-based paradigm</i>			
MemoryCode [19]	✓	✗	✗
StoryBench [25]	~ ^d	~ ^f	✗
Mem2ActBench [21]	✓	✗	✗
STATE-Bench [11]	✓	~ ^g	✗
MemoryArena [6]	✓	~ ^a	✗
MEMTRACK [5]	✓	✗	✗
DolphinBench (ours)	✓	✓	✓

Table 1: Memory benchmarks compared on the three properties an action-based memory evaluation requires. ✓: yes, ✗: no, ~: partial: ^alatency but no cost; ^blatency only in an appendix; ^ctoken counts as the only cost signal, no latency; ^dactions in a story/text-game setting rather than app state; ^echecks task design and information disclosure after evaluation; ^ftiming and token counts, but no monetary cost; ^gmonetary cost but no latency.

order to succeed, and an LLM judge checks that the restaurant is Blue Bottle.

The request specifies the drink, size, and pickup, but not the café. The agent must use Morgan’s earlier preference to choose where to place the order.

3.1 Construction

DolphinBench is built in hierarchical stages (Figure 1), an approach also used by BEAM [24]. We first outline each persona’s history over several years, then develop it into quarterly plans, weekly events, and conversations. The resulting messages and app records provide the basis for test construction.

Personas

We define each persona’s name, role, relationships, and tools, including calendar, email, CRM, and deployment tools. An LLM drafts these descriptions, which later stages use when planning events and writing messages. We update the definitions as people or tools change throughout the history. DolphinBench’s three personas are Morgan, a startup CEO who manages investors, hiring, board communication, and strategy; Alex, an infrastructure engineer who handles deploys, incidents, on-call rotations, and runbooks; and Riley, a product manager who runs experiments, funnels, and retention work.

History Planning

We outline the persona’s work and personal life over several years. An LLM develops this narrative into quarterly plans, specifying dated events and the facts they are intended to establish or change. These include decisions, preferences, and developments in ongoing projects and relationships. We review the plans before generating the corresponding history.

An LLM then develops each quarter into weekly events and conversations. For each planned conversation, it specifies a date, why the persona contacts the agent, and what they ask for or share. The plan includes everyday exchanges alongside the quarter’s main events.

History Generation

Before writing these conversations, we update the simulated apps to reflect the planned events. An LLM specifies the required tool calls and arguments, such as updating an event in Google Calendar. We execute these calls in chronological order so that the messages can refer to the resulting app records.

The model writing the messages receives these records and tool outputs, together with the weekly plan, relevant facts, and earlier messages. It writes the user messages in the persona’s voice, using this context to keep them consistent with the apps and preceding conversations. Each released history ends at the first complete session that brings it to at

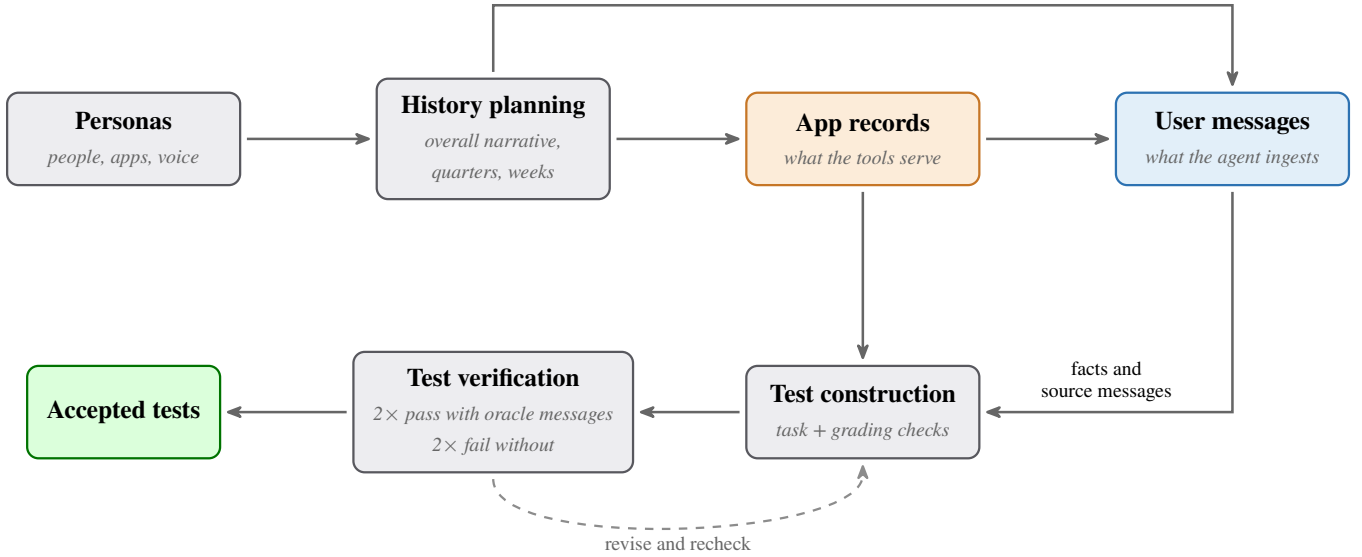


Figure 1: The DolphinBench construction pipeline.

least 500k user-message tokens.

We author only the user side of each conversation, because each agent would reply differently. The agent produces its own replies during ingestion. QA benchmarks instead provide a fixed transcript of both sides [14, 27], which fits a system that reads a conversation but not one that participates in it.

We record which facts each message mentions and which it introduces or changes. Before using a fact to construct a test, we check its description against the source messages and correct any claims they do not support.

Test Construction

An LLM proposes tasks that require the agent to use earlier information to complete new work. To design these tasks, it receives facts from the history, descriptions of the available tools, the evaluation date, and summaries of previously proposed tasks. For each task, it specifies the required facts, the intended actions, and what would be missing or wrong without the history. Each selected fact must affect a necessary part of the completed work.

An LLM then receives the proposed task, relevant source messages and later updates, available tools, and app records. It writes the user request and grading checks. The request supplies the information needed to act while leaving the remembered details for the agent to determine. We avoid cues that announce a stored preference, such as “the way I usually do it”. Checks on remembered information cite the supporting source messages and require only the details needed for the task.

We select the app records needed to carry out the task and add present-day records where the proposed situation

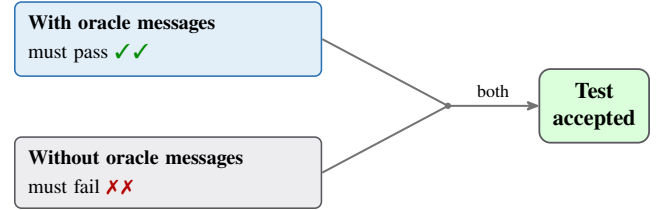


Figure 2: Test verification. The agent must pass both runs with oracle messages and fail both runs without them.

requires them. These records supply ordinary task inputs without revealing the remembered answer. We review the request, app records, and grading checks against the source messages before verification.

Test Verification

For each test, we provide the original history messages needed to complete it as oracle messages. We run the test twice with GPT-5.6-Luna given these oracle messages, and twice without them (Figure 2). The task, tools, and starting app records remain the same. Both runs with history must pass every grading check, and both runs without history must fail.

We inspect the runs to check that failure without history reflects missing information rather than tool errors or defects in the test. We revise candidates that fail these checks and exclude those that do not meet the acceptance conditions.

Harness	Agent model	Memory system	Accuracy (%)	Cost (\$)			Median latency (s)
				Agent	Memory system	Total	
Hermes	GPT-5.6-Luna	Built-in memory	65.67	61.48	0.00	61.48	44.35
		Mem0	70.67	61.54	34.68	96.21	37.69
		Honcho	68.50	96.56	46.30	142.85	44.66
		Hindsight	69.50	57.99	26.66	84.65	55.31
		Supermemory	59.17	63.86	281.79	345.64	52.68
Hermes	MiniMax M3	Built-in memory	26.50	107.03	0.00	107.03	51.40
		Mem0	47.83	111.71	36.31	148.02	46.95
		Honcho	47.33	168.39	43.46	211.85	80.33
		Hindsight	41.00	119.51	38.82	158.33	144.28
		Supermemory	20.00	111.68	386.02	497.70	137.52
Claude Code	Claude Sonnet 5	Built-in memory	26.33	1,132.75	0.00	1,132.75	32.19
		Mem0	32.33	1,803.63	26.94	1,830.57	42.30
		Honcho	35.83	1,556.02	9.79	1,565.82	37.92

Table 2: Results on DolphinBench: comparing the accuracy, cost, and latency of agents using different memory systems across 600 tasks spanning three personas. Bold marks the highest accuracy, lowest total cost, and lowest median latency within each harness–model group.

3.2 Grading & Metrics

We grade task completion by checking the agent’s tool calls against the actions and results required by the task. During test construction, we choose how to check each requirement:

- We use an LLM judge [29] (GPT-5.6-Sol) when checking a requirement requires interpreting meaning, such as whether an email conveys a particular decision. The judge receives the user’s request and the arguments of the tool call being checked, and is instructed to accept equivalent wording.
- We use deterministic checks when a requirement can be tested directly, such as whether a required tool was called or whether an identifier, number, date, or list matches the required value.

The checks must establish that each required action was completed correctly. For example, an email must have both the correct recipient and the required content in the same send call; satisfying those checks across different emails does not count. If a task requires two separate emails, one send call cannot count as both. A task passes only when every check passes.

Every submission must report accuracy, total cost across ingestion and testing, and median task latency. We calculate accuracy as the fraction of tasks passed and take the median of the individual test latencies across all personas.

4 Results

We evaluate agents with different combinations of harnesses, language models, and memory systems. Table 2 reports the

accuracy, cost, and latency of each combination.

Accuracy is the percentage of tasks passed. Total cost covers ingestion and testing, including memory processing. Latency is the median time per test, including tool use.

Memory performance reflects the capabilities of the model, the harness, and the memory system working together. With Hermes [15] and Mem0 [4], accuracy reaches 70.67% with GPT-5.6-Luna, compared with 47.83% with MiniMax M3. The memory systems also rank differently across configurations: Mem0 scores highest with both Hermes models, while Honcho [18] scores highest with Claude Code [2] and Claude Sonnet 5.

Higher accuracy need not require longer response times. With Hermes and GPT-5.6-Luna, Mem0 improves accuracy over built-in memory by five percentage points and reduces median task latency from 44.35 to 37.69 seconds, while total cost rises from \$61.48 to \$96.21.

Lower cost and higher accuracy can also come with longer response times. With the same harness and model, Hindsight [9] costs \$84.65 compared with Honcho’s \$142.85 and scores one percentage point higher, but its median task latency is 55.31 seconds compared with Honcho’s 44.66 seconds.

Evaluating each configuration on all three metrics lets DolphinBench map the Pareto frontier and helps users choose the configuration that best matches their accuracy, cost, and latency requirements.

We release the configuration, cost calculations, and full trace of every run in Table 2 (tool calls, arguments, and per-check verdicts) alongside the dataset.

5 Limitations

Synthetic data. We generate conversation histories rather than collect them from real users. These histories may not capture the variety of writing styles, languages, and interaction patterns found in real conversations. Checking facts against source messages and verifying the tests helps catch errors, but inconsistencies may remain.

Domain coverage. Three personas cover only a small part of knowledge work. They exercise different tools and memory requirements, but do not represent domains such as law, medicine, or long-term customer support. The construction pipeline supports adding personas to broaden this coverage.

Simulated environment. Agents interact with simulated apps rather than live services. The apps support selected operations and errors, but do not reproduce the full behavior of production systems, including service outages and changes made by other users during a task. This makes the tasks reproducible while leaving some difficulties of real deployments untested.

Task length. Each task requires one to four separately graded actions, though agents may make additional calls to find information or correct mistakes. These tasks test whether agents use remembered information correctly without requiring long workflows. Longer tasks would extend the evaluation to using memory across many dependent steps.

Separate ingestion and testing. Agents process the full history before taking any tests. This lets us compare agents against the same completed history, but does not test how they handle new conversations and tasks arriving throughout an ongoing evaluation. Interleaving ingestion and testing would better represent that continuous use.

6 Conclusion

DolphinBench evaluates agent memory through the actions agents take. We grade task completion, verify every test through agent runs with and without the relevant history, and require every result to report cost and latency alongside accuracy.

Our results show that a more accurate agent need not be slower, and a more expensive agent need not be more accurate. We therefore compare accuracy, cost, and latency together to assess each agent configuration as a whole.

We intend to keep the benchmark growing: histories that exceed model context windows, more personas, longer tasks, and interleaved ingestion and testing that better matches real deployments. As agents improve, the benchmarks should become harder with them.

References

- [1] Qingyao Ai, Yichen Tang, Changyue Wang, Jianming Long, Weihang Su, and Yiqun Liu. MemoryBench: A benchmark for memory and continual learning in LLM systems. *arXiv preprint arXiv:2510.17281*, 2025. URL <https://arxiv.org/abs/2510.17281>.
- [2] Anthropic. Claude Code. <https://github.com/anthropics/claude-code>. Accessed September 21, 2026.
- [3] Haonan Bian, Zhiyuan Yao, Sen Hu, Zishan Xu, Shaolei Zhang, Yifu Guo, Ziliang Yang, Xueran Han, Huacan Wang, and Ronghao Chen. RealMem: Benchmarking LLMs in real-world memory-driven interaction. *arXiv preprint arXiv:2601.06966*, 2026. URL <https://arxiv.org/abs/2601.06966>.
- [4] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025. URL <https://arxiv.org/abs/2504.19413>.
- [5] Darshan Deshpande, Varun Gangal, Hersh Mehta, Anand Kannappan, Rebecca Qian, and Peng Wang. MEMTRACK: Evaluating long-term memory and state tracking in multi-platform dynamic agent environments. *arXiv preprint arXiv:2510.01353*, 2025. URL <https://arxiv.org/abs/2510.01353>. Accepted to the NeurIPS 2025 SEA Workshop.
- [6] Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv preprint arXiv:2602.16313*, 2026. URL <https://arxiv.org/abs/2602.16313>.
- [7] Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *International Conference on Learning Representations (ICLR)*, 2026. URL <https://arxiv.org/abs/2507.05257>.
- [8] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://arxiv.org/abs/2310.06770>.
- [9] Chris Latimer, Nicol   Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects. *arXiv preprint arXiv:2512.12818*, 2025. URL <https://arxiv.org/abs/2512.12818>.

- [10] Dong-Ho Lee, Adyasha Maharana, Jay Pujara, Xiang Ren, and Francesco Barbieri. REALTALK: A 21-day real-world dataset for long-term conversation. *arXiv preprint arXiv:2502.13270*, 2025. URL <https://arxiv.org/abs/2502.13270>.
- [11] Lewis Liu and Nishant Yadav. Introducing STATE-Bench: A benchmark for AI agent memory. Microsoft Open Source Blog, May 2026. URL <https://opensource.microsoft.com/blog/2026/05/19/introducing-state-bench-a-benchmark-for-ai-agent-memory/>. STATE = Stateful Task Agent Evaluation; code at <https://github.com/microsoft/STATE-Bench>.
- [12] LongMemEval Contributors. LongMemEval ground-truth issues #19 and #22. GitHub issues, xiaowu0162/LongMemEval, 2025. Issue #19: <https://github.com/xiaowu0162/LongMemEval/issues/19>; Issue #22: <https://github.com/xiaowu0162/LongMemEval/issues/22>.
- [13] Enze Ma, Yufan Zhou, Wei-Chieh Huang, Jie Yang, Huanhuan Ma, Zixuan Wang, Chengze Li, Chunyu Miao, Philip S. Yu, and Zhen Wang. MEMPROBE: Probing long-term agent memory via hidden user-state recovery. *arXiv preprint arXiv:2606.24595*, 2026. URL <https://arxiv.org/abs/2606.24595>.
- [14] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL <https://arxiv.org/abs/2402.17753>.
- [15] Nous Research. Hermes Agent. Open-source AI agent framework, 2026. <https://github.com/NousResearch/hermes-agent>.
- [16] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL <https://arxiv.org/abs/2310.08560>.
- [17] Penfield Labs. We audited LoCoMo: 6.4% of the answer key is wrong and the judge accepts up to 63% of intentionally-wrong answers. DEV Community blog post, April 2026. URL <https://dev.to/penfieldlabs/we-audited-locomo-64-of-the-answer-key-is-wrong-and-the-judge-accepts-up-to-63-of-intentionally-33lg>. Audit repository at <https://github.com/dial481/locomo-audit>.
- [18] Plastic Labs. Honcho: Memory infrastructure for stateful agents. Open-source memory library, 2025. <https://github.com/plastic-labs/honcho>.
- [19] Nathanaël Carraz Rakotonirina, Mohammed Hamdy, Jon Ander Campos, Lucas Weber, Alberto Testoni, Marzieh Fadaee, Sandro Pezzelle, and Marco Del Tredici. From tools to teammates: Evaluating LLMs in multi-session coding interactions. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. URL <https://arxiv.org/abs/2502.13791>.
- [20] Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025. URL <https://arxiv.org/abs/2501.13956>.
- [21] Yiting Shen, Kun Li, Wei Zhou, and Songlin Hu. Mem2ActBench: A benchmark for evaluating long-term memory utilization in task-oriented autonomous agents. *arXiv preprint arXiv:2601.19935*, 2026. URL <https://arxiv.org/abs/2601.19935>.
- [22] Supermemory. Supermemory. <https://github.com/supermemoryai/supermemory>. Accessed September 21, 2026.
- [23] Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. URL <https://arxiv.org/abs/2506.21605>.
- [24] Mohammad Tavakoli, Alireza Salemi, Carrie Ye, Mohamed Abdalla, Hamed Zamani, and J. Ross Mitchell. Beyond a million tokens: Benchmarking and enhancing long-term memory in LLMs. *arXiv preprint arXiv:2510.27246*, 2025. URL <https://arxiv.org/abs/2510.27246>.
- [25] Luanbo Wan and Weizhi Ma. StoryBench: A dynamic benchmark for evaluating long-term memory with multi turns. *arXiv preprint arXiv:2506.13356*, 2025. URL <https://arxiv.org/abs/2506.13356>.
- [26] 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*, 18(6):186345, 2024. doi: 10.1007/s11704-024-40231-1. URL <https://doi.org/10.1007/s11704-024-40231-1>.
- [27] Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations (ICLR)*, 2025. URL <https://arxiv.org/abs/2410.10813>.
- [28] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ -bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024. URL <https://arxiv.org/abs/2406.12045>.
- [29] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhaghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and chatbot arena. In *Advances in Neural*

Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, 2023. URL <https://arxiv.org/abs/2306.05685>.

- [30] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL <https://arxiv.org/abs/2305.10250>.
- [31] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024. URL <https://arxiv.org/abs/2307.13854>.