Evaluating Memory in LLM Agents via Incremental Multi-Turn Interactions

Yuanzhe Hu [*1] **Yu Wang** [*1] **Julian McAuley** [1]

Abstract

Recent benchmarks for Large Language Model (LLM) agents have primarily focused on evaluating planning and execution capabilities, while another critical component—memory, encompassing how agents store, retrieve, and update long-term information—has much fewer benchmarks for evaluation. We term agents with memory mechanisms as **memory agents**. In this paper, we identify four core competencies essential for memory agents: accurate retrieval, test-time learning, long-range understanding, and conflict resolution. Existing datasets either rely on limited context lengths or are tailored for static, long-context settings like book-based QA, which do not reflect the interactive, multi-turn nature of memory agents that incrementally accumulate information. Furthermore, no existing benchmarks cover all four competencies. Therefore, we introduce **MemAE** (Memory Agent Evaluation), a unified evaluation framework specifically designed for memory agents. Our benchmark combines reformulated existing datasets with newly constructed ones, covering above four identified memory competencies, providing a systematic and challenging testbed for assessing memory quality. We evaluate a diverse set of memory agents, ranging from simple context-based and retrieval-augmented generation (RAG) systems to advanced agents with external memory modules and tool integration. Empirical results reveal that current methods fall short of mastering all four competencies, underscoring the need for further research into comprehensive memory mechanisms for LLM agents.

[*]Equal contribution [1]UC San Diego. Correspondence to: Yuanzhe Hu <yuh127@ucsd.edu>, Yu Wang <yuw164@ucsd.edu>, Julian McAuley <jmcauley@ucsd.edu>.

Proceedings of the 1st Workshop on Long-Context Foundation Models, Vienna, Austria. 2024. Copyright 2024 by the author(s).

1. Introduction

Large Language Model (LLM) agents have rapidly transitioned from proof-of-concept chatbots to end-to-end systems that can write software (Wang et al., 2024a), control browsers (Müller & Žunič, 2024), and reason over multi-modal inputs. Frameworks such as MANUS, OWL (Hu et al., 2025), OPENHANDS (Wang et al., 2024a), and CODEX routinely solve complex, tool-rich tasks and achieve state-of-the-art results on agentic benchmarks like GAIA (Mialon et al., 2023) and SWE-Bench (Jimenez et al., 2023). Yet these evaluations focus almost exclusively on *skill* (planning, tool use, code synthesis) and leave the equally important question of *memory*—how an agent stores, retrieves, and updates private long-term context—largely unexplored.

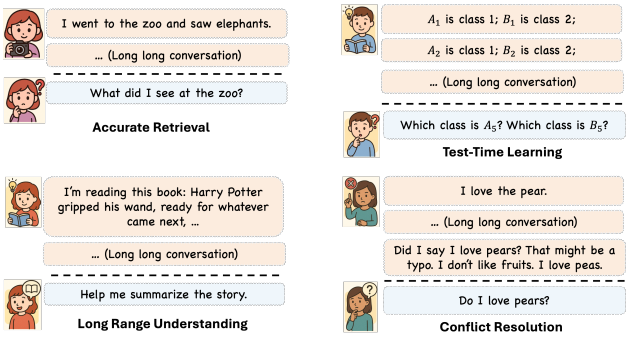


Figure 1: Four complementary competencies that memory agents should have.

Recent memory-centric architectures—ranging from parametric memory systems like MemoryLLM (Wang et al., 2024b) and M+(Wang et al., 2025) to commercial token-level memory solutions such as MEMGPT(Packer et al., 2023), MEM0(Chhikara et al., 2025), COGNEE, and ZEP(Rasmussen et al., 2025)—employ diverse strategies for storing and retrieving episodic information. Despite growing interest, their real-world effectiveness remains largely anecdotal, and there is currently no unified benchmark for systematically evaluating the quality of memory in agents. In this paper, we refer to agents equipped with memory mechanisms as **Memory Agents**, where memory can take various forms, including parameters, vectors, textual his-

tories, or external databases. In this paper, we primarily focus on memory agents that utilize textual histories and external databases, as these approaches are most commonly deployed in real-world applications. In contrast, memory encoded in model parameters (Wang et al., 2024b; 2025; Yin et al., 2024) remains largely within academic research and is typically less capable than proprietary memory systems equipped on closed-sourced API models..

To evaluate memory agents, we identify four complementary competencies (Examples shown in Figure 1): **Accurate Retrieval**, **Test-Time Learning**, **Long-Range Understanding** and **Conflict Resolution**. We also introduce a unified evaluation framework, **MemAE**, specifically designed to assess a broad spectrum of memory mechanisms in agent systems. In this framework, agents are presented with sequences of textual inputs that simulate multi-turn interactions with users. We repurpose existing datasets originally developed for long-context LLM evaluation by segmenting their inputs into multiple chunks and feeding them incrementally to the agent. However, since these datasets do not fully capture all four targeted memory competencies, we also introduce two new datasets: **EventQA** and **FactConsolidation**, each designed to evaluate long-range understanding, conflict resolution, and memory-based inference. Our benchmark includes evaluations of state-of-the-art commercial memory agents (such as Mem0 and MemGPT), long-context agents that treat the full input as memory, and retrieval-augmented generation (RAG) agents that extend their memory through retrieval methods. We examine how techniques developed for long-context models and RAG transfer to the memory agent setting, and how commercial memory agents perform under more challenging, competency-specific tests. By providing a consistent evaluation protocol across diverse agent architectures and datasets, **MemAE** delivers comprehensive insights into agent performance across the four core memory competencies.

Our contributions are summarized as follows:

- **Datasets:** We re-structure existing datasets and create two new datasets to construct a comprehensive benchmark, covering four distinct memory competencies.
- **Framework:** We provide a unified evaluation framework MemAE, and open-source the codebase to encourage reproducibility and further research.
- **Empirical Study:** We implement various simple agents with diverse memory mechanisms, adopt commercial agents, and evaluate these agents on our proposed benchmark. With our results, we show that existing memory agents, while effective in some tasks, still face significant challenges on some aspects.

2. Methodology

In this section, we introduce the four aspects of evaluation in detail, and the datasets we repurposed and curated for these aspects, as well as the MemAE framework.

2.1. Aspects of the Evaluation

The evaluation of memory agents encompasses the following key dimensions:

Accurate Retrieval (AR) The task of accurately retrieving information has been extensively explored in prior work. In the domain of long-context modeling, the Needle-in-a-Haystack (NIAH) task is widely used to evaluate a model's ability to locate a specific value based on a given key within a lengthy input. In the RAG setting, this corresponds to document-based question answering (QA), where the model must identify and extract relevant snippets from one or more documents to answer a query. These snippets might reside in a single location or be distributed across multiple documents. In this paper, we define AR as the ability of an agent to identify and retrieve important information that may be dispersed throughout a long dialogue history.

Test-Time Learning (TTL) An essential capability for real-world agents is the ability to acquire new skills dynamically through interaction with users. This mirrors the concept of In-Context Learning (ICL) in LLMs, where the model learns from a prompt containing a small number of examples, often framed as few-shot classification tasks. Ideally, performance improves with additional examples in the prompt. We define Test-Time Learning (TTL) as the agent's ability to learn to perform new tasks directly from the conversation. This property is crucial for enabling self-evolving agents that can continuously adapt and improve in real-world deployments.

Long-Range Understanding (LRU) Long-range understanding refers to the agent's ability to form abstract, high-level comprehension over extended conversations. For example, when a user narrates a long story, the agent should retain the content and derive a holistic understanding rather than just recall isolated facts. We define Long-Range Understanding (LRU) as the ability to reason about long-form inputs and answer high-level questions that require an understanding of the overall content, rather than detailed recall.

Conflict Resolution (CR) In long-term interactions, agents often face evolving or conflicting information—whether about the external world (e.g., changes in political leadership) or user-specific facts (e.g., a new occupation). We define Conflict Resolution (CR) as the agent's ability to detect and resolve contradictions between existing knowledge and newly acquired information, ensuring the agent remains aligned with current realities and user states. CR is distinct from Abstractive Retrieval (AR) in

two key ways. (1) Certain questions requiring CR cannot be answered solely through AR. As illustrated in Figure 1, an agent that retrieves all facts related to `pears` may fail to identify the updated information in the second message. (2) In AR, earlier messages remain relevant and should be retained, even when multiple pieces of evidence are required. In contrast, CR involves identifying outdated or incorrect information and discarding it. That is, AR requires preservation of all related content, whereas CR requires overwriting prior facts to reflect the most up-to-date truth.

2.2. Various Memory Agents

We evaluate three major types of memory agents that reflect common strategies for handling long-term information. These approaches differ in how they store, retrieve, and reason over past inputs.

Long Context Agents Modern language models often support extended context windows ranging from 128K to over 1M tokens. A straightforward strategy for memory is to maintain a context buffer of the most recent tokens. For example, in a model with a 128K-token limit, the agent concatenates all incoming chunks until the total exceeds the window size. Once the limit is reached, the earliest chunks are evicted in a FIFO (first-in, first-out) manner.

RAG Agents RAG-based agents address context limitations by storing past information in an external memory pool and retrieving relevant content as needed. We consider three RAG variants: (1) *Simple RAG Agents*: All input chunks are stored as raw text. During inference, a keyword or rule-based string matching mechanism retrieves relevant passages. (2) *Embedding-based RAG Agents*: Each input chunk is embedded and saved. At query time, the agent embeds the query and performs retrieval using cosine similarity between embeddings. (3) *Structure-Augmented RAG Agents*: After ingesting all input chunks, the agent constructs a structured representation (e.g., knowledge graph or event timeline). Subsequent queries are answered based on this structured memory.

Agentic Memory Agents Agentic memory agents extend beyond static memory stores by employing agentic loops—iterative reasoning cycles in which the agent may reformulate questions, perform memory lookups, and update its working memory. These agents are designed to simulate a more human-like process of recalling, verifying, and integrating knowledge.

2.3. MemAE

Dataset Formulation We standardize all datasets into the format: $c_1, c_2, \cdots, c_n$ (chunks), $q_1, q_2, \cdots, q_m$ (questions), and $a_1, a_2, \cdots, a_m$ (answers), where c_i denotes the i-th chunk wrapped to construct a user message with instructions of memorizing the content in a sequential input, and $c_1, c_2, \cdots, c_n$ represents a single conversation. Each chunk is accompanied by instructions prompting the agent to memorize its contents. Example prompts are provided in Appendix C.1. When curating datasets like EventQA and FactConsolidation, we deliberately design scenarios where multiple questions follow a single context. This allows us to probe the model's memory multiple times with one sequential injection. For example, in LME (S*), five contexts are paired with 300 questions (shown in Table 3 in Appendix B). This design choice reflects a key trend: as LLMs support increasingly long context windows and memory agents become more capable of handling extended inputs, evaluation datasets must also scale accordingly. Injecting 1M tokens for just one question is resource-inefficient, whereas associating the same input with many questions provides significantly higher utility.

Agents Formulation In our framework, all agents are required to take the chunks one by one, absorb them into memory, and incrementally update the memory. After seeing all the chunks, we ask the agent to answer the related questions.

3. Experiments

3.1. Experimental Setup

The datasets are split into four categories and the statistics of all datasets are also shown in Table 3. The evaluation metrics for all datasets are shown in Table 2 in Appendix B, along with more dataset details. Then for the agents, as described in Section 2.2, we consider three categories of agents: *Long-Context Agents*, *RAG agents* and *Agentic Memory Agents*.

3.2. Overall Performance Comparison

Table 1 presents the overall performance across different benchmarks. We summarize the key findings as follows: (1) **Superiority of RAG methods in Accurate Retrieval Tasks.** Most RAG Agents are better than the backbone model "GPT-4o-mini" in the tasks within the Accurate Retrieval Category. This matches our intuition where RAG agents typically excel at extracting a small snippet of text that is crucial for answering the question. (2) **Superiority of Long-Context Models in Test-Time Learning and Long-Range Understanding** Long-context models achieve the best performance on TTL and LRU. This highlights a fundamental limitation of retrieval-augmented generation (RAG) methods and commercial memory agents, which still follow an agentic RAG paradigm. These systems retrieve only partial information from the past context, lacking the ability to capture a holistic understanding of the input—let alone perform learning across it. (3) **Limitation of All Existing Methods on Conflict Resolution** Although being a

Table 1: Overall Performance Comparison. All RAG agents and commercial memory agents use GPT-4o-mini as the backbone. Thus we highlight the performance of GPT-4o-mini as the reference. FactCon-SH and FactCon-MH mean FactConsolidation Single Hop and FactConsolidation Multi Hop, respectively.

	AR					TTL		LRU	CR	
Agent Type	RULER	NIAH	∞Bench-QA	LME(S*)	EventQA	MCC	Recom	∞Bench-Sum	FactCon-SH	FactCon-MH
Long-Context Agents										
GPT-4o	61.5	25.0	**55.4**	32.0	77.2	87.6	12.3	32.2	**60.0**	**5.0**
GPT-4o-mini	53.5	22.8	44.9	30.7	59.0	82.4	15.1	28.9	45.0	5.0
GPT-4.1-mini	**74.5**	**94.8**	45.8	**55.7**	**82.6**	75.6	16.7	41.9	36.0	5.0
Gemini-2.0-Flash	73.0	83.8	53.2	47.0	67.2	84.0	8.7	23.9	30.0	3.0
Claude-3.7-Sonnet	65.0	38.0	50.6	34.0	74.6	**89.4**	**18.3**	**52.5**	43.0	2.0
GPT-4o-mini	53.5	22.8	44.9	30.7	59.0	82.0	15.1	28.9	45.0	5.0
Simple RAG Agents										
BM25	**61.0**	**95.5**	45.6	**48.3**	**74.6**	**75.4**	13.6	20.9	**44.0**	5.0
Embedding RAG Agents										
Contriever	41.0	8.8	38.1	19.0	66.8	70.6	15.2	21.2	25.0	5.0
Text-Embed-3-Small	36.0	12.3	44.4	39.0	63.0	70.0	15.3	**25.7**	21.0	4.0
Text-Embed-3-Large	36.5	13.5	50.1	39.3	70.0	72.4	**16.2**	21.6	22.0	3.0
NV-Embed-v2	48.0	31.8	**51.4**	43.0	72.8	69.4	13.5	20.7	42.0	**6.0**
Structure-Augmented RAG Agents										
RAPTOR	23.5	4.5	31.3	31.7	45.8	59.4	12.3	13.4	19.0	2.0
GraphRAG	43.0	8.0	35.8	36.7	34.4	39.8	9.8	0.4	10.0	3.0
HippoRAG-v2	43.5	23.3	45.7	37.3	67.6	61.4	10.2	14.6	29.0	3.0
Mem0	28.0	4.8	22.4	36.0	37.5	3.4	10.0	0.8	18.0	2.0
Cognee	33.5	4.0	19.7	29.3	26.8	35.4	10.1	2.3	28.0	3.0
Agentic Memory Agents										
Self-RAG	38.5	7.0	28.5	23.0	31.8	11.6	12.8	0.9	14.0	2.0
MemGPT	31.0	3.5	20.8	32.0	26.2	67.6	14.0	2.5	13.0	3.0

well-discussed task in model-editing community (Mitchell et al., 2022; Fang et al., 2024), resolving conflict poses a significant challenge on memory agents. We observe that all methods fail on the multi-hop situation. Only long context agents can achieve fairly reasonable results on single-hop scenarios. In Section E.3, we show that current reasoning models can have much better performance, while it does not change the conclusion that Conflict Resolution still poses a significant challenge on all memory mechanisms. (4) **Limited Performance of Commercial Memory Agents.** Commercial memory agents, such as MemGPT and Mem0, perform poorly across most benchmarks. This may be attributed to three main factors. First, these systems usually miss plenty of information when saving information into the memory. Mem0 relies on extracting the factual knowledges from the input, which naturally miss a large amount of information and may fail to reconstruct the original input and content from the memory, making the question answering based on the memory become much harder. For Accurate Retrieval Tasks, the agent could perform better in conversation tasks (Mem0 has shown their performance on LOCOMO), this is because the information is not as dense in conversations as in documents such as RULER, ∞-Bench, and it may fail drastically in the benchmarks which have very dense information. For TTL and LRU, the problem is even more severe. Second, both MemGPT and Mem0 relies on retrievers to retrieve part of the existing information in the memory. For Mem0, the retrieval is done only once like other RAG methods, thus the amount of the retrieved information is very limited. For MemGPT, even though the agent is an agentic framework and multiple retrievals are allowed, the information stored in the memory does not include temporal information, which makes it hard for the agent to look at the saved long documents with the original structure, leading to poor performances on LRU tasks. Lastly, the methods like MemGPT replies heavily on embedding-based retrieval mechanisms, which are potentially insufficient for fine-grained tasks like NIAH, where retrieving the precise "needle" in the haystack is critical.

4. Conclusion and Future Work

In this paper, we introduce **MemAE**, a unified benchmark designed to evaluate memory agents across four essential competencies.While prior benchmarks focus largely on skill execution or long-context reasoning, MemAE fills a critical gap by assessing how agents store, update, and utilize long-term information across multi-turn interactions. To build this benchmark, we restructure existing datasets and propose two new ones—**EventQA** and **FactConsolidation**—tailored to stress specific memory behaviors often overlooked in prior work. We evaluate a wide spectrum of agents, including long-context models, RAG-based systems, and commercial memory agents, under a consistent evaluation protocol. Our results reveal that, despite recent advances, current memory agents still exhibit substantial limitations when faced with tasks requiring dynamic memory updates and long-range consistency.

References

Anthropic. Claude 3.7 sonnet, 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet. This announcement introduces Claude 3.7 Sonnet, described as Anthropic's most intelligent model to date and the first hybrid reasoning model generally available on the market.

Asai, A., Wu, Z., Wang, Y., Sil, A., and Hajishirzi, H. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2023.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Bai, Y., Tu, S., Zhang, J., Peng, H., Wang, X., Lv, X., Cao, S., Xu, J., Hou, L., Dong, Y., et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024.

Bertsch, A., Ivgi, M., Xiao, E., Alon, U., Berant, J., Gormley, M. R., and Neubig, G. In-context learning with long-context models: An in-depth exploration. *arXiv preprint arXiv:2405.00200*, 2024.

Casanueva, I., Temčinas, T., Gerz, D., Henderson, M., and Vulić, I. Efficient intent detection with dual sentence encoders. In Wen, T.-H., Celikyilmaz, A., Yu, Z., Papangelis, A., Eric, M., Kumar, A., Casanueva, I., and Shah, R. (eds.), *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5. URL https://aclanthology.org/2020.nlp4convai-1.5/.

Chhikara, P., Khant, D., Aryan, S., Singh, T., and Yadav, D. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

DeepMind. Gemini pro, 2025. URL https://deepmind.google/technologies/gemini/pro/. This page provides an overview of Gemini Pro, highlighting its advanced capabilities and applications in various fields.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., and Larson, J. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Fang, J., Jiang, H., Wang, K., Ma, Y., Jie, S., Wang, X., He, X., and Chua, T.-S. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*, 2024.

Gutiérrez, B. J., Shu, Y., Qi, W., Zhou, S., and Su, Y. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*, 2025.

He, Z., Xie, Z., Jha, R., Steck, H., Liang, D., Feng, Y., Majumder, B. P., Kallus, N., and McAuley, J. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pp. 720–730, 2023a.

He, Z., Xie, Z., Jha, R., Steck, H., Liang, D., Feng, Y., Majumder, B. P., Kallus, N., and McAuley, J. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pp. 720–730, 2023b.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. RULER: What's the Real Context Size of Your Long-Context Language Models?, August 2024. URL http://arxiv.org/abs/2404.06654. arXiv:2404.06654 [cs].

Hu, M., Zhou, Y., Fan, W., Nie, Y., Xia, B., Sun, T., Ye, Z., Jin, Z., Li, Y., Zhang, Z., Wang, Y., Ye, Q., Luo, P., and Li, G. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation, 2025. URL https://github.com/camel-ai/owl.

Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Karpinska, M., Thai, K., Lo, K., Goyal, T., and Iyyer, M. One thousand and one pairs: A" novel" challenge for long-context language models. *arXiv preprint arXiv:2406.16264*, 2024.

Karpukhin, V., Oguz, B., Min, S., Lewis, P. S., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pp. 6769–6781, 2020.

Larson, S., Mahendran, A., Peper, J. J., Clarke, C., Lee, A., Hill, P., Kummerfeld, J. K., Leach, K., Laurenzano, M. A., Tang, L., and Mars, J. An evaluation dataset for intent classification and out-of-scope prediction. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1311–1316, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1131. URL https://aclanthology.org/D19-1131/.

Lee, C., Roy, R., Xu, M., Raiman, J., Shoeybi, M., Catanzaro, B., and Ping, W. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*, 2024.

Li, J., Wang, M., Zheng, Z., and Zhang, M. Loogle: Can long-context language models understand long contexts? *arXiv preprint arXiv:2311.04939*, 2023.

Li, R., Ebrahimi Kahou, S., Schulz, H., Michalski, V., Charlin, L., and Pal, C. Towards deep conversational recommendations. *Advances in neural information processing systems*, 31, 2018.

Li, X. and Roth, D. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. URL https://aclanthology.org/C02-1150/.

Li, X., Lipp, J., Shakir, A., Huang, R., and Li, J. Bmx: Entropy-weighted similarity and semantic-enhanced lexical search. *arXiv preprint arXiv:2408.06643*, 2024.

Liu, X., Eshghi, A., Swietojanski, P., and Rieser, V. Benchmarking natural language understanding services for building conversational agents, 2019. URL https://arxiv.org/abs/1903.05566.

Mialon, G., Fourrier, C., Wolf, T., LeCun, Y., and Scialom, T. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

Mitchell, E., Lin, C., Bosselut, A., Manning, C. D., and Finn, C. Memory-based model editing at scale. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15817–15831. PMLR, 2022.

Modarressi, A., Deilamsalehy, H., Dernoncourt, F., Bui, T., Rossi, R. A., Yoon, S., and Schütze, H. Nolima: Long-context evaluation beyond literal matching. *arXiv preprint arXiv:2502.05167*, 2025.

Müller, M. and Žunič, G. Browser use: Enable ai to control your browser, 2024. URL https://github.com/browser-use/browser-use.

OpenAI. Gpt-4o system card, 2025. URL https://openai.com/index/gpt-4o-system-card/. This report outlines the safety work carried out prior to releasing GPT-4o including external red teaming, frontier risk evaluations according to our Preparedness Framework, and an overview of the mitigations we built in to address key risk areas.

Packer, C., Fang, V., Patil, S., Lin, K., Wooders, S., and Gonzalez, J. Memgpt: Towards llms as operating systems. 2023.

Rasmussen, P., Paliychuk, P., Beauvais, T., Ryan, J., and Chalef, D. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., and Manning, C. D. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*, 2024.

Wang, X., Li, B., Song, Y., Xu, F. F., Tang, X., Zhuge, M., Pan, J., Song, Y., Li, B., Singh, J., et al. Openhands: An open platform for ai software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2024a.

Wang, Y., Gao, Y., Chen, X., Jiang, H., Li, S., Yang, J., Yin, Q., Li, Z., Li, X., Yin, B., et al. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*, 2024b.

Wang, Y., Krotov, D., Hu, Y., Gao, Y., Zhou, W., McAuley, J., Gutfreund, D., Feris, R., and He, Z. M+: Extending memoryllm with scalable long-term memory. *arXiv preprint arXiv:2502.00592*, 2025.

Wang, Y., Han, C., Wu, T., He, X., Zhou, W., Sadeq, N., Chen, X., He, Z., Wang, W., Haffari, G., Ji, H., and McAuley, J. J. Towards lifespan cognitive systems. *TMLR*, 2025/02.

Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., and Yu, D. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Wu, Q., Tao, C., Shen, T., Xu, C., Geng, X., and Jiang, D. Pcl: Peer-contrastive learning with diverse augmentations for unsupervised sentence embeddings. *arXiv preprint arXiv:2201.12093*, 2022.

Yen, H., Gao, T., Hou, M., Ding, K., Fleischer, D., Izsak, P., Wasserblat, M., and Chen, D. Helmet: How to evaluate long-context language models effectively and thoroughly. *arXiv preprint arXiv:2410.02694*, 2024.

Yin, Z., Sun, Q., Guo, Q., Zeng, Z., Cheng, Q., Qiu, X., and Huang, X.-J. Explicit memory learning with expectation maximization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16618–16635, 2024.

Yu, T., Zhang, S., and Feng, Y. Auto-rag: Autonomous retrieval-augmented generation for large language models. *arXiv preprint arXiv:2411.19443*, 2024.

Zhang, X., Chen, Y., Hu, S., Xu, Z., Chen, J., Hao, M., Han, X., Thai, Z., Wang, S., Liu, Z., et al. ∞bench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, 2024.

Zhong, Z., Wu, Z., Manning, C. D., Potts, C., and Chen, D. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*, 2023.

A. Related Work

A.1. Benchmarks with Long Input

In this section, we review prior work on long-context benchmarks. Early benchmarks designed for long-context evaluation include LongBench(Bai et al., 2023) and LooGLE(Li et al., 2023), with average input lengths of approximately 20k and 24k tokens, respectively. More recent benchmarks—such as ∞-Bench (Zhang et al., 2024), HELMET(Yen et al., 2024), RULER(Hsieh et al., 2024), NOCHA(Karpinska et al., 2024), NoLiMa (Modarressi et al., 2025) and LongBench V2(Bai et al., 2024)—extend context lengths to over 100k tokens and are primarily intended to evaluate the capabilities of long-context models. However, despite their scale, these benchmarks are not designed to assess memory agents, and no prior work has repurposed them for that goal. More recently, LongMemEval (Wu et al., 2024) has been proposed specifically for evaluating memory agents. While promising, LongMemEval uses synthetic conversations with limited topical diversity, making the dialogues less realistic and potentially less representative of real-world memory use cases.

A.2. Agents with Memory Mechanisms

Memory mechanisms are attracting more and more attention lately (Wang et al., 2025/02). Recent advancements in large language models (LLMs) have demonstrated the capability to process extended context lengths, ranging from 100K to over 1 million tokens. For instance, models such as GPT-4o (OpenAI, 2025) and Claude 3.7 (Anthropic, 2025) can handle inputs of approximately 100K to 200K tokens, while models like Gemini 2.0 Pro (DeepMind, 2025) and the GPT-4.1 series extend this capacity beyond 1 million tokens. These strong long-context capabilities enable a simple yet effective form of memory: storing information directly within the context window. However, this approach is inherently constrained by a hard limit—once the context window is exceeded, earlier information must be discarded.

In parallel, Retrieval-Augmented Generation (RAG) continues to serve as a dominant paradigm for managing non-parametric memory. By retrieving relevant information from external sources and feeding it to the LLM, RAG allows systems to overcome context length limitations. For example, OpenAI's recent memory functionality[1] combines explicit user preference tracking with retrieval-based methods that reference prior interactions. RAG methods can be broadly classified into three categories: **1. Simple RAG**: These methods rely on string-matching techniques such as TF-IDF, BM25, and BMX (Li et al., 2024), which are entirely non-neural and operate on string-level similarity. **2. Embedding-based RAG**: This class leverages neural encoders, primarily transformers, to map text into dense vector representations (Wu et al., 2022). Early methods like DPR (Karpukhin et al., 2020) and Contriever (Izacard et al., 2021) are based on BERT (Devlin et al., 2019), while more recent models such as NV-Embed-v2 (Lee et al., 2024) utilize decoder-only backbones and achieve significantly improved retrieval performance. **3. Structure-Augmented RAG**: These approaches enhance retrieval with structural representations such as graphs or trees. Representative systems include GraphRAG (Edge et al., 2024), RAPTOR (Sarthi et al., 2024), HippoRAG-V2 (Gutiérrez et al., 2025), Cognee, Zep (Rasmussen et al., 2025), and Mem0 (Chhikara et al., 2025), the latter of which also offers a graph-augmented variant, Mem^g, built on structured factual knowledge. Despite their effectiveness, RAG-based methods face challenges with ambiguous queries, multi-hop reasoning, and long-range comprehension. When questions require integrating knowledge across an entire session or learning from long, skill-encoding inputs, the retrieval mechanism—limited to the top-k most relevant passages—may fail to surface the necessary information. To address these limitations, Agentic Memory Agents introduce an iterative, decision-driven framework. Rather than relying on a single-pass retrieval, these agents dynamically process the query, retrieve evidence, reflect, and iterate through multiple retrieval and reasoning cycles. Examples include MemGPT (Packer et al., 2023), Self-RAG (Asai et al., 2023), and Auto-RAG (Yu et al., 2024). This agentic design is particularly effective for resolving ambiguous or multi-step queries. Nonetheless, these methods remain fundamentally constrained by the limitations of RAG—namely, the inability to fully understand or learn from long-range context that is inaccessible via retrieval alone.

B. Details of Dataset

Here we provide a detailed introduction to the datasets used for evaluating the four core competencies, including the corresponding metrics, average context length, and a brief description. Details are shown in Table 2.

[1] https://openai.com/index/memory-and-new-controls-for-chatgpt/

Table 2: Overview of evaluation datasets. We select datasets that cover various important long-context capabilities. SubEM: substring exact match. In the table, we underline the datasets we constructed ourselves.

Category	Dataset	Metrics	Description
Accurate Retrieval	RULER-QA1	SubEM	Gold passage retrieval QA.
	RULER-QA2		
	RULER-NIAH-MQ	Recall	Retrieve multiple "needles" from the "haystack".
	∞Bench-QA	ROUGE F1	Novel QA with entity replacement.
	LongMemEval (S)	Model Based Acc.	Dialogues based QA.
	LongMemEval (S*)		
	EventQA (*ours*)	Accuracy	Reasoning style NIAH. Novel multiple-choice QA on characters events.
Test-time Learning	BANKING77	Accuracy	Banking intent classification, 77 labels
	CLINC150		Intent classification, 151 labels
	NLU		Task intent classification, 68 labels
	TREC Coarse		Question type classification, 6 labels
	TREC Fine		Question type classification, 50 labels
	Movie Recommendation	Recall@5	Recommend movies based on provided dialogues examples.
Long Range Understanding	∞Bench-Sum	Model Based F1	Novel summarization with entity replacement.
Conflict Resolving	FactConsolidation-SH (*ours*)	SubEM	Conflict solving in single hop reasoning.
	FactConsolidation-MH (*ours*)		Conflict solving in multiple hop reasoning.

B.1. Datasets Preparation

In this section, we describe how we adapt existing datasets and construct new ones for evaluating each aspect introduced in Section 2.1. All datasets with their categories are shown in Table 3.

Datasets for Accurate Retrieval (AR) We adopt five datasets to evaluate the accurate retrieval capability of memory agents. Four are adapted from existing benchmarks, and one is newly constructed: (1) **RULER-QA**: This is a NIAH-style QA task where a long passage contains a small snippet answering the input question. The agent must identify and extract this relevant snippet from the extended context. (2) **NIAH**: We use the multiple-query (MQ) version of the Needle-in-a-Haystack dataset from RULER (Hsieh et al., 2024), where each query seeks a different numeric value embedded in a long passage. The agent must retrieve multiple distinct answers, requiring precise multi-needle retrieval. (3) **∞Bench-En.QA**: This task from ∞Bench presents free-form QA questions based on entire books, with all entities replaced by fictitious names to avoid contamination from model pretraining. Compared to synthetic datasets like RULER-QA, this benchmark is more realistic and challenging due to the natural narrative structure of books. (4) **LongMemEval**: This benchmark evaluates memory agents on long dialogue histories. Although task types like information extraction (IE) or multi-session reasoning are included, most tasks can be reformulated as single-retrieval problems requiring agents to retrieve the correct segments spanning a long multi-turn conversation. Among these, LongMemEval is already formatted for agent-based evaluation with session separation. We use the original LongMemEval(S) dataset ($\sim$110K tokens) and reformulated chat history into five long dialogues ($\sim$355K tokens) with 300 questions (LongMemEval (S*) in Table 3). We create LongMemEval (S*) specifically for increasing the number of questions per context, mitigating the exhaustive needs of reconstructing the memory for each quesiton. (5) **EventQA (ours)**: We introduce EventQA to evaluate agents' ability to recall and reason about temporal sequences in long-form narratives. Using five books from ∞Bench (each >390K tokens, counted using the `gpt-4o-mini` tokenizer), we identify the ten most frequently mentioned characters via `SpaCy` NER. We extract 101 events experienced by key characters using `gpt-4o`. For each event, we construct a 6-way multiple-choice question by pairing the true event with five distractors generated via `gpt-4o`. The agent receives five previous events and must identify the correct continuation. For these datasets, which are originally designed for long-context modeling, we split documents into chunks and sequentially inject them into the agent.

Datasets for Test-Time Learning (TTL) We evaluate TTL via two task categories: (1) **Multi-Class Classification (MCC)**: We adopt five classification datasets used in prior TTL work (Bertsch et al., 2024; Yen et al., 2024): BANKING77 (Casanueva et al., 2020), CLINC150 (Larson et al., 2019), TREC-Coarse, TREC-Fine (Li & Roth, 2002), and NLU (Liu et al., 2019). Each task requires the agent to map sentences to class labels, leveraging previously seen labeled examples in context. (2) **Recommendation (Recom)**: We use the Redial (Li et al., 2018) dataset to evaluate movie recommendation via dialogue.

Table 3: Datasets categorized by the specific aspects of evaluation. Avg. Len: Average Context Length (measured using the GPT-4o-mini model's tokenizer).

Capability	Benchmarks / Tasks	# of Sequences	# of QAs	Avg Len
Accurate Retrieval	RULER-QA (Hsieh et al., 2024)	200	200	305K
	RULER-NIAH-MQ (Hsieh et al., 2024)	100	100	448K
	∞Bench-QA (Zhang et al., 2024)	100	100	183K
	LongMemEval (S) (Wu et al., 2024)	500	500	110K
	LongMemEval (S*) (Wu et al., 2024)	5	300	355K
	EventQA (*ours*)	5	500	534K
Test-Time Learning	BANKING-77	100	100	
	CLINC-150	100	100	
	NLU	100	100	103K
	TREC (Coarse)	100	100	
	TREC (Fine)	100	100	
	Movie-Rec Redial (He et al., 2023b)	1	200	1.44M
Long-Range Understanding	∞Bench-Sum (Zhang et al., 2024)	100	100	172K
Conflict Resolution	FactConsolidation-SH (*ours*)	1	100	262K
	FactConsolidation-MH (*ours*)	1	100	

Following the setup from He et al. (2023a), the agent is exposed to thousands of movie-related dialogue turns and is asked to recommend twenty relevant movies based on the long interaction history.

Datasets for Long Range Understanding (LRU) For this task, we adopt the Summarization task `En.Sum` from ∞-Bench (Zhang et al., 2024). The agent is required to analyze and organize the plot and characters of the novel, and then compose a summary of 1000 to 1200 words.

Datasets for Conflict Resolution (CR) To assess whether an agent can consolidate conflicting factual updates and reason over them, we construct a new dataset called FactConsolidation. Specifically, We build this benchmark using counterfactual edit pairs from MQUAKE (Zhong et al., 2023). Each pair contains a true fact and a rewritten, contradictory version. These are ordered such that the rewritten (new) fact appears after the original, simulating a realistic update scenario. We concatenate multiple such edit pairs to create long contexts of length 32K, 64K, 262K. We then adapt MQUAKE's original questions and categorize them into: (1) **FactConsolidation-SH (Ours)** (SH means Single-Hop), requiring direct factual recall (e.g., "Which country was tool A created in?"), and (2) **FactConsolidation-MH (Ours)** (MH refers to Multi-Hop), requiring inference over multiple facts (e.g., "What is the location of death of the spouse of person B?"). Agents are prompted to prioritize later information in case of conflict and reason based on the final memory state. This setup directly evaluates the strength and consistency of conflict resolution over long sequences.

B.2. Evaluation Metrics

Accurate Retrieval (AR) We use five datasets to evaluate the accurate retrieval capability of memory agents. (1) **RULER-QA**: Since most answers in this dataset are short informational entities, such as years, names, or yes/no responses, we use substring exact match (SubEM) as the evaluation metric. SubEM measures whether the predicted answer exactly matches the gold answer as a substring, which is a common standard in question answering systems. (2) **NIAH**: The primary evaluation criterion is whether the agent can successfully retrieve the correct numbers. Therefore, we use average recall as the evaluation metric. (3) ∞**Bench-En.QA**: it is also a QA task and answers are mostly entity names. We use ROUGE F1 as the evaluation metric for this dataset. (4) **LongMemEval**: Since some of the questions have open-ended answers, we adopt the approach used in previous work and employ the GPT-4o model to assess whether the agent's responses meet the requirements. If a response is deemed satisfactory, it is marked as True. Finally, we calculate the proportion of satisfactory responses as the evaluation metric. (5) **EventQA**: In this dataset, we report the mean accuracy over 100 such questions per book, and ultimately present the average accuracy across all five books.

Test-time Learning (TTL) We evaluate TTL via two task categories: (1) **Multi-Class Classification (MCC)**: We adopt five classification datasets used in prior TTL work. In this dataset, we use average accuracy as the evaluation metric. (2)

Recommendation (Recom): In this task, the agent is required to recommend 20 movies based on the content of the dialogue. We evaluate the recommendations by calculating Recall@5, which measures the overlap between the top 5 recommended movies and the ground truth.

Long-Range Understanding (LRU) We evaluate LRU via the Summarization task `En.Sum` from ∞-Bench (Zhang et al., 2024). We follow the settings from (Yen et al., 2024) and use the GPT-4o model in evaluating the summarized text. In this process, we assess the fluency of the input text (scored as 0 or 1) and use the dot product of this score with the F1 score as the final evaluation metric.

Conflict Resolution (CR) We evaluate the CR via two datasets: **Single-Hop Editing** and **Multi-Hop Editing**. In these tasks, the agent's responses are mostly informational entities. Therefore, we also use SubEM (Substring Exact Match) as the evaluation metric.

C. Prompts

We introduce some example prompts used in this section.

C.1. Instructions for Memory Construction

When processing long-context inputs, we split the content into chunks of a specified size and feed these chunks into the agent as memory. The agent can then extract relevant information from its memory based on the query to assist with query execution. This chunking approach helps organize and manage large amounts of contextual information, making retrieval and reasoning more efficient. In Figure 2, we provide several example instructions that require the agent to memorize the corresponding context.

Prompts Used for Memory Construction on Various Tasks

IF *LongMemEval*:
Memorize the following conversation between the user and the assistant: \n <chunk> \n
ELIF *Movie Recommendation*:
Memorize the following dialogues between a user and recommender system: \n <chunk> \n
ELIF *Fact Consolidation*:
Memorize these following facts: \n <chunk> \n
ELSE:
Memorize the following content: \n <chunk> \n

Figure 2: The prompts we use for the agents to create the memory.

C.2. Instructions for Long-Context Agents

In Figure 3, we provide the examples of instructions used on different of datasets. For some existing datasets, we adopted the prompt settings from previous work such as (Hsieh et al., 2024; Wu et al., 2024). For example, for the dataset ∞**Bench-QA** and ∞**Bench-Sum**, we also inserted two answer examples as <demo> in the prompt to help the agent better understand the questions and standardize its outputs.

C.3. Instructions for RAG Agents

We provide examples of prompts used for the RAG based Agents in Figure 4. For this type of agent, after storing the input long context in memory, we use <question> as the memory retrieval query for most tasks. For tasks without this input element, such as **RULER-NIAH-MQ**, we use the question "What are all the special magic <type_needle_v> for <query> mentioned in the memorized text?" as the query. And for ∞**Bench-Sum**, we use the entire query without the <demo> for the memory retrieval.

Prompts Used for Long-Context Agents on Various Tasks

RULER-QA
The context is given as below: <memory>. \n please memorize it.\n Answer the question based on the memorized documents. Only give me the answer and do not output any other words. \n Now Answer the Question: <question> \n Answer:

RULER-NIAH-MQ
The context is given as below: <memory>. \n Please memorize it. \n Some special magic <type_needle_v> are hidden within the memorized text. Make sure to memorize it. I will quiz you about the <type_needle_v> afterwards.\n Now Answer the Question: What are all the special magic <type_needle_v> for <query> mentioned in the memorized text? \n The special magic <type_needle_v> for <query> mentioned in the memorize text are:

∞Bench-QA
The context is given as below: <memory>. \n Please memorize it. \n Based on the context you memorized, answer the question as concisely as you can, using a single phrase if possible. \n <demo>.\n Now Answer the Question: <question>.\n Answer:

LongMemEval
Here are several history chats between you and a user : <memory> \n Please memorize them. \n The history chats are between you and a user. Based on the relevant chat history, answer the question as concisely as you can, using a single phrase if possible.\n Current Date: <question_date>, \n Now Answer the Question: <question> \n Answer:

EventQA
The context is given as below: <memory>. \n Please memorize it. \n Based on the context you memorized, complete the task below: \n These are the events that have already occurred: \n <previous_events> \n Below is a list of possible subsequent events:\n <question> \n Your task is to choose from the above events which event happens next based on the book excerpt. In your response to me, only include the answer without anything else. \n The event that happens next is:

Label Matching (BANKING77, etc.)
The context is given as below: <memory>. \n Please memorize them. \n Use the provided mapping from the context to numerical label to assign a numerical label to the context. Only output "label: {{label}}" and nothing else. \n Question: <question> \n label:

Movie Recommendation
Here are dialogues between a user and recommender system: <memory>. \n Please memorize them. \n Pretend you are a movie recommender system. You need to recommend movies based on the dialogues you have memorized. Now I will give you a new conversation between a user and you (a recommender system). Based on the conversation, you reply me with 20 recommendations without extra sentences. \n For Example:\n [Conversation] \n The recommendations are: \n 1.movie1 \n 2.movie2 \n ...\n Here is the conversation: <question> \n The recommendations are:

∞Bench-Sum
The book is given as below: <memory> \n Please memorize it. \n You are given a book above and you are tasked to summarize it. Write a summary of about 1000 to 1200 words. Only write about the plot and characters of the story. Do not discuss the themes or background of the book. Do not provide any analysis or commentary. \n <demo> \n Now summarize the book.

Fact Consolidation
Here is a knowledge pool with lots of new facts: <memory>. \n Please memorize it. \n Pretend you are a knowledge management system. Each fact in the knowledge pool is provided with a serial number at the beginning, and the newer fact has larger serial number. \n You need to solve the conflicts of facts in the knowledge pool by finding the newest fact. You need to answer a question based on this rule. You should give a very concise answer without saying other words for the question **only** from the knowledge pool you have memorized rather than the real facts in real world. \n For example: \n [Knowledge Pool] \n Question: Based on the provided Knowledge Pool, what is the name of the current president of Country R? \n Answer: Person D. \n Now Answer the Question: Based on the provided Knowledge Pool, <question> \n Answer:

Figure 3: The prompts we use for the *Long-Context Agents* in Table 1. Here <memory> refers to the accumulated text from the sequential inputs.

D. Detailed Experimental Results

In this section, we provide a detailed version of the data presented in the main text.

Prompts Used for RAG Based Agents on Various Tasks

RULER-QA
Here is the context retrieved from memory: <memory>.\n Answer the question based on the retrieved context. Only give me the answer and do not output any other words. \n Now Answer the Question: <question> \n Answer:

RULER-NIAH-MQ
Here is the context retrieved from memory: <memory>.\n Some special magic <type_needle_v> are hidden within the retrieved text. Make sure to memorize it. I will quiz you about the <type_needle_v> afterwards.\n Now Answer the Question: What are all the special magic <type_needle_v> for <query> mentioned in the memorized text? \n The special magic <type_needle_v> for <query> mentioned in the memorize text are:

∞Bench-QA
Here is the context retrieved from memory: <memory>.\n Based on the context you retrieved, answer the question as concisely as you can, using a single phrase if possible. \n <demo>.\n Now Answer the Question: <question>.\n Answer:

LongMemEval
Here are retrieved several history chats between you and a user from memory: <memory> \n The retrieved history chats are between you and a user. Based on the relevant chat history, answer the question as concisely as you can, using a single phrase if possible.\n Current Date: <question_date>, \n Now Answer the Question: <question> \n Answer:

EventQA
Here is the context retrieved from memory: <memory>.\n Based on the context you retrieved, complete the task below: \n These are the events that have already occurred: \n <previous_events> \n Below is a list of possible subsequent events:\n <question> \n Your task is to choose from the above events which event happens next based on the book excerpt. In your response to me, only include the answer without anything else. \n The event that happens next is:

Label Matching (BANKING77, etc.)
Here are the examples retrieved from memory: <memory>. \n Use the retrieved mapping from the context to numerical label to assign a numerical label to the context. Only output "label: {{label}}" and nothing else. \n Question: <question> \n label:

Movie Recommendation
Here are retrieved dialogues between a user and recommender system from memory: <memory>. \n Pretend you are a movie recommender system. You need to recommend movies based on the example dialogues you have retrieved. Now I will give you a new conversation between a user and you (a recommender system). Based on the conversation, you reply me with 20 recommendations without extra sentences. \n For Example:\n [Conversation] \n The recommendations are: \n 1.movie1 \n 2.movie2 \n ...\n Here is the conversation: <question> \n The recommendations are:

∞Bench-Sum
The book context is retrieved from memory and it is given as below: <memory> \n You are given retrieved context above and you are tasked to summarize it. Write a summary of about 1000 to 1200 words. Only write about the plot and characters of the story. Do not discuss the themes or background of the book. Do not provide any analysis or commentary. \n <demo> \n Now summarize the book.

Fact Consolidation
Here is a list of knowledge retrieved from memory: <memory>. \n Pretend you are a knowledge management system. Each fact in the retrieved knowledge pool is provided with a serial number at the beginning, and the newer fact has larger serial number. \n You need to solve the conflicts of facts in the retrieved knowledge pool by finding the newest fact. You need to answer a question based on this rule. You should give a very concise answer without saying other words for the question **only** from the retrieved knowledge pool you have memorized rather than the real facts in real world. \n For example: \n [Knowledge Pool] \n Question: Based on the provided Knowledge Pool, what is the name of the current president of Country R? \n Answer: Person D. \n Now Answer the Question: Based on the provided Knowledge Pool, <question> \n Answer:

Figure 4: The prompts we use for the *Simple RAG Agents*, *Embedding RAG Agents*, *Structure-Augmented RAG Agents* and *Agentic Memory RAG Agents* in Table 1. Here <memory> refers to the retrieved text from the sequential inputs. For MemGPT method, we also add the phrase "Search Archival Memory" in prompt of each task.

D.1. Detailed Results on AR

In Table 4, we present the detailed results for each agent on every dataset. For AR tasks, using Simple RAG Agents equipped with retrievers like BM25 can significantly improve performance compared to the backbone model. This is because the GPT-4o-mini is limited by its 128K context length, which restricts the amount of information it can process at once.

Table 4: Overall Performance Comparison on the datasets for AR. All RAG agents and commercial memory agents use GPT-4o-mini as the backbone. Thus we highlight the performance of GPT-4o-mini as the reference.

Agent Type	RULER-QA1	RULER-QA2	RULER-NIAH	∞Bench-QA	LME(S)	LME(S*)	EventQA
Long-Context Agents							
GPT-4o	72.0	51.0	25.0	**55.4**	**61.4**	32.0	77.2
GPT-4o-mini	64.0	43.0	22.8	44.9	55.6	30.7	59.0
GPT-4.1-mini	83.0	**66.0**	**94.8**	45.8	**61.4**	**55.7**	**82.6**
Gemini-2.0-Flash	**87.0**	59.0	83.8	53.2	52.6	47.0	67.2
Claude-3.7-Sonnet	77.0	53.0	38.0	50.6	59.0	34.0	74.6
GPT-4o-mini	64.0	43.0	22.8	44.9	55.6	30.7	59.0
Simple RAG Agents							
BM25	**66.0**	**56.0**	**95.5**	45.6	**55.2**	**48.3**	**74.6**
Embedding RAG Agents							
Contriever	40.0	42.0	8.8	38.1	32.8	19.0	66.8
Text-Embed-3-Small	39.0	33.0	12.3	44.4	49.0	39.0	63.0
Text-Embed-3-Large	39.0	34.0	13.5	50.1	44.6	39.3	70.0
NV-Embed-v2	57.0	39.0	31.8	**51.4**	45.4	43.0	72.8
Structure-Augmented RAG Agents							
RAPTOR	28.0	19.0	4.5	31.3	38.8	31.7	45.8
GraphRAG	40.0	46.0	8.0	35.8	39.2	36.7	34.4
HippoRAG-v2	49.0	38.0	23.3	45.7	44.2	37.3	67.6
Mem0	24.0	32.0	4.8	22.4	45.0	36.0	37.5
Cognee	31.0	26.0	4.0	19.7	31.3	29.3	26.8
Agentic Memory Agents							
Self-RAG	39.0	38.0	7.0	28.5	23.4	23.0	31.8
MemGPT	27.0	35.0	3.5	20.8	41.4	32.0	26.2

Table 5: Overall performance comparison on the datasets for TTL, LRU and CR. All RAG agents and commercial memory agents use GPT-4o-mini as the backbone.

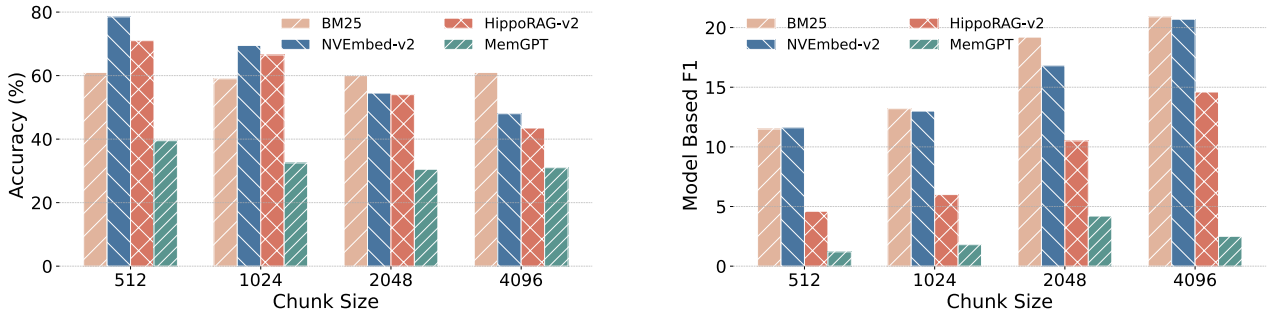
Agent Type	BANKING	CLINC	NLU	TREC C	TREC F	Recom	∞Bench-Summ	FactCon-SH	FactCon-MH
Long-Context Agents									
GPT-4o	96.0	96.0	**90.0**	87.0	69.0	12.3	32.2	**60.0**	**5.0**
GPT-4o-mini	93.0	93.0	87.0	73.0	66.0	15.1	28.9	45.0	**5.0**
GPT-4.1-mini	93.0	82.0	85.0	68.0	50.0	16.7	41.9	36.0	**5.0**
Gemini-2.0-Flash	91.0	90.0	84.0	**88.0**	67.0	8.7	23.9	30.0	3.0
Claude-3.7-Sonnet	**97.0**	**98.0**	86.0	87.0	**79.0**	**18.3**	**52.5**	43.0	2.0
GPT-4o-mini	93.0	93.0	87.0	73.0	66.0	15.1	28.9	45.0	5.0
Simple RAG Agents									
BM25	89.0	89.0	**84.0**	**62.0**	**53.0**	13.6	20.9	**44.0**	5.0
Embedding RAG Agents									
Contriever	89.0	88.0	80.0	55.0	41.0	15.2	21.2	25.0	5.0
Text-Embed-3-Small	88.0	89.0	83.0	54.0	36.0	15.3	**25.7**	21.0	4.0
Text-Embed-3-Large	**90.0**	**91.0**	80.0	55.0	46.0	**16.2**	21.6	22.0	3.0
NV-Embed-v2	88.0	89.0	82.0	40.0	48.0	13.5	20.7	42.0	**6.0**
Structure-Augmented RAG Agents									
RAPTOR	78.0	75.0	73.0	48.0	23.0	12.3	13.4	19.0	2.0
GraphRAG	64.0	54.0	49.0	24.0	6.0	9.8	0.4	10.0	3.0
HippoRAG-v2	81.0	86.0	73.0	38.0	29.0	10.2	14.6	29.0	3.0
Mem0	5.0	4.0	1.0	6.0	1.0	10.0	0.8	18.0	2.0
Cognee	34.0	42.0	42.0	41.0	18.0	10.1	2.3	28.0	3.0
Agentic Memory Agents									
Self-RAG	19.0	13.0	6.0	15.0	5.0	12.8	0.9	14.0	2.0
MemGPT	89.0	83.0	79.0	56.0	31.0	14.0	2.5	13.0	3.0

Meanwhile, the overall performance of Embedding RAG Agents surpasses that of both Structure-Augmented RAG Agents and Agentic Memory Agents. This advantage is primarily attributed to the use of dense retrieval in Embedding RAG Agents. It enables the extraction of longer contextual information from memory. As a result, Embedding RAG Agents are able to provide richer and more comprehensive context for tasks.

D.2. Detailed Results on TTL, LRU and CR

We give detailed results on each dataset in Table 5. For all three types of tasks, RAG-based agents generally underperform compared to their respective GPT-4o-mini backbones. This observation highlights certain limitations inherent to the RAG approach. For instance, in TTL tasks, RAG-based methods often struggle to accurately retrieve context from memory that is closely associated with the input. In LRU tasks, these methods face challenges in achieving a comprehensive understanding of long contexts. Furthermore, for CR tasks—especially the multi-hop variants—effective handling requires strong reasoning and information extraction capabilities, which remain beyond the reach of most current agents.

E. Ablation Study



(a) RULER-QA performance

(b) ∞Bench-Sum performance

Figure 5: Performances on RULER-QA with different chunk sizes.

E.1. Input Chunk Size

In our main experiments, we fix the chunk size to 4096 tokens, as input sequences often exceed 500k tokens, and constructing memory with smaller chunk sizes (e.g., 512) would be prohibitively time-consuming. However, to understand how chunk size impacts performance, particularly for retrieval-augmented generation (RAG) methods and agentic memory agents, we conduct an additional analysis where we vary the chunk size while fixing the number of retrieved chunks to 10. The results are presented in Figure 5. From the figure, we observe the following: (1) In the RULER-QA task, reducing chunk size has little effect on BM25 performance. This is expected, as BM25 relies on string matching and does not benefit from finer-grained segmentation. In contrast, embedding-based methods—including MemGPT, which uses `text-embedding-3-small` as its retriever—consistently perform better with smaller chunks. This suggests that finer segmentation improves the granularity and relevance of retrieved results for these models. (2) In ∞Bench-Sum, however, smaller chunk sizes lead to worse performance. This task requires the agent to summarize an entire conversation, and smaller chunks correspond to fewer available tokens per retrieval. As a result, the agent has access to less context, which degrades summarization quality. The results suggest that, when resources permit, using smaller chunk sizes and increasing the number of retrieval calls during memory construction can improve performance on Accurate Retrieval (AR) tasks. Finer-grained segmentation enhances the relevance of retrieved information, particularly for embedding-based methods. However, for tasks requiring Long-Range Understanding (LRU), varying the chunk size hurts the performance. This is likely because RAG methods are inherently less suited for tasks that demand integration of information across a large, coherent context.

E.2. Retrieval TopK

In our experiments, although we report results with the number of retrieved chunks set to 10 in Table 1, we also conducted ablation studies with varying retrieval sizes. A subset of these results is visualized in Figure 6. The results indicate that increasing the number of retrieved chunks generally improves performance across most tasks. It is worth noting that, with a

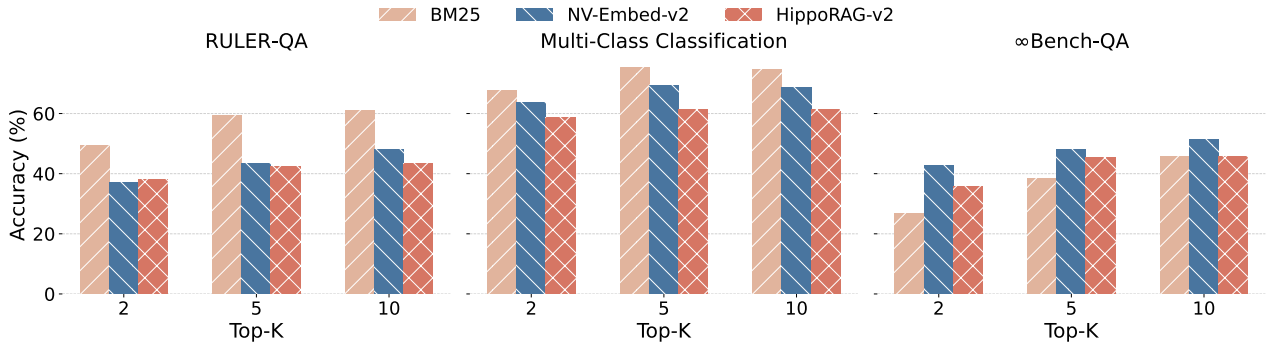


Figure 6: The accuracies on different benchmarks when varying the retrieval top-k to be 2, 5 and 10.

chunk size of 4096 tokens, retrieving 10 chunks already yields an input of approximately 40k tokens. This places significant demands on model capacity. Due to this high token volume, we do not evaluate settings with 20 retrieved chunks.

E.3. Validation of Dataset FactConsolidation

As the performance of different models on this dataset remains drastically low, we turn to the stronger reasoning model o4-mini and validate our dataset by checking the performance of o4-mini on a smaller version of this dataset. The results are shown in Table 6.

Table 6: Performances of reasoning models on the dataset FactConsolidation.

	FactCon-SH		FactCon-MH	
	6K	32K	6K	32K
GPT-4o	92.0	88.0	28.0	10.0
O4-mini	**100.0**	61.0	**80.0**	14.0

E.4. Analysis of Computational Latency

To illustrate the latency of various memory agents in terms of (1) Memory Construction (M.C.); (2) Query Execution (Q.E.), we randomly choose 5 examples from RULER-QA2 and LME (S*), and report the latency of various memory agents. This part of experiments is done on a server with Four NVDIA L40 GPU and AMD EPYC 7713 64-Core CPU. We use the NV-Embed-v2 (7B) as the embedding model in HippoRAG-v2. We show the summarized results in Table 7 and put the full results in Table 8 and 9. From the table, we find that using a smaller chunk size requires significantly more time for memory construction, especially for methods such as HippoRAG-v2, Mem0, Cognee, and MemGPT. Meanwhile, methods such as Mem0, Cognee need extremely high resources when constructing the memory, which may pose challenges in real-world applications.

Table 7: Computational Latency (in seconds).

	512		4096	
	M.C.	Q.E.	M.C.	Q.E.
GPT-4o-mini	0.09	5.2	0.07	5.1
BM25	0.11	0.79	0.10	1.8
Contriever	7.2	0.76	1.7	2.0
Text-Embed-3-Large	6.3	0.54	5.4	1.8
NV-Embed-v2	93.4	0.83	42.9	1.8
RAPTOR	151	0.51	133	0.60
GraphRAG	123	9.9	90.4	9.4
HippoRAG-v2	817	1.1	284	2.6
Mem0	14644	1.2	2140	1.2
Cognee	8309	33.2	962	4.5
Self-RAG	8.4	2.0	6.7	1.7
MemGPT	413	10.6	93.3	11.4

F. Experimental Settings

In this section, we present the experimental settings.

F.1. Max Output Tokens

We provide the token number limitation for each task in Table 10.

F.2. Settings of the RAG Agents

For the embedding model selection in Structure-Augmented RAG Agents and Agentic Memory Agents, most approaches utilize OpenAI's embedding models, such as Text-Embed-3-Small. While for the HippoRAG-v2 method, we follow the same experimental setting as in Gutiérrez et al. (2025), employing the NV-Embed-v2 model.

Table 8: Computational latency (in seconds) comparison on Long-Context Agents.

	RULER-QA2	LME (S*)
GPT-4o	17.0	20.1
GPT-4o-mini	4.9	5.4
GPT-4.1-mini	9.0	7.4
Gemini-2.0-Flash	12.4	10.1
Claude-3.7-Sonnet	23.3	22.7

Table 9: Computational latency (in seconds) comparison on RAG based agents. M.C. means Memory Construction and Q.E. means Query Execution.

	RULER-QA2				LME (S*)			
	512		4096		512		4096	
	M.C.	Q.E.	M.C.	Q.E.	M.C.	Q.E.	M.C.	Q.E.
BM25	0.12	0.47	0.11	1.7	0.09	1.1	0.08	1.9
Contriever	7.4	0.59	1.7	2.0	6.9	0.92	1.6	1.9
Text-Embed-3-Large	6.1	0.46	5.0	1.7	6.5	0.62	5.8	1.8
NV-Embed-v2	102	0.63	47.0	1.8	85.1	1.0	38.8	1.7
RAPTOR	193	0.41	161	0.67	108	0.60	104	0.53
GraphRAG	97.8	12.8	91.9	10.9	149	7.0	88.8	7.8
HippoRAG-v2	1089	0.71	380	1.71	544	1.5	188	3.5
Mem0	10804	0.79	1334	0.65	18483	1.6	2946	1.7
Cognee	11890	58.7	1185	4.8	4728	7.7	738	4.1
Self-RAG	11.4	3.1	8.1	2.4	5.3	0.82	5.2	1.0
MemGPT	433	9.4	101	10.5	392	11.7	85.5	12.3

We implement three open-sourced memory agents in our main experiments. (1) For Mem0, we use **memory.add()** function to add the message with the content from each context chunk into the agent's memory repository during memory consolidation. During query execution, the relevant memory elements are retrieved through **memory.search()** function. The retrieved memories are then integrated into the query before being processed by the GPT-4o-mini backbone model to complete the requested tasks. (2) For MemGPT, we employ the **insert_passage()** function during the memory consolidation phase to inject long context chunks into the Archival Memory structure. During query execution, this agent processes requests via the **send_message()** function which generates appropriate responses based on the archived information. (3) For Cognee, we utilize the **cognee.add()** and **cognee.cognify()** functions to construct the memory graph from input chunks wherein the memory consolidation phase. During query execution, the **cognee.search()** function is used to retrieve contextually relevant information from the memory graph based on the input query.

Table 10: Maximum output token limits for various tasks

Task	Max Output Tokens
RULER-QA	50
RULER-NIAH-MQ	100
∞ Bench-QA	10
LongMemEval	100
EventQA	40
ICL_Five	20
Movie Recommendation	300
∞ Bench-Sum	1,200
FactConsolidation	10