MEMO: Memory as a Model

Ryan Wei Heng Quek [*1 2] **Sanghyuk Lee** [*3 4 5] **Alfred Wei Lun Leong** [*1 6] **Arun Verma** [*7]
Alok Prakash [7] **Nancy F. Chen** [2] **Bryan Kian Hsiang Low** [1 7] **Daniela Rus** [5 7] **Armando Solar-Lezama** [5 7]

Abstract

Large language models (LLMs) achieve strong performance across a wide range of tasks, but remain frozen after pretraining until subsequent updates. Many real-world applications require timely, domain-specific information, motivating the need for efficient mechanisms to incorporate new knowledge. In this paper, we introduce **MEMO** (**Me**mory as a **Mo**del), a modular framework that encodes new knowledge into a dedicated MEMORY model while keeping the LLM parameters unchanged. Compared to existing methods, MEMO offers several advantages: **(a)** it captures complex cross-document relationships, **(b)** it is robust to retrieval noise, **(c)** it avoids catastrophic forgetting in the LLM, **(d)** it does not require access to the LLM's weights or output logits, enabling plug-and-play integration with both open and proprietary closed-source LLMs, and **(e)** its retrieval cost is independent of corpus size at inference time. Our experimental results on three benchmarks, BrowseComp-Plus, NarrativeQA, and MuSiQue, show that MEMO achieves strong performance compared to existing methods across diverse settings. The code is publicly available in the MeMo GitHub repository.

1. Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across diverse tasks (Kojima et al., 2023; Zhao et al., 2023; Jiang et al., 2026). Despite their successes, these models are effectively *frozen* for extended periods after pretraining (Xu et al., 2024) until subsequent updates, causing their pretrained knowledge to become increasingly outdated as the world evolves. For applications that require up-to-date (Cheng et al., 2024; Kasai et al., 2024) or domain-specific (Singhal et al., 2022; Wu et al., 2023) knowledge, this dependence on static knowledge presents a fundamental architectural limitation (Lewis et al., 2021; Kandpal et al., 2023). Retraining is a natural solution but remains prohibitively expensive at modern scales (Wu et al., 2022a), motivating the need for an efficient mechanism to integrate new external knowledge into LLMs without full retraining.

Existing methods for integrating new knowledge into LLMs fall into three categories. ① *Non-parametric methods* retrieve relevant information from an external store at inference time via lexical (Robertson & Walker, 1994), dense (Lee et al., 2024), or graph-based retrievers (Lewis et al., 2020; Edge et al., 2024; Gutiérrez et al., 2024; 2025), before incorporating it through in-context learning (Brown et al., 2020; Dong et al., 2024). However, these methods are constrained by limited context windows and struggle to synthesize cross-document relationships when relevant information is distributed across multiple documents (Tang & Yang, 2024; Lin et al., 2025). ② *Parametric methods* internalize knowledge directly into model parameters via continual pretraining (Ke et al., 2023) or fine-tuning (Ouyang et al., 2022; Wang et al., 2023; Chung et al., 2024) on the target corpus directly. While effective, they are computationally expensive, prone to catastrophic forgetting (Luo et al., 2025), and tend to memorize training distributions rather than acquire transferable knowledge, limiting generalization to unseen queries (Chu et al., 2025). ③ *Latent memory methods* (Chevalier et al., 2023; Mu et al., 2023; Ge et al., 2024; Zhang et al., 2026a) compress knowledge into soft tokens or other model-specific representations, but suffer from representation coupling: the memory is tightly bound to the specific model used to produce these representations, limiting transferability across LLMs.

We introduce **MEMO** (**Me**mory as a **Mo**del), a modular framework where a dedicated MEMORY model is trained on new knowledge, and an EXECUTIVE model retrieves relevant information from the MEMORY model at inference time via targeted sub-queries and then reasons over the retrieved information to respond to user queries. MEMO combines the complementary strengths of the three paradigms above while mitigating their individual limitations. Like the

[*]Equal contribution [1]National University of Singapore, Singapore [2]Agency for Science, Technology, Research (A*STAR), Singapore [3]University of Tokyo, Japan [4]Liquid AI, USA [5]CSAIL, Massachusetts Institute of Technology, USA [6]AI Singapore [7]Singapore-MIT Alliance for Research and Technology Centre. Correspondence to: Arun Verma <arun.verma@smart.mit.edu>.

Presented at the ICML 2026 Workshop "Continual Adaptation at Scale: Towards Sustainable AI". Copyright 2026 by the author(s).

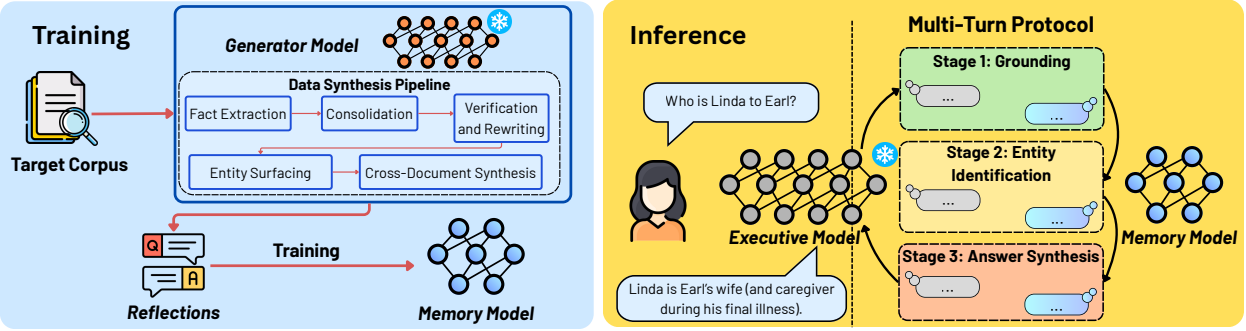

Figure 1. **Overview of the training and inference pipeline of MEMO.** During MEMORY model training (left), a frozen GENERATOR model transforms a target corpus into a reflection QA dataset via fact extraction, consolidation, verification, entity surfacing, and cross-document synthesis, which is then used to train a dedicated MEMORY model. During inference (right), the frozen EXECUTIVE model answers complex user queries by querying the MEMORY model through a structured multi-turn protocol: it decomposes the input into targeted sub-queries, retrieves responses from the MEMORY model, and reasons over them to produce an answer to the user's query.

non-parametric methods, it is able to leverage off-the-shelf frontier models unchanged by separating the memory from the reasoning model; it shares with the parametric methods the ability to internalize knowledge in model parameters, and it shares the benefits of a compact, queryable memory artifact with latent memory methods. As a result, MEMO offers the following advantages: **(a)** it captures complex cross-document relationships, **(b)** it is robust to retrieval noise, **(c)** it avoids catastrophic forgetting by keeping the EXECUTIVE model parameters unchanged, **(d)** it does not require access to the EXECUTIVE model's weights or output logits, enabling plug-and-play integration with both open and proprietary LLMs, and **(e)** its retrieval cost is independent of corpus size at inference time due to the fixed size of the MEMORY model. However, designing MEMO to comprehensively capture cross-document relationships during training while accurately answering arbitrary queries at inference time introduces two key challenges, which we outline below and address them with novel methods.

① **Training MEMORY model.** A core challenge in the MEMORY model is ensuring it can accurately answer diverse, unseen queries at inference time, including those requiring cross-document reasoning and long-context understanding. A natural approach is to train directly on the raw corpus using standard data augmentation techniques such as paraphrasing (Li et al., 2022; Chen et al., 2023; Allen-Zhu & Li, 2024), additional sampling of generated QA pairs (Alberti et al., 2019; Puri et al., 2020), or targeted gap-filling, where the model identifies and completes missing knowledge from the corpus (Feng et al., 2024; Jie et al., 2024). However, these approaches fail to consolidate related facts into compositional representations necessary for robust generalization to unseen queries (Chu et al., 2025). With this challenge in mind, we design a novel *five-step data synthesis pipeline* guided by a GENERATOR model (Sec. C.1) that distills the corpus into a question–answer (QA) dataset of *reflections*: compositional representations that expose underlying corpus knowledge under diverse query variations (illustrated in Fig. 1 (left) and more details are given in Sec. C.1). We train MEMORY model on the synthesized reflection QA dataset via supervised fine-tuning (see Sec. C.2), enabling MEMORY model to capture more complex, cross-document relationships and compositional structure than retrieval-based methods.

② **Querying MEMORY model.** At inference time, complex or compositional queries often require multi-step reasoning and aggregation of information across multiple documents. Naively querying MEMORY model via single-turn or unstructured multi-turn interactions fails to reliably retrieve the knowledge required to answer such queries. To address this, we design a *three-stage inference pipeline* in which EXECUTIVE model queries and retrieves information from MEMORY model via a structured multi-turn protocol, decomposing complex user queries into targeted sub-queries that align with the shared reflection interface (illustrated in Fig. 1 (right); further details are given in Sec. C.4). Unlike retrieval-based methods, this approach incurs a per-query retrieval cost that is independent of corpus size and is robust to retrieval noise (see Sec. 2.2). Crucially, because MEMO treats EXECUTIVE model as a black box and does not access its weights, gradients, or output logits, it supports *plug-and-play* integration with *any LLM*, including both open-source and proprietary closed-source models.

Our method is guided by a single design principle: *reflections*, corpus-derived structures that require no knowledge of future queries, yet naturally serve as the precise interface through which any query can access the underlying corpus without ever observing it directly. During training, the MEMORY model internalizes these reflections; EXECUTIVE model retrieves relevant knowledge through targeted sub-queries at inference time. Building on the challenges outlined above and the methods proposed to address them, we summarize the key contributions of this paper as follows:

- **Novel data synthesis pipeline.** We propose a five-step data synthesis pipeline that uses a GENERATOR model, an

LLM that may be smaller than EXECUTIVE model, to distill a target corpus into reflections, enabling a dedicated MEMORY model to internalize knowledge in compositional forms that capture complex cross-document relationships and generalize robustly to diverse, unseen query variations at inference time (see Sec. C.1 and Sec. C.2).

- **Structured multi-turn protocol.** We introduce a *structured* multi-turn protocol that systematically decomposes complex queries into targeted sub-queries aligned with the shared reflection interface. The protocol supports plug-and-play integration with any arbitrary LLM, including proprietary closed-source LLMs, and has retrieval cost independent of corpus size (see Sec. C.4).
- **Empirical validation.** We evaluate MEMO on BrowseComp-Plus, NarrativeQA, and MuSiQue, demonstrating strong performance against both parametric and non-parametric baselines. We further empirically validate MEMO's robustness to retrieval noise (see Sec. 2).

2. Experiments

Datasets. We evaluate MEMO on three knowledge-intensive benchmarks. **BrowseComp-Plus** (Chen et al., 2025) is a deep-research benchmark requiring multi-hop, multi-document retrieval and reasoning; we filter non-English instances with LangDetect (Danilák, 2021), sample 300 questions, and pair each question's evidence documents with an equal number of negative documents,[1] yielding 3,541 documents. **NarrativeQA** (Kočiskỳ et al., 2018) tests discourse understanding over long documents such as books and movie scripts; we use 293 questions across 10[2] documents. **MuSiQue** (Trivedi et al., 2022) requires composing 2–4 reasoning steps across multiple Wikipedia paragraphs; we use 1,000 questions and construct the target corpus following the same procedure as for BrowseComp-Plus, yielding 5,296 documents. Further details and additional experiments are provided from App. D to App. P.

Baselines. We compare MEMO against four baselines: **BM25** (Robertson & Walker, 1994) (lexical retrieval), **NV-Embed-V2** (Lee et al., 2024) (dense retrieval), **HippoRAG2** (Gutiérrez et al., 2025) (graph-based RAG, state-of-the-art), and **Cartridges** (Eyuboglu et al., 2025) (a trained KV-cache loaded onto EXECUTIVE model at inference; the closest existing parametric baseline to MEMO). Newer methods exist (Chevalier et al., 2023; Cao et al., 2025b) but typically require white-box access to EXECUTIVE model and are therefore not directly comparable. We additionally include **Perfect Retrieval** as an *empirical upper bound*, where EXECUTIVE model receives exclusively the evidence documents in context (Brown et al., 2020). Retrieval baselines use top-k=9 with adaptive backoff: reducing k progressively until the retrieved context fits EXECUTIVE model's context window.

Implementation and evaluation. *(a) Data generation.* We use Qwen2.5-32B-Instruct (Yang et al., 2025) as the GENERATOR model, served via vLLM (Kwon et al., 2023) with YaRN RoPE scaling (Su et al., 2024; Peng et al., 2024) to support a 131K-token context window during long-context generation. *(b) Training.* We train MEMORY model, which is initialized from Qwen2.5-14B-Instruct for 3 epochs with fused AdamW (Loshchilov & Hutter, 2017) and DeepSpeed 2 (Rajbhandari et al., 2020) at learning rate 2×10^{-5}; full hyperparameters are provided in App. G. *(c) Evaluation.* We instantiate EXECUTIVE model with either Qwen2.5-32B-Instruct or Gemini-3-Flash (Google DeepMind, 2025) to evaluate the same trained MEMORY model across models of varying reasoning capability; both models have minimal prior knowledge of the evaluation datasets (App. J). EXECUTIVE model queries MEMORY model through the multi-turn protocol described in Sec. C.4. We report binary accuracy judged by Gemini-2.5-Flash-Lite (Comanici et al., 2025) via DeepEval (Ip & Vongthongsri, 2025), as mean $\pm$ standard deviation over three runs for Qwen2.5-32B-Instruct and a single run for Gemini-3.0-Flash. *(d) Continual integration.* For the model-merging experiment (App. D.3), we partition NarrativeQA into two pairwise-disjoint subsets (K=2, with $\sim$640k QA pairs each), SFT a separate Qwen2.5-14B-Instruct MEMORY model on each, and sweep six merging methods at three densities (yielding 14 configurations total).

2.1. Experimental Results

MEMO achieves strong performance across benchmarks. As shown in Table 1, MEMO consistently outperforms all baselines on NarrativeQA and MuSiQue across both EXECUTIVE models. On NarrativeQA, the most challenging benchmark (App. J), MEMO achieves 26.85% with Qwen2.5-32B-Instruct and 53.58% with Gemini-3-Flash, substantially surpassing all baselines. This is notable: NarrativeQA requires reasoning over long passages with complex connections, where retrieval-based methods are constrained by context windows and struggle to synthesize information across long documents; MEMO instead captures these connections via reflections during training and retrieves them through its multi-turn protocol at inference. The same trend holds on MuSiQue, where MEMO achieves 48.30% and 58.70%, respectively, outperforming baselines that struggle with multi-hop reasoning across independently retrieved passages. On BrowseComp-Plus, MEMO leads with Gemini-3-Flash (66.67%) and remains competitive with Qwen2.5-32B-Instruct (54.22%, narrowly trailing Hip-

[1] BrowseComp-Plus and MuSiQue provide annotations of gold (correct), evidence (supporting), and negative (distractor) documents. Gold documents are a subset of the evidence documents.

[2] We follow HippoRAG2 and evaluate on 10 such documents from the NarrativeQA validation split (294 questions); one duplicate is removed for consistency.

poRAG2's 56.11%). This gap reflects BrowseComp-Plus's nature: its answers are absent from EXECUTIVE model's parametric knowledge (App. J), making direct access to evidence documents especially valuable and favoring retrieval methods that pass raw documents to EXECUTIVE model.

Table 1. Accuracy (%) on BrowseComp-Plus, NarrativeQA, and MuSiQue under two EXECUTIVE models: Qwen2.5-32B-Instruct (Qwen2.5-32B-I) and Gemini-3-Flash (Gemini-3-F). Bold values indicate the best result in each column, excluding Perfect Retrieval. MEMO uses Qwen2.5-14B-Instruct as MEMORY model, and results are reported at the best training epoch. *Perfect Retrieval represents an empirical upper bound.

Method	BrowseComp-Plus		NarrativeQA		MuSiQue	
	Qwen2.5-32B-I	Gemini-3-F	Qwen2.5-32B-I	Gemini-3-F	Qwen2.5-32B-I	Gemini-3-F
Perfect Retrieval*	79.67 ± 1.45	88.33	51.54 ± 0.34	59.39	51.53 ± 0.74	66.90
BM25	1.11 ± 0.69	27.00	10.24 ± 0.34	14.33	20.00 ± 0.30	23.20
NV-Embed-V2	50.67 ± 0.33	57.00	20.59 ± 0.86	26.62	37.47 ± 0.15	46.60
HippoRAG2[3]	$\mathbf{56.11 \pm 0.51}$	66.33	21.39 ± 0.20	23.21	42.17 ± 0.12	57.00
Cartridges[4]	1.22 ± 0.69	-	13.65 ± 1.23	-	7.53 ± 0.23	-
MEMO	54.22 ± 0.84	**66.67**	$\mathbf{26.85 \pm 0.39}$	**53.58**	$\mathbf{48.30 \pm 1.25}$	**60.20**

MEMO supports plug-and-play integration. Across the three benchmarks, MEMO consistently achieves higher performance when paired with a more capable EXECUTIVE model (Gemini-3-Flash): switching from Qwen2.5-32B-Instruct to Gemini-3-Flash yields gains of 12.45%, 26.73%, 11.90% on BrowseComp-Plus, NarrativeQA and MuSiQue, respectively. This demonstrates that MEMO can be trained once with a weaker GENERATOR model, and seamlessly paired with *any* LLM at inference, including proprietary models such as Gemini-3-Flash. This *plug-and-play* capability allows MEMO to directly leverage state-of-the-art models without any additional modification to these models.

2.2. Ablation on the Amount of Noise for the Dataset

We investigate the robustness of MEMO against two strong retrieval-based baselines, NV-Embed-V2 and HippoRAG2, under increasing levels of retrieval noise, controlled by varying the number of negative (distractor) documents added to the target corpus as a multiple of the total number of ground-truth evidence documents in each dataset ($N^{\text{dataset}}_{\text{evidence}} = 1{,}775$ for BrowseComp-Plus and $N^{\text{dataset}}_{\text{evidence}} = 2{,}648$ for MuSiQue). The datasets used throughout this paper (detailed in App. E) correspond to a ratio of $1 \times N^{\text{dataset}}_{\text{evidence}}$; we additionally evaluate at ratio $0 \times N^{\text{dataset}}_{\text{evidence}}$ (no distractors) as an idealized noise-free reference to isolate the effect of distractors.

Results in Tab. 2 demonstrate that retrieval-based methods exhibit pronounced sensitivity to noise. Both NV-Embed-V2 and HippoRAG2 suffer drops of up to 6.22% on BrowseComp-Plus and up to 5.16% on MuSiQue when scaling from $0 \times N$ to $1 \times N$, confirming that these systems struggle to filter irrelevant documents under realistic corpus conditions. In contrast, MEMO maintains stable performance across both benchmarks, with a marginal improvement of 0.55% on BrowseComp-Plus and a decline of only 1.77% on MuSiQue, both within one standard deviation, demonstrating that MEMO is robust to increasing retrieval noise. We attribute this robustness to MEMO's design: despite being trained on a corpus containing negative documents, MEMORY model provides more precise information to EXECUTIVE model's sub-queries than direct document retrieval. Additional analysis of performance degradation in retrieval-based methods is provided in App. M.

Table 2. Accuracy (%) on BrowseComp-Plus and MuSiQue with Qwen2.5-32B-Instruct as EXECUTIVE model. MEMO results are based on Qwen2.5-14B-Instruct and reported at the best training epoch. $N = N^{\text{dataset}}_{\text{evidence}}$ denotes the number of ground-truth evidence documents in the corpus; column headers indicate the number of additional negative (distractor) documents added, as a multiple of N. Δ denotes accuracy difference (%) compared to $0N$.

Method	Dataset	$0 \times N$	$1 \times N$	
		Acc. (%)	Acc. (%)	Δ
NV-Embed-V2	BrowseComp-Plus	56.89 ± 0.51	50.67 ± 0.33	$\downarrow 6.22$
	MuSiQue	42.30 ± 0.53	37.47 ± 0.15	$\downarrow 4.83$
HippoRAG2	BrowseComp-Plus	62.33 ± 1.15	56.11 ± 0.51	$\downarrow 6.22$
	MuSiQue	47.33 ± 0.74	42.17 ± 0.12	$\downarrow 5.16$
MEMO	BrowseComp-Plus	53.67 ± 1.15	54.22 ± 0.84	$\uparrow 0.55$
	MuSiQue	50.07 ± 0.81	48.30 ± 1.25	$\downarrow 1.77$

3. Conclusion

We introduced MEMO, a modular framework for integrating updated or domain-specific knowledge into LLMs via a MEMORY model trained on a synthesized reflection QA dataset. MEMO addresses key limitations of existing methods: it bypasses context constraints and limited cross-document reasoning in retrieval-based approaches, avoids costly and brittle parametric updates (including catastrophic forgetting), and removes representation coupling in latent memory methods. Its core components are a data synthesis pipeline capturing explicit facts and implicit relationships, and a multi-turn inference protocol that decomposes complex queries into targeted sub-queries for retrieving the desired information from the memory model. While MEMO demonstrates strong performance, it has limitations regarding training cost, evaluation scope, and the capacity of MEMORY model to scale with corpus size (see App. R). Empirically, MEMO outperforms strong baselines across diverse benchmarks. It also provides a scalable pathway for knowledge integration, supporting efficient updates and plug-and-play deployment with both open-source and proprietary closed-source LLMs. Future work includes more efficient memory construction, extensions to dynamic corpora, and tighter coordination between the EXECUTIVE model and MEMORY model. We view **MEMO** (**Me**mory as a **Mo**del) as a promising foundation for more flexible, updatable, and knowledge-aware AI systems.

[3]These results differ from Gutiérrez et al. (2025), which uses Llama3.3-70B-Instruct instead of Qwen2.5-32B-Instruct.

[4]Cartridges requires white-box access to EXECUTIVE model as well; its results for Gemini-3-Flash are therefore omitted.

ACKNOWLEDGMENTS

This research is supported by the National Research Foundation (NRF), Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. The Mens, Manus, and Machina (M3S) is an interdisciplinary research group (IRG) of the Singapore MIT Alliance for Research and Technology (SMART) centre. We would like to acknowledge that computational work involved in this research work is partially supported by NUS IT's Research Computing group under grant number NUSREC-HPC-00004. This research is also supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-PhD-2025-08-061). Ryan Wei Heng Quek is supported by Agency for Science, Technology and Research (A*STAR) Graduate Academy. The authors would also like to thank Zhaoxuan Wu, Zijian Zhou, and Rui Qiao for the initial discussions related to this work while they were working at the SMART Centre.

References

Alberti, C., Andor, D., Pitler, E., Devlin, J., and Collins, M. Synthetic qa corpora generation with roundtrip consistency. In *Proc. ACL*, pp. 6168–6173, 2019.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.2, knowledge manipulation. *arXiv:2309.14402*, 2023.

Allen-Zhu, Z. and Li, Y. Physics of language models: part 3.1, knowledge storage and extraction. In *Proc. ICML*, pp. 1067–1077, 2024.

Amini, A., Banaszak, A., Benoit, H., Böök, A., Dakhran, T., et al. LFM2 technical report. *arXiv:2511.23404*, 2025.

Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A. C., Korbak, T., and Evans, O. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv:2309.12288*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *Proc. NeurIPS*, pp. 1877–1901, 2020.

Cao, J., Wang, J., Wei, R., Guo, Q., Chen, K., Zhou, B., and Lin, Z. Memory decoder: A pretrained, plug-and-play memory for large language models. *arXiv:2508.09874*, 2025a.

Cao, J., Wang, J., Wei, R., Guo, Q., Chen, K., Zhou, B., and Lin, Z. Memory decoder: A pretrained, plug-and-play memory for large language models. *arXiv:2508.09874*, 2025b.

Chen, J., Tam, D., Raffel, C., Bansal, M., and Yang, D. An empirical survey of data augmentation for limited data learning in nlp. *Transactions of the Association for Computational Linguistics*, pp. 191–211, 2023.

Chen, Z., Ma, X., Zhuang, S., Nie, P., Zou, K., Liu, A., Green, J., Patel, K., Meng, R., Su, M., Sharifymoghaddam, S., Li, Y., Hong, H., Shi, X., Liu, X., Thakur, N., Zhang, C., Gao, L., Chen, W., and Lin, J. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv:2508.06600*, 2025.

Cheng, J., Marone, M., Weller, O., Lawrie, D., Khashabi, D., and Durme, B. V. Dated data: Tracing knowledge cutoffs in large language models. *arXiv:2403.12958*, 2024.

Chevalier, A., Wettig, A., Ajith, A., and Chen, D. Adapting language models to compress contexts. In *Proc. EMNLP*, 2023.

Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., and Ma, Y. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv:2501.17161*, 2025.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, pp. 1–53, 2024.

Comanici, G., Bieber, E., et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv:2507.06261*, 2025.

Cuconasu, F., Trappolini, G., Siciliano, F., Filice, S., Campagnano, C., Maarek, Y., Tonellotto, N., and Silvestri, F. The power of noise: Redefining retrieval for rag systems. In *Proc. SIGIR*, 2024.

Danilák, M. langdetect. https://github.com/Mimino666/langdetect, 2021.

Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Chang, B., et al. A survey on in-context learning. In *Proc. EMNLP*, 2024.

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:2404.16130*, 2024.

Eyuboglu, S., Ehrlich, R., Arora, S., Guha, N., Zinsley, D., Liu, E., Tennien, W., Rudra, A., Zou, J., Mirhoseini, A., et al. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv:2506.06266*, 2025.

Feng, S., Shi, W., Wang, Y., Ding, W., Balachandran, V., and Tsvetkov, Y. Don't hallucinate, abstain: Identifying llm knowledge gaps via multi-llm collaboration. In *Proc. ACL*, pp. 14664–14690, 2024.

Ge, T., Jing, H., Wang, L., Wang, X., Chen, S.-Q., and Wei, F. In-context autoencoder for context compression in a large language model. In *Proc. ICLR*, 2024.

Gelada, C., Buckman, J., Zhang, S., and Bach, T. Scaling context requires rethinking attention. *arXiv:2507.04239*, 2025.

Google DeepMind. Gemini 3 flash model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf, December 2025.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv:2312.00752*, 2023.

Gutiérrez, B. J., Shu, Y., Gu, Y., Yasunaga, M., and Su, Y. Hipporag: Neurobiologically inspired long-term memory for large language models. In *Proc. NeurIPS*, pp. 59532–59569, 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. In *Proc. ICML*, 2025.

Harmon, J., Hochlehnert, A., Bethge, M., and Prabhu, A. Mapping post-training forgetting in language models at scale. *arXiv:2510.17776*, 2025.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., and Ginsburg, B. RULER: What's the real context size of your long-context language models? In *Proc. COLM*, 2024.

Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. In *Proc. ICLR*, 2022.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. In *Proc. ICLR*, 2023.

Ip, J. and Vongthongsri, K. deepeval. https://github.com/confident-ai/deepeval, 2025.

Jiang, J., Wang, F., Shen, J., Kim, S., and Kim, S. A survey on large language models for code generation. *ACM Transactions on Software Engineering and Methodology*, 2026.

Jie, Y. W., Ferdinan, T., Kazienko, P., Satapathy, R., and Cambria, E. Self-training large language models through knowledge detection. In *Proc. EMNLP Findings*, pp. 15033–15045, 2024.

Kandpal, N., Deng, H., Roberts, A., Wallace, E., and Raffel, C. Large language models struggle to learn long-tail knowledge. *arXiv:2211.08411*, 2023.

Kasai, J., Sakaguchi, K., Takahashi, Y., Bras, R. L., Asai, A., Yu, X., Radev, D., Smith, N. A., Choi, Y., and Inui, K. Realtime qa: What's the answer right now? *arXiv:2207.13332*, 2024.

Ke, Z., Shao, Y., Lin, H., Konishi, T., Kim, G., and Liu, B. Continual pre-training of language models. *arXiv:2302.03241*, 2023.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. Generalization through memorization: Nearest neighbor language models. In *Proc. ICLR*, 2020.

Kočiskỳ, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, pp. 317–328, 2018.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *arXiv:2205.11916*, 2023.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. *arXiv:2309.06180*, 2023.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L. J. V., Liu, A., Dziri, N., Lyu, S., et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv:2411.15124*, 2024.

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:2405.17428*, 2024.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proc. NeurIPS*, pp. 9459–9474, 2020.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv:2005.11401*, 2021.

Li, B., Hou, Y., and Che, W. Data augmentation approaches in natural language processing: A survey. *AI Open*, pp. 71–90, 2022.

Li, Z. and Hoiem, D. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2935–2947, 2018.

Lin, J., Liu, J., and Liu, Y. Optimizing multi-hop document retrieval through intermediate representations. *arXiv:2503.04796*, 2025.

Liu, J., Lin, J., and Liu, Y. Tackling the inherent difficulty of noise filtering in rag. *arXiv:2601.01896*, 2026.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, pp. 157–173, 2024.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.

Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., and Zhang, Y. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv:2308.08747*, 2025.

Manchanda, J., Boettcher, L., Westphalen, M., and Jasser, J. The open source advantage in large language models (llms). *arXiv:2412.12004*, 2025.

Mu, J., Li, X., and Goodman, N. D. Learning to compress prompts with gist tokens. In *Proc. NeurIPS*, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. In *Proc. NeurIPS*, 2022.

Ovadia, O., Brief, M., Mishaeli, M., and Elisha, O. Fine-tuning or retrieval? comparing knowledge injection in llms. In *Proc. EMNLP*, pp. 237–250, 2024.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models. In *Proc. ICLR*, 2024.

Puri, R., Spring, R., Shoeybi, M., Patwary, M., and Catanzaro, B. Training question answering models from synthetic data. In *Proc. EMNLP*, pp. 5811–5826, 2020.

Qi, X., Zeng, Y., Xie, T., Chen, P.-Y., Jia, R., Mittal, P., and Henderson, P. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *Proc. ICLR*, 2024.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pp. 1–16, 2020.

Robertson, S. E. and Walker, S. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proc. SIGIR*, 1994.

Shoemake, K. Animating rotation with quaternion curves. In *Proc. SIGGRAPH*, pp. 245–254, 1985.

Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., Payne, P., Seneviratne, M., Gamble, P., Kelly, C., Scharli, N., Chowdhery, A., Mansfield, P., y Arcas, B. A., Webster, D., Corrado, G. S., Matias, Y., Chou, K., Gottweis, J., Tomasev, N., Liu, Y., Rajkomar, A., Barral, J., Semturs, C., Karthikesalingam, A., and Natarajan, V. Large language models encode clinical knowledge. *arXiv:2212.13138*, 2022.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, pp. 127063, 2024.

Sun, Y., Wang, S., Li, Y., Feng, S., Tian, H., Wu, H., and Wang, H. ERNIE 2.0: A continual pre-training framework for language understanding. In *Proc. AAAI*, 2020.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language models. *arXiv:2307.08621*, 2023.

Sutton, R. S., Barto, A. G., et al. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

Tang, Y. and Yang, Y. MultiHop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv:2401.15391*, 2024.

Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., bastien Grill, J., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J.-T., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, A., Sharma, A., Gilady, A. M., Goedeckemeyer, A., Saade, A., Feng, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., György, A., Pinto, A. S., Das, A., Bapna, A., Miech, A., Yang, A., Paterson, A., Shenoy, A., Chakrabarti, A., Piot, B., Wu, B., Shahriari, B., Petrini, B., Chen, C., Lan, C. L., Choquette-Choo, C. A., Carey, C., Brick, C., Deutsch, D., Eisenbud, D., Cattle, D., Cheng, D., Paparas, D., Sreepathihalli, D. S., Reid, D., Tran, D., Zelle, D., Noland, E., Huizenga, E., Kharitonov, E., Liu, F.,

Amirkhanyan, G., Cameron, G., Hashemi, H., Klimczak-Plucińska, H., Singh, H., Mehta, H., Lehri, H. T., Hazimeh, H., Ballantyne, I., Szpektor, I., Nardini, I., Pouget-Abadie, J., Chan, J., Stanton, J., Wieting, J., Lai, J., Orbay, J., Fernandez, J., Newlan, J., yeong Ji, J., Singh, J., Black, K., Yu, K., Hui, K., Vodrahalli, K., Greff, K., Qiu, L., Valentine, M., Coelho, M., Ritter, M., Hoffman, M., Watson, M., Chaturvedi, M., Moynihan, M., Ma, M., Babar, N., Noy, N., Byrd, N., Roy, N., Momchev, N., Chauhan, N., Sachdeva, N., Bunyan, O., Botarda, P., Caron, P., Rubenstein, P. K., Culliton, P., Schmid, P., Sessa, P. G., Xu, P., Stanczyk, P., Tafti, P., Shivanna, R., Wu, R., Pan, R., Rokni, R., Willoughby, R., Vallu, R., Mullins, R., Jerome, S., Smoot, S., Girgin, S., Iqbal, S., Reddy, S., Sheth, S., Põder, S., Bhatnagar, S., Panyam, S. R., Eiger, S., Zhang, S., Liu, T., Yacovone, T., Liechty, T., Kalra, U., Evci, U., Misra, V., Roseberry, V., Feinberg, V., Kolesnikov, V., Han, W., Kwon, W., Chen, X., Chow, Y., Zhu, Y., Wei, Z., Egyed, Z., Cotruta, V., Giang, M., Kirk, P., Rao, A., Black, K., Babar, N., Lo, J., Moreira, E., Martins, L. G., Sanseviero, O., Gonzalez, L., Gleicher, Z., Warkentin, T., Mirrokni, V., Senter, E., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Matias, Y., Sculley, D., Petrov, S., Fiedel, N., Shazeer, N., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Alayrac, J.-B., Anil, R., Dmitry, Lepikhin, Borgeaud, S., Bachem, O., Joulin, A., Andreev, A., Hardin, C., Dadashi, R., and Hussenot, L. Gemma 3 technical report. *arXiv:2503.19786*, 2025.

Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. Musique: Multihop questions via single-hop question composition. *arXiv:2108.00573*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Proc. NeurIPS*, 2017.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions. In *Proc. ACL*, pp. 13484–13508, 2023.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proc. ICML*, pp. 23965–23998, 2022.

Wu, C.-J., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Aga, F., Huang, J., Bai, C., et al. Sustainable ai: Environmental implications, challenges and opportunities. In *Proc. MLSys*, pp. 795–813, 2022a.

Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., and Mann, G. Bloomberggpt: A large language model for finance. *arXiv:2303.17564*, 2023.

Wu, T., Luo, L., Li, Y.-F., Pan, S., Vu, T.-T., and Haffari, G. Continual learning for large language models: A survey. *arXiv:2402.01364*, 2024.

Wu, Y., Rabe, M. N., Hutchins, D., and Szegedy, C. Memorizing transformers. In *Proc. ICLR*, 2022b.

Xia, Y., Kim, J., Chen, Y., Ye, H., Kundu, S., Hao, C. C., and Talati, N. Understanding the performance and estimating the cost of llm fine-tuning. In *Proc. IISWC*, 2024.

Xu, R., Qi, Z., Guo, Z., Wang, C., Wang, H., Zhang, Y., and Xu, W. Knowledge conflicts for llms: A survey. *arXiv:2403.08319*, 2024.

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. Ties-merging: Resolving interference when merging models. In *Proc. NeurIPS*, 2023.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., et al. Qwen2.5 technical report. *arXiv:2412.15115*, 2025.

Yang, E., Shen, L., Guo, G., Wang, X., Cao, X., Zhang, J., and Tao, D. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 2024.

Yu, L., Yu, B., Yu, H., Huang, F., and Li, Y. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Proc. ICML*, 2024.

Zhang, G., Fu, M., and YAN, S. Memgen: Weaving generative latent memory for self-evolving agents. In *Proc. ICLR*, 2026a.

Zhang, L., Liu, X., Li, Z., Pan, X., Dong, P., Fan, R., Guo, R., Wang, X., Luo, Q., Shi, S., et al. Dissecting the runtime performance of the training, fine-tuning, and inference of large language models. *arXiv:2311.03687*, 2023.

Zhang, Z. Y., Verma, A., Doshi-Velez, F., and Low, B. K. H. Understanding the relationship between prompts and response uncertainty in large language models. In *Proc. ACL Findings*, 2026b.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv:2303.18223*, 2023.

A. Related Work

Non-parametric methods. Non-parametric alternatives (Robertson & Walker, 1994; Lee et al., 2024; Gutiérrez et al., 2025) avoid parameter updates entirely, instead, supplying new knowledge at inference time. In particular, in-context learning (ICL) (Brown et al., 2020; Dong et al., 2024) inserts relevant knowledge directly into the prompt, avoiding catastrophic forgetting. However, ICL scales poorly with increasing context length: the computational cost of autoregressive generation (Vaswani et al., 2017) leads to substantial token overhead and inference latency as the knowledge base grows (Gelada et al., 2025), and even explicitly long-context models exhibit significant performance degradation as context length increases (Liu et al., 2024; Hsieh et al., 2024). Retrieval-augmented generation (RAG) (Lewis et al., 2020; Edge et al., 2024; Gutiérrez et al., 2024; 2025) addresses this scalability bottleneck by selectively retrieving relevant chunks of knowledge at inference time. However, RAG systems are highly sensitive to retrieval noise (Cuconasu et al., 2024), where irrelevant or misleading passages substantially degrade generation quality (Liu et al., 2026; Zhang et al., 2026b). In addition, RAG systems often struggle to reason over complex cross-document dependencies (Tang & Yang, 2024), as they lack robust mechanisms for synthesizing information that is distributed across multiple chunks or a large corpus (Lin et al., 2025).

Parametric methods. Existing post-training approaches, such as continual pretraining on new corpora (Ke et al., 2023; Sun et al., 2020) or supervised fine-tuning (SFT) on curated instruction data (Ouyang et al., 2022; Wang et al., 2023; Chung et al., 2024), attempt to address this limitation by incorporating new knowledge into LLMs during post-training. While conceptually straightforward, these parametric methods often suffer from catastrophic forgetting, whereby adaptation to newly observed knowledge degrades previously acquired knowledge, learned capabilities (Luo et al., 2025; Li & Hoiem, 2018; Harmon et al., 2025), and safety alignment learned during LLM post-training (Qi et al., 2024). In addition, the scale of modern LLMs makes frequent fine-tuning computationally expensive (Zhang et al., 2023; Xia et al., 2024), and fine-tuning is often infeasible for proprietary, closed-source models (Manchanda et al., 2025), substantially limiting the practicality of parametric methods in real-world, large-scale applications.

Latent memory methods. Another approach to storing knowledge is via *compressed latent representations*, which lie between non-parametric retrieval and fully parametric methods. Context compression techniques such as AutoCompressor (Chevalier et al., 2023), Gist tokens (Mu et al., 2023), and ICAE (Ge et al., 2024) encode knowledge into compact soft tokens prepended at inference, reducing ICL token overhead without discarding information. However, these representations are tightly coupled to the encoder and cannot be consumed by other model families, limiting compatibility with black-box LLMs. Similarly, recurrent-state models (Gu & Dao, 2023; Sun et al., 2023) and nearest-neighbor memory methods such as Memorizing Transformers (Wu et al., 2022b) and kNN-LM (Khandelwal et al., 2020) rely on model-specific representations or architectures, preventing post hoc use with pretrained LLMs. Although Memory Decoder (Cao et al., 2025a) is a plug-and-play pretrained memory module that integrates without modifying model parameters, it is limited to architectures sharing a common tokenizer, enabling reuse only within this subset. The core limitation of these methods is *representation coupling*: latent memory is inseparable from the model that produces it. In contrast, MEMO allows a plug-and-play integration with any LLM, including closed-source models.

Table 3. A comparison of desirable properties across different memory paradigms, showing that MEMO satisfies them through its modular memory construction and memory-augmented reasoning.

Methods	Frozen base LLM	No retrieval index	Black-box compatible	No catastrophic forgetting	Constant-size memory	Cross-LLM transferable
Non-parametric (RAG, ICL)	✓	×	✓	✓	×	✓[5]
Parametric (CPT, SFT)	×	✓	×	×	✓	×
Latent memory (AutoCompressor, Gist, ICAE)	✓	✓	×	✓	×	×
MEMO (Ours)	✓	✓	✓	✓	✓	✓

B. Preliminaries

Problem setting. Let $\mathcal{M}_\theta$ denote a large language model with frozen parameters $\theta \in \mathbb{R}^p$, pretrained on a corpus $\mathcal{D}_{\text{pre}}$. We treat $\mathcal{M}_\theta$ as a conditional distribution that maps a prompt x to a response $\mathcal{M}_\theta(x)$, and assume only black-box access; in particular, $\mathcal{M}_\theta$ may be either a white-box model or a closed-source model accessed via API. Let $\mathcal{D} = \{d_1, \ldots, d_N\}$

[5]We assume a fixed, task-agnostic embedding model decoupled from the EXECUTIVE model for RAG, enabling the retrieval index to be reused across models. For ICL, prompts are assumed to be raw-text-based, free of model-specific formatting.

denote a target corpus of N documents containing knowledge that $\mathcal{M}_\theta$ cannot reliably recall[6]. Let $\mathcal{Q}$ be a set of queries, each $q \in \mathcal{Q}$ associated with a ground-truth answer $a^\star(q)$ and a set of supporting documents $\mathcal{S}(q) \subseteq \mathcal{D}$. Note that $\mathcal{S}(q)$ is a theoretical construct used to characterize query complexity.

Knowledge integration mechanism. A *knowledge integration mechanism* is a pair (Φ, f), where Φ maps the corpus to a representation $\mathcal{K} \doteq \Phi(\mathcal{D})$ and f combines $\mathcal{K}$ with $\mathcal{M}_\theta$ at inference to produce responses $f(\mathcal{M}_\theta, \mathcal{K}, q)$. We formalize the goal as follows.

Definition 1 (Knowledge Integration Problem)**.** *Given a frozen model $\mathcal{M}_\theta$ and a target corpus $\mathcal{D}$, find a mechanism (Φ, f) that maximizes $\mathbb{E}_{q\sim\mathcal{Q}}\left[\mathbb{P}\left\{f(\mathcal{M}_\theta, \Phi(\mathcal{D}), q) = a^\star(q)\right\}\right]$ without modifying θ.*

Existing approaches. Existing methods differ in their choice of (Φ, f). *ICL* sets $\mathcal{K} = \mathcal{D}$ and $f(\mathcal{M}_\theta, \mathcal{K}, q) = \mathcal{M}_\theta([\mathcal{D}; q])$, i.e., appending the corpus directly to the prompt. *RAG* constructs $\mathcal{K}$ as a retrieval index and defines f to retrieve a subset $\hat{\mathcal{S}} \subseteq \mathcal{D}$ before passing $[\hat{\mathcal{S}}; q]$ to $\mathcal{M}_\theta$. *Fine-tuning* sets $\mathcal{K} = \emptyset$ and $f = \mathcal{M}_{\theta'}$, where θ' is obtained by updating θ on $\mathcal{D}$. In contrast, MEMO defines $\mathcal{K}$ as the parameters of a small, dedicated MEMORY model $\mathcal{M}_\varphi$ with $\varphi \ll \theta$, trained on a reflection QA dataset derived from $\mathcal{D}$, and queried by a frozen EXECUTIVE model $\mathcal{M}_\theta$ at inference time. Tab. 3 summarizes how these paradigms compare across desirable properties.

C. MeMo: Memory as a Model

MEMO addresses the knowledge integration problem (Def. 1) through two components: a frozen *model* $\mathcal{M}_\theta$ (EXECUTIVE model), which handles reasoning and responds to user queries, and a MEMORY *model* $\mathcal{M}_\varphi$, which is trained to encode knowledge in its parameters from a target corpus $\mathcal{D}$. Our pipeline operates in two phases: (i) a *training phase* that constructs MEMORY model from $\mathcal{D}$, and (ii) an *inference phase* in which EXECUTIVE model queries and retrieves information from MEMORY model to answer knowledge-intensive questions (see Sections C.1, C.2 and C.4).

C.1. Data Synthesis Pipeline

Given a corpus of documents $\mathcal{D}$, our objective in the data generation process is to construct a reflection QA dataset $\mathcal{Q}_{\text{final}}$ that captures both single-document facts and cross-document relationships. This process is driven by a GENERATOR model $\mathcal{M}_{\text{gen}}$ and proceeds through five steps, as summarized in Alg. 1 and illustrated in Fig. 1: (1) fact extraction from raw documents, (2) consolidation of redundant or overlapping information, (3) verification and rewriting to ensure correctness and clarity, (4) entity surfacing to explicitly represent key entities, and (5) cross-document synthesis to integrate evidence across the corpus. Importantly, no document identifiers or watermarks are embedded in the generated QA pairs at any step, preventing MEMORY model from exploiting shortcut signals during evaluation.

Step 1: Fact extraction. Each document $d \in \mathcal{D}$ is segmented into chunks C, where each chunk corresponds either to an entire document or to a contiguous segment of a longer document. For each chunk, $\mathcal{M}_{\text{gen}}$ performs two parallel extraction processes: *direct extraction*, which captures explicitly stated facts (producing $\mathcal{Q}_{\text{dir}}$), and *indirect extraction*, which targets inferred or synthesized information beyond the surface text (producing $\mathcal{Q}_{\text{indir}}$). This dual extraction process ensures that both factual recall and inferential reasoning are represented in the training signal for MEMORY model.

Step 2: Consolidation. The GENERATOR model $\mathcal{M}_{\text{gen}}$ consolidates $\mathcal{Q}_{\text{dir}} \cup \mathcal{Q}_{\text{indir}}$ by identifying QA pairs that share a common underlying context (such as entity, time period, or relationship type) and combining them into QA pairs that encompass multiple facts, denoted $\mathcal{Q}_{\text{mrg}}$. This merging process produces training instances that require integrating multiple facts within the same contextual chunk, going beyond single-fact question answering pairs. The synthesized QA pairs are subsequently unified with the original sets to form the consolidated dataset $\mathcal{Q}_{\text{con}} = \mathcal{Q}_{\text{dir}} \cup \mathcal{Q}_{\text{indir}} \cup \mathcal{Q}_{\text{mrg}}$.

Step 3: Verification and rewriting. Each QA pair in $\mathcal{Q}_{\text{con}}$ is evaluated for *self-containment* by $\mathcal{M}_{\text{gen}}$, i.e., whether it can be fully understood and correctly answered in isolation, without access to the source chunk. Common failure modes include unresolved pronouns (e.g., "What did *they* propose?") and implicit references (e.g., "As noted in the above table…"). Non-self-contained QA pairs are rewritten by $\mathcal{M}_{\text{gen}}$ using the source chunk C as context; QA pairs that remain ambiguous after rewriting are discarded. This check-and-rewrite procedure yields the verified set $\mathcal{Q}_{\text{ver}}$, a set of QA pairs that can be used as training examples without access to the source chunk.

[6]We do not assume $\mathcal{D}$ is disjoint from $\mathcal{D}_{\text{pre}}$, as training data is rarely disclosed by model providers. A document is considered *effectively absent* from $\mathcal{M}_\theta$'s knowledge if the model fails to answer questions grounded in it, either because it never appeared in $\mathcal{D}_{\text{pre}}$ or because the training process was insufficient to retain it. For more information, refer to App. J.

Algorithm 1 Reflection QA Dataset Generation Pipeline from Target Corpus

Require: Corpus $\mathcal{D}$, generator $\mathcal{M}_{\text{gen}}$, document groups $\mathcal{G} = \{G_1, \ldots, G_R\}$ with $G_i \subseteq \mathcal{D}$
$\mathcal{Q}_{\text{final}} \leftarrow \emptyset$
for all document $d \in \mathcal{D}$ **do**
  $C \leftarrow \text{Chunk}(d)$ ▷ Segment into chunks
  $\mathcal{Q}^d_{\text{ver}} \leftarrow \emptyset$
  for all chunk $c \in C$ **do**
    $\mathcal{Q}_{\text{dir}},\ \mathcal{Q}_{\text{indir}} \leftarrow \mathcal{M}_{\text{gen}}(c)$ ▷ Step 1: Direct and indirect extraction
    $\mathcal{Q}_{\text{raw}} \leftarrow \mathcal{Q}_{\text{dir}} \cup \mathcal{Q}_{\text{indir}}$ ▷ Step 2a: Merge direct and indirect
    $\mathcal{Q}_{\text{mrg}} \leftarrow \mathcal{M}_{\text{gen}}(\mathcal{Q}_{\text{raw}})$ ▷ Step 2b: Consolidate related pairs
    $\mathcal{Q}_{\text{con}} \leftarrow \mathcal{Q}_{\text{raw}} \cup \mathcal{Q}_{\text{mrg}}$ ▷ Step 2c: Full merge set
    $\mathcal{Q}_{\text{ver}} \leftarrow \mathcal{M}_{\text{gen}}(\mathcal{Q}_{\text{con}},\ c)$ ▷ Step 3: Verify self-containment; rewrite or discard
    $\mathcal{Q}^d_{\text{ver}} \leftarrow \mathcal{Q}^d_{\text{ver}} \cup \mathcal{Q}_{\text{ver}}$
  end for
  $\mathcal{Q}^d_{\text{ent}} \leftarrow \mathcal{M}_{\text{gen}}(\mathcal{Q}^d_{\text{ver}})$ ▷ Step 4: Entity-surfacing pairs
  $\mathcal{Q}_{\text{final}} \leftarrow \mathcal{Q}_{\text{final}} \cup \mathcal{Q}^d_{\text{ver}} \cup \mathcal{Q}^d_{\text{ent}}$
end for
for all $G_i \in \mathcal{G}$ **do**
  $\mathcal{Q}_{\text{cross}} \leftarrow \mathcal{M}_{\text{gen}}\Big(\bigcup_{d \in G_i} \big(\mathcal{Q}^d_{\text{ver}} \cup \mathcal{Q}^d_{\text{ent}}\big)\Big)$ ▷ Step 5: Cross-document synthesis
  $\mathcal{Q}_{\text{final}} \leftarrow \mathcal{Q}_{\text{final}} \cup \mathcal{Q}_{\text{cross}}$
end for
return $\mathcal{Q}_{\text{final}}$

Step 4: Entity surfacing. For each named entity in $\mathcal{Q}_{\text{ver}}$, $\mathcal{M}_{\text{gen}}$ generates a set of entity-surfacing QA pairs in which the question encodes the entity's attributes and relationships (including connections to other named entities) and the answer reveals its identity. Facts about each entity are aggregated across all QA pairs within the chunk prior to generation, enabling the integration and composition of information from multiple source pairs. Questions are generated at varying levels of complexity, ranging from single-fact to multi-fact queries. These pairs, denoted $\mathcal{Q}_{\text{ent}}$, aim to mitigate the reversal curse (Berglund et al., 2023; Allen-Zhu & Li, 2023) by training MEMORY model to infer entities from indirect or partially specified descriptions. This capability supports the *entity identification turn* at inference time (Sec. C.4).

Step 5: Cross-document synthesis. The final step operates over pre-defined document groups $\mathcal{G} = \{G_1, \ldots, G_R\}$, where chunks within each group G_i are topically related. Such groups arise naturally, for example, when a large document is segmented into chunks (forming a single group) or from human-provided labels. For each group G_i, $\mathcal{M}_{\text{gen}}$ is provided with both the verified pairs $\mathcal{Q}^d_{\text{ver}}$ and the entity-surfacing pairs $\mathcal{Q}^d_{\text{ent}}$ for all $d \in G_i$ from all member documents and identifies two types of cross-document connections:

- *Converging clues*: multiple documents provide complementary facts about the same entity, which together enable its identification.
- *Parallel properties*: different entities across documents share a common attribute or role, enabling comparative and analogical reasoning.

Both types yield QA pairs with support size $\mathcal{S}(q) > 1$ (Sec. B), directly targeting the cross-document synthesis objective. The final dataset is $\mathcal{Q}_{\text{final}} = \mathcal{Q}_{\text{ver}} \cup \mathcal{Q}_{\text{ent}} \cup \mathcal{Q}_{\text{cross}}$, which collectively captures self-contained, entity-centric, and cross-document reflections for training MEMORY model. Ablations of the pipeline design are presented in App. F.

C.2. Training the MEMORY Model

Given $\mathcal{Q}_{\text{final}}$, MEMORY model is trained via SFT to map questions directly to answers *without* access to source documents at inference time. MEMORY model is initialized from a small pretrained language model, smaller than EXECUTIVE model (e.g., 1.5B vs. 32B parameters), and optimized by minimizing the next-token prediction loss over answer tokens only.

$$\mathcal{L}(\varphi) \;=\; -\sum_{(q_i,\, a_i)\, \in\, \mathcal{Q}_{\text{final}}} \;\sum_{t=1}^{|a_i|} \log \mathcal{M}_{\varphi}\Big(a_i^{(t)} \;\Big|\; q_i,\, a_i^{(1:t-1)}\Big)\,.$$

Conditioning only on the question and preceding answer tokens, and never on source documents, forces MEMORY model to internalize knowledge *parametrically* rather than rely on copying from retrieved context. This constitutes a key distinction from RAG-based readers: at inference time, MEMORY model generates answers solely from its internalized parametric knowledge, without access to any external corpus. Further details on hyperparameter choices are provided in App. G and training paradigms (full SFT vs. LoRA) are provided in App. P.

C.3. Continual Knowledge Integration via Model Merging

A practical desideratum of any knowledge integration system is the ability to incorporate new corpora incrementally without retraining on or rebuilding from all previously ingested sources. For parametric models, integrating new knowledge typically requires retraining on the union of all observed corpora, a cost that grows prohibitively with the number of sources. In contrast, non-parametric systems such as knowledge graphs and vector databases support efficient incremental updates. We explore *model merging* (Yang et al., 2024) as an approach to close this gap for parametric models. Model merging aims to preserve knowledge from multiple sources without requiring joint training on their union, by combining K MEMORY model models, each trained independently on a distinct corpus, into a single model.

Continual knowledge integration. Let $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$ be a collection of pairwise disjoint target corpora. For each corpus $\mathcal{D}_i$, we generate a reflection QA dataset $\mathcal{Q}_{\text{final}}^{(i)}$ (Sec. C.1) and train a corresponding MEMORY model $\mathcal{M}_{\varphi_i}$ via SFT (Sec. C.2), initializing all K models from the same pretrained base $\mathcal{M}_{\varphi_0}$. We define the *task vector* for $\mathcal{D}_i$ as $\tau_i = \varphi_i - \varphi_0$, capturing the parametric shift induced by training on $\mathcal{D}_i$ alone. The merged MEMORY model is then obtained as

$$\varphi_{\text{merged}} = \text{Merge}(\varphi_0,\ \{\tau_i\}_{i=1}^{K};\ \Theta),$$

where Θ denotes method-specific hyperparameters (e.g., merging coefficients, sparsification densities). We discuss alternative merging methods and their respective limitations in App. I.

C.4. Inference-Time Integration

At inference time, EXECUTIVE model queries and retrieves information from MEMORY model through a structured multi-turn protocol, with EXECUTIVE model treating MEMORY model as an external knowledge oracle. The pipeline has three sequential stages, each designed to progressively improve the likelihood of producing a correct final answer, as illustrated in Fig. 1 (right). Each stage utilizes distinct prompts, sampling temperatures and independent budgets to control the number of interactions between EXECUTIVE model and MEMORY model.

Stage 1: Grounding. Given a query q, EXECUTIVE model decomposes it into a set of atomic, clue-probing sub-questions $\{q'_1, \ldots, q'_J\}$, where each sub-question targets a single identifying constraint in q, and J is adaptively determined by EXECUTIVE model. The MEMORY model answers each sub-question independently, without shared context, producing grounding responses $\{m_1, \ldots, m_J\}$. These responses draw on MEMORY model's parametric knowledge to provide additional contextual grounding for subsequent interactions in the later stages.

Stage 2: Entity identification. Using the grounding responses as context, EXECUTIVE model iteratively narrows a set of candidate entities by issuing targeted follow-up sub-queries to MEMORY model across multiple interactions. This process continues until EXECUTIVE model converges on a single entity $e^\star$ or the stage budget is exhausted. If no candidates are identified, Stage 3 is skipped and EXECUTIVE model synthesizes a final answer from the grounding responses alone. This stage leverages MEMORY model's training on the entity-surfacing QA pairs $\mathcal{Q}_{\text{ent}}$ (Sec. C.1).

Stage 3: Answer seeking and synthesis. Conditioned on the identified entity $e^\star$, EXECUTIVE model queries MEMORY model for additional supporting facts through targeted follow-up questions. Once sufficient evidence is gathered, or the stage budget is exhausted, EXECUTIVE model synthesizes the accumulated responses into a final answer:

$$\hat{a} \;=\; \mathcal{M}_\theta\big(q,\ \{m_j\}_{j=1}^{J},\ e^\star,\ m_{\text{seek}}\big).$$

Notably, the MEMORY model responses m_j and m_{seek} are compact natural-language snippets whose lengths are independent of the corpus size, ensuring constant-time inference. As all interactions with $\mathcal{M}_\theta$ occur through its input–output interface, MEMO remains fully compatible with black-box EXECUTIVE models, including proprietary APIs, without requiring access to internal parameters. For full implementation details, refer to App. K and the supplementary materials.

D. Leftover Experiments

D.1. Ablation on MEMORY model size

We investigate how the size of MEMORY model affects downstream task performance by comparing models of 1.5B and 14B parameters in the Qwen2.5 family. Implementation details are provided in App. N. Results in Tab. 4 show a consistent positive scaling trend: larger MEMORY models yield improved performance across all benchmarks and EXECUTIVE models. However, the results also show that a stronger EXECUTIVE model reasoning capability modulates this gap non-uniformly across tasks: the performance difference between MEMORY model sizes widens for NarrativeQA but shrinks for BrowseComp-Plus and MuSiQue. This suggests that the interaction between EXECUTIVE model reasoning capability and MEMORY model size is task-dependent.

Table 4. Ablation on MEMORY model size within the Qwen2.5 family. Bold results indicate best performing results in the column.

	BrowseComp-Plus		NarrativeQA		MuSiQue	
MEMORY Model	**Qwen2.5-32B**	**Gemini-3-Flash**	**Qwen2.5-32B**	**Gemini-3-Flash**	**Qwen2.5-32B**	**Gemini-3-Flash**
Qwen2.5-1.5B-Instruct	44.11 ± 2.22	61.00	24.00 ± 0.20	47.44	42.90 ± 1.39	59.70
Qwen2.5-14B-Instruct	$\mathbf{54.22 \pm 0.84}$	**66.67**	$\mathbf{26.85 \pm 0.39}$	**53.58**	$\mathbf{48.30 \pm 1.25}$	**60.20**

D.2. Ablation on MEMORY model family

We investigate whether the choice of MEMORY model family affects performance by comparing three models of similar parameter scale ($\sim$1–2B) but distinct architectures and pretraining lineages: Qwen2.5-1.5B-Instruct (Yang et al., 2025), Gemma3-1B-IT (Team et al., 2025), and LFM2.5-1.2B-Instruct (Amini et al., 2025). Implementation details are provided in App. O. Results in Tab. 5 show that MEMO performance is largely robust to the choice of MEMORY model architecture, demonstrating that the framework is not sensitive to the specific pretraining lineage of MEMORY model, and that the parametric knowledge compression induced by our training procedure generalizes across diverse model families.

Table 5. Ablation across MEMORY models at similar parameter scales ($\sim$1–2B). Bold results indicate best performing results in the column.

	BrowseComp-Plus		NarrativeQA		MuSiQue	
MEMORY Model	**Qwen2.5-32B-I**	**Gemini-3-F**	**Qwen2.5-32B-I**	**Gemini-3-F**	**Qwen2.5-32B-I**	**Gemini-3-F**
Qwen2.5-1.5B-Instruct	$\mathbf{44.11 \pm 2.22}$	**61.00**	$\mathbf{24.00 \pm 0.20}$	47.44	42.90 ± 1.39	**59.70**
Gemma3-1B-IT	41.67 ± 2.03	59.00	22.30 ± 2.47	**48.81**	41.17 ± 1.20	56.20
LFM2.5-1.2B-Instruct	37.33 ± 1.86	59.67	21.96 ± 1.97	46.42	$\mathbf{45.23 \pm 2.49}$	58.30

D.3. Continual integration via model merging

We test the streaming-update scenario described in Sec. C.2 on NarrativeQA, comparing model merging against full retraining of MEMORY model on the union of both subsets when the second arrives. Of the 14 sweep configurations (see Tab. 12, App. I), we report TIES (Yadav et al., 2023) at ρ=0.3 in the main paper, the top-performing one. Letting X and Y denote the SFT cost on each subset alone (cost scales approximately linearly with the number of QA pairs, so the union costs $X+Y$), cumulative compute across the two arrivals is $X+Y$ for merging versus $X+(X+Y)$ for full retraining.

Table 6. **Model merging vs. full retraining on NarrativeQA.** MEMORY model = Qwen2.5-14B-Instruct. Merge-TIES (ρ=0.3) is the best of 14 configurations swept (Tab. 12, App. I). Cumulative compute is reported in 8×H100 GPU-hours for K=2 subsets of $\sim$640k reflection QA pairs each. Δ denotes accuracy difference (%) relative to full retraining.

Method	**Cumulative compute**	**Qwen2.5-32B-I**		**Gemini-3-F**	
	(8×H100 GPU-h)	Acc. (%)	Δ	Acc. (%)	Δ
Full retrain ($X+(X+Y)$)	$\approx$ 72h	$\mathbf{26.85 \pm 0.39}$	—	**53.58**	—
Merge-TIES (ρ=0.3, $X+Y$)	$\approx$ 48h	15.81 ± 0.39	$\downarrow$ 11.04	34.47	$\downarrow$ 19.11

Merging cuts compute by 33% at K=2, with widening returns at scale. As reported in Tab. 6, the full-retrain baseline

incurs $X+(X+Y) \approx 72$ GPU-hours of cumulative compute, while merging accumulates only $X+Y \approx 48$ GPU-hours — a 33% reduction (Fig. 2). The gap widens with K: under the same per-corpus cost, merging scales as $\Theta(K)$ while full retraining scales as $\Theta(K^2)$, yielding a $5.5\times$ saving at $K=10$ (240 vs. 1,320 GPU-hours).

Merging trades a measurable accuracy gap for the compute saving, but still beats retrieval. Merge-TIES (ρ=0.3) trails the full-retrain MEMORY model by 11.0% under Qwen2.5-32B-Instruct and 19.1% under Gemini-3-Flash (Tab. 6); across the 14-configuration sweep, accuracy ranges from 7.85% (SLERP, worst) to 15.81% (TIES, best), shown in Fig. 2. Despite this gap, the merged MEMORY model still outperforms every retrieval baseline (BM25, NV-Embed-V2, HippoRAG2, Cartridges; see Tab. 1) on NarrativeQA, indicating that even an aggressively-cheaper method preserves most of MEMO's qualitative advantage over retrieval-based approaches. TIES and DARE-Linear at ρ=0.3 dominate the sweep, suggesting that aggressive sparsification combined with sign-conflict resolution is the most reliable merging recipe in this regime.

E. Preparation of datasets

Corpus construction. Extending from our description in Sec. 2, we distinguish between two types of documents[7]: evidence documents, which contain information relevant to answering a given question, and negative documents, which are irrelevant and serve as noise. For BrowseComp-Plus, we used 1,775 unique evidence documents and 1,766 unique negative documents (after removal of non-English documents), yielding 3,541 documents. For MuSiQue, we used 2,648 documents for each of the evidence and negative documents, yielding 5,296 documents. NarrativeQA does not have negative documents.

Chunking strategy. As shown in Tab. 7, NarrativeQA full documents span the 32,769–131,072 token range with a median length of 65,925 tokens, reflecting the long-form nature of the source novels. Processing such documents without chunking risks reduced coverage of extractable QA pairs in Step 1 of Alg. 1, as attention quality is known to deteriorate over longer contexts (Hsieh et al., 2024). We therefore chunk NarrativeQA documents using a fixed sliding window of 6,400 words with a 640-word overlap (10% overlap ratio), yielding 75 chunks concentrated in the 4,097–16,384 token range and accounting for 96% of all chunks, with a median group size of 7 per document as shown in Tab. 8. Unlike NarrativeQA, MuSiQue documents are compact with 99.70% falling below 512 tokens, and each MuSiQue document is treated as a single chunk.

Table 7. Token length distribution across corpora at the *chunk level*, where n represents the total number of individual chunks processed by Alg. 1. Each entry reflects the token count of a single text chunk. Statistics for NarrativeQA are reported before and after chunking.

Token Range	**BrowseComp-Plus** (n = 3,541)	**NarrativeQA Full Docs** (n = 10)	**NarrativeQA Chunks** (n = 75)	**MuSiQue** (n = 5,296)
0–512	606 (17.11%)	0 (0.00%)	0 (0.00%)	5,280 (99.70%)
513–1,024	591 (16.69%)	0 (0.00%)	0 (0.00%)	16 (0.30%)
1,025–2,048	746 (21.07%)	0 (0.00%)	1 (1.33%)	0 (0.00%)
2,049–4,096	598 (16.89%)	0 (0.00%)	2 (2.67%)	0 (0.00%)
4,097–8,192	428 (12.09%)	0 (0.00%)	36 (48.00%)	0 (0.00%)
8,193–16,384	323 (9.12%)	0 (0.00%)	36 (48.00%)	0 (0.00%)
16,385–32,768	145 (4.09%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
32,769–65,536	56 (1.58%)	5 (50.00%)	0 (0.00%)	0 (0.00%)
65,537–131,072	20 (0.56%)	5 (50.00%)	0 (0.00%)	0 (0.00%)
$>$ 131,072	28 (0.79%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Min tokens	14	32,804	1,943	23
Median tokens	1,756	65,925	8,158	105
Mean tokens	7,192	66,324	8,713	123
p_{95} **tokens**	20,330	119,267	11,266	270
Max tokens	1,235,897	119,267	12,104	828

[7]Note that for BrowseComp-Plus and MuSiQue, the gold documents are a subset of the evidence documents.

Table 8. Distribution of document group sizes across datasets, where group size denotes the number of chunks associated with a single question or document. For BrowseComp-Plus and MuSiQue, each question is associated with a subset of chunks drawn from the corpus, and group size represents the number of chunks per *question*. For NarrativeQA, each subset of chunks is derived from the original document used for multiple questions, and group size represents the number of chunks per *document*.

Document Group Size Range	**BrowseComp-Plus** ($n_{\text{group}} = 300$)	**NarrativeQA Chunks** ($n_{\text{group}} = 10$)	**MuSiQue** ($n_{\text{group}} = 1{,}000$)
0–2	2 (0.67%)	0 (0.00%)	0 (0.00%)
3–4	14 (4.67%)	3 (30.00%)	518 (51.80%)
5–8	78 (26.00%)	4 (40.00%)	482 (48.20%)
9–16	159 (53.00%)	3 (30.00%)	0 (0.00%)
>16	47 (15.67%)	0 (0.00%)	0 (0.00%)
Min group size	2	3	4
Median group size	12	7	4
Mean group size	11.8	7.5	5.3
p_{95} **group size**	20	16	8
Max group size	23	16	8

BrowseComp-Plus documents are also treated as a single chunk. The time complexity of Step 5 in Alg. 1 is $O(k \cdot C^2 \cdot Q^2)$, where $k = n_{\text{group}}$ is the number of groups, $C = |G_i|$ is the number of participating chunks per group, and $Q = \bar{Q}_i$ is the average number of QA pairs extracted per chunk. Since chunking increases C, pipeline costs at Step 5 scale quadratically as the number of chunks per group increases. Given that only 2.93% of BrowseComp-Plus documents exceed $32{,}768$ tokens, the majority of documents fit within a single chunk, making the cost of chunking difficult to justify. We therefore opted against chunking in favor of lower pipeline cost, and leave a systematic evaluation of chunking strategies and related tradeoffs to future work.

Subset selection of negative documents. We include only a subset of negative documents for BrowseComp-Plus and MuSiQue due to computational constraints arising from the quadratic scaling of Step 5. As reported in Tab. 8, BrowseComp-Plus currently has a mean group size of 11.8 and a maximum of 23, while MuSiQue has a mean group size of 5.3 and a maximum of 8. Incorporating all available negative documents, which average 78 per question (up to 197) for BrowseComp-Plus and 17 per question (up to 18) for MuSiQue, would cause the group size to increase substantially. Given the quadratic dependence on C in Step 5, this would result in a prohibitive increase in pipeline cost for BrowseComp-Plus ($k = 300$) and MuSiQue ($k = 1{,}000$). Hence, we only include up to $N_{\text{evidence}}^{\text{dataset}}$ negative documents for each question in the corpus.

F. Discussion on steps in data generation pipeline

F.1. Ablation of data synthesis steps

We experiment with the data generation pipeline to show the importance of each step. We perform a leave-one-out (LOO) ablation for each step of data synthesis and train the model on the synthesized QA pairs generated. Results are reported in Tab. 9 on the NarrativeQA and MuSiQue datasets using Qwen2.5-32B-Instruct as the EXECUTIVE model and Qwen2.5-1.5B-Instruct as the MEMORY model.

Step 5 (Cross-document synthesis) is the most critical component of the pipeline. Its removal causes accuracy to collapse to 6.37% and 24.17% on NarrativeQA and MuSiQue respectively, against baseline scores of 24.00% and 42.90%, accompanied by a near-total loss of training data ($0.002\times$ and $0.195\times$ retention). As described in Sec. C.1, Step 5 enables cross-document synthesis where $\mathcal{M}_{\text{gen}}$ constructs $\mathcal{Q}_{\text{cross}}$ pairs spanning inter-document connections and cross-chunk connections within a single long document, making it the dominant source of training pairs in $\mathcal{Q}_{\text{final}}$ and directly targeting the multi-source synthesis objective central to both benchmarks.

An interesting anomaly arises with Step 2 and Step 3, where their removal does not consistently hurt performance and improves accuracy on NarrativeQA. Step 2 merges related QA pairs from a single document chunk into multi-fact questions by identifying commonalities such as shared entities, overlapping time periods, and sequential events. For MuSiQue,

Table 9. LOO ablation accuracy at best performing Qwen2.5-1.5B-Instruct epoch across datasets. Data ratio indicates the number of QA pairs retained relative to the baseline. For each step removed, the Qwen2.5-1.5B-Instruct was retrained, and we report the mean ± std. dev. over 3 runs at the same training epoch as the baseline.

	NarrativeQA		MuSiQue	
Ablation	**Data Ratio**	**Accuracy (%)**	**Data Ratio**	**Accuracy (%)**
Baseline (all steps)	$1.000\times$	24.00 ± 0.20	$1.000\times$	42.90 ± 1.39
Step 1a removed	$0.434\times$	20.48 ± 0.90	$0.381\times$	30.00 ± 0.17
Step 1b removed	$0.598\times$	22.98 ± 1.04	$0.651\times$	37.33 ± 0.25
Step 2 removed	$0.739\times$	24.69 ± 1.10	$0.621\times$	37.10 ± 1.76
Step 3 removed	$2.078\times$	28.90 ± 0.86	$1.128\times$	41.70 ± 0.78
Step 4 removed	$0.378\times$	23.21 ± 1.56	$0.501\times$	39.10 ± 0.02
Step 5 removed	$0.002\times$	6.37 ± 0.39	$0.195\times$	24.17 ± 0.25

these commonalities reflect genuine knowledge relationships that directly resemble the multi-hop factual reasoning the benchmark evaluates, such that removing Step 2 eliminates a large fraction of useful training pairs, leading to a drop in accuracy from 42.90% to 37.10%. For NarrativeQA, however, the same consolidation patterns operate on superficial narrative co-occurrences rather than meaningful knowledge relationships. The predominant commonality categories are event or scene groupings that NarrativeQA does not evaluate, and entity co-occurrence patterns that are trivially satisfied given the pervasive presence of central characters across scenes. Removing Step 2 eliminates these low-quality pairs, leading to the marginal accuracy improvement from 24.00% to 24.69%.

Removing Step 3 retains more data than the baseline ($2.078\times$ and $1.100\times$ for NarrativeQA and MuSiQue respectively), yet the effect on performance diverges. For MuSiQue, performance drops from 42.90% to 41.78%, whereas for NarrativeQA, performance improves from 24.00% to 28.90%. Step 3 applies a self-containment filter that rewrites or discards pairs whose questions cannot be understood without access to the source chunk. For MuSiQue, violations are predominantly localized and shallow, making them amenable to filtering; the proposed filter effectively identifies and removes defective pairs. For NarrativeQA, long-form narrative text frequently contains pronouns and temporal references that span many paragraphs, which are structural features of the domain rather than fixable defects. This causes the rewriting loop to introduce substitute unrelated content and corrupt the pairs produced by earlier steps. Removing Step 3 for NarrativeQA therefore avoids this domain-induced corruption and retains the original pairs intact, explaining both the data retention ratio increase and the accuracy improvement. This suggests that Step 3 is most beneficial when applied to domains where self-containment violations are well-defined and resolvable.

The remaining steps follow a consistent trend: removing Step 1a, Step 1b, or Step 4 reduces both data volume and accuracy across both datasets, confirming that each step contributes a distinct and meaningful role to the final training corpus quality.

F.2. Additional steps considered but excluded

Three additional steps were considered but ultimately excluded from the pipeline. These include paraphrasing (Ovadia et al., 2024), increasing the number of sampling trials at Step 1 of Alg. 1, and a targeted fill whereby $\mathcal{M}_{\text{gen}}$ reviews the generated QA pairs and rewrites them to incorporate additional missed information. Paraphrasing was excluded as the scale of generated pairs already provides sufficient coverage (≈600k–1.6M across the three datasets, see Table 11), and the potential gains were outweighed by the additional computational overhead. Increasing sampling trials proved unreliable, as additional trials did not consistently extract facts that the initial pass had failed to extract. The targeted fill similarly offered limited gains, where appending the existing QA pairs as context to prompt a revision only lengthens the context when the model had already failed to extract a fact from the original chunk, likely exacerbating attention further degradation over long inputs (Hsieh et al., 2024) and making retrieval of relevant information less reliable at inference time.

G. MEMORY model hyperparameter settings

Training was conducted on H100 and H200 GPUs using the hyperparameter settings reported in Tab. 10. The effective batch size for each dataset is summarized in Tab. 11.

Table 10. MEMORY model SFT Training Configuration

Parameter	Value
Optimizer	Fused AdamW
Gradient checkpointing	True
Learning rate (LR)	2×10^{-5}
Num of Training epochs	3
LR scheduler type	Constant with warmup
Warmup ratio	0.05
Weight decay	0.01
Max gradient norm	1.0
Max sequence length	8096
Precision	BF16
Attention implementation	Flash Attention 2

Table 11. Effective batch sizes and number of QA pairs used. NarrativeQA.1 and NarrativeQA.2 are independent subsets partitioned from the original that were used for model merging.

Dataset	Target Num of Questions	Num of QA Pairs	Effective Batch Size
BrowseComp-Plus	300	1,639,995	512
NarrativeQA	293	1,276,676	512
NarrativeQA.1	146	635,009	256
NarrativeQA.2	147	641,667	256
MuSiQue	1,000	664,762	256

H. Compute resources

All experiments were conducted using NVIDIA H200 GPUs. We report computational cost in GPU-hours.

Data generation. Generating the full reflection dataset for BrowseComp-Plus, NarrativeQA, and MuSiQue took approximately 240, 200, and 150 GPU-hours respectively.

Training. MEMORY model (Qwen2.5-14B-Instruct) training for a single run BrowseComp-Plus, NarrativeQA, and MuSiQue took approximately 180, 150, 90 GPU-hours.

I. Model training discussion

We considered three training paradigms: CPT, SFT, and LoRA-based SFT. CPT was excluded as it risks degrading instruction-following capability (Wu et al., 2024), which is critical for downstream QA evaluation. Full SFT was selected as it directly optimizes for the target task while preserving alignment (Ouyang et al., 2022). LoRA-based SFT serves as a parameter-efficient alternative and we include a comparison to these training methods in App. P.

Model merging targets the practical streaming setting in which new corpora arrive over time and MEMORY model must continually integrate them. Retraining MEMORY model from scratch on the union of all observed corpora is the natural baseline but quickly becomes prohibitive at scale, since its cost grows with the cumulative corpus size. Model merging instead trains a separate MEMORY model on each new corpus and combines it with the existing model in parameter space, so the cost of each update scales only with the size of the new corpus rather than the entire history. This decoupling comes at a measurable accuracy cost relative to full retraining, which we quantify in Fig. 2. We assume the corpora to be merged are pairwise disjoint.

I.1. Model merging

Merging methods. We consider the following methods, all of which produce φ_{merged} without ever training on $\mathcal{D}_1 \cup \cdots \cup \mathcal{D}_K$:

- **Linear merging** (Wortsman et al., 2022) computes a weighted sum of task vectors: $\varphi_{\text{merged}} = \varphi_0 + \sum_{i=1}^{K} \lambda_i \tau_i$, where $\lambda_i > 0$ are merging coefficients.
- **SLERP** (Shoemake, 1985) interpolates between two task vectors along the unit sphere, preserving their magnitudes: $\varphi_{\text{merged}} = \varphi_0 + \text{SLERP}(\tau_1, \tau_2;\, t)$, with $t \in [0, 1]$ controlling the interpolation factor.
- **Task arithmetic** (Ilharco et al., 2023) adds task vectors directly without further processing, recovering linear merging as a special case with uniform λ_i.
- **TIES** (Yadav et al., 2023) resolves interference among task vectors before summation by (i) *trimming* each τ_i to its top-ρ fraction of largest-magnitude entries, (ii) *electing* a sign at each coordinate by magnitude-weighted majority vote, and (iii) *disjoint-merging* only the entries that agree with the elected sign.
- **DARE** (Yu et al., 2024) sparsifies each task vector by randomly dropping a fraction $1 - \rho$ of its entries and rescaling the survivors by $1/\rho$ to preserve expected magnitude, before linear merging.
- **DARE-TIES** (Yu et al., 2024) combines DARE-style stochastic sparsification with TIES sign-conflict resolution, retaining the diversity of random dropout while filtering out conflicting updates.

Avoiding catastrophic forgetting. Because no individual MEMORY model $\mathcal{M}_{\varphi_i}$ is ever fine-tuned on another corpus' data, model merging cannot induce the kind of distributional interference that drives catastrophic forgetting in sequential fine-tuning (Luo et al., 2025). Knowledge from each corpus is preserved within its own task vector τ_i, and conflicts between task vectors are addressed at merge time via the methods above rather than during gradient updates.

Scalability. When a new corpus $\mathcal{D}_{K+1}$ arrives, we train auxiliary model $\mathcal{M}_{\varphi_{K+1}}$ on its reflection QA dataset, derive τ_{K+1}, and re-merge in $\mathcal{O}(1)$ additional cost relative to the full collection. This enables modular, plug-and-play integration over a continuous stream of disjoint knowledge sources, unlike retraining from scratch on $\bigcup_i \mathcal{D}_i$, which scales linearly with the cumulative corpus size.

Inference. The merged MEMORY model is queried identically to a single-corpus MEMORY model via the structured multi-turn protocol described in Sec. C.4. Because merging operates entirely in parameter space and produces a model with the same architecture and interface as $\mathcal{M}_{\varphi_0}$, it inherits the plug-and-play property of MEMO without requiring changes to the EXECUTIVE model or the inference protocol. Importantly, the EXECUTIVE model queries a *single* merged MEMORY model at inference rather than dispatching across K separate per-corpus MEMORY models, keeping the multi-turn retrieval pipeline unchanged regardless of how many corpora have been integrated.

Procedure. For our experiments we partition NarrativeQA into two pairwise-disjoint subsets, NarrativeQA.1 and NarrativeQA.2, of ∼640k reflection QA pairs each. Each subset is used to fine-tune an independent MEMORY model from the same Qwen2.5-14B-Instruct base via SFT for 3 epochs, producing $\mathcal{M}_{\varphi_1}$ and $\mathcal{M}_{\varphi_2}$ at SFT costs of X and Y GPU-hours, respectively (each is ≈ 24 GPU-hours on 8×H100; full-retrain on the union NarrativeQA.1 ∪ NarrativeQA.2 costs $X{+}Y \approx 48$ GPU-hours by linear scaling). We evaluate every saved checkpoint of each run on the held-out NarrativeQA evaluation set and select the best-performing checkpoint per subset; the corresponding task vectors τ_1 and τ_2 are the inputs to the merging step. We then sweep all six merging methods listed above (Linear, Task arithmetic, SLERP, TIES, DARE, DARE-TIES) at three sparsification densities $\rho \in \{0.3, 0.5, 0.7\}$ (or three interpolation factors $t \in \{0.3, 0.5, 0.7\}$ for SLERP), giving 14 merged-MEMORY model configurations in total. Each configuration is evaluated on NarrativeQA with Qwen2.5-32B-Instruct as EXECUTIVE model (mean ± std over 3 runs). The configuration that we report in the Sec. 2 as *Merge-TIES* is the best of the sweep (TIES with ρ=0.3).

Results. A single SFT run consumes ≈ 24 GPU-hours on 8×H100; after two arrivals, full retraining incurs $X{+}(X{+}Y){=}72$ GPU-hours of cumulative compute, whereas merging accumulates only $X{+}Y{=}48$ GPU-hours — a 33% reduction (Fig. 2). The asymptotic gap widens with K: under the same per-corpus cost, merging scales as $\Theta(K)$ while full retraining scales as $\Theta(K^2)$, yielding a $5.5\times$ saving at $K{=}10$ (240 vs. 1,320 GPU-hours). On accuracy, Merge-TIES (ρ=0.3) trails full retraining by 11.0% with Qwen2.5-32B-Instruct as EXECUTIVE model (15.81% vs. 26.85%) and by 19.1% with Gemini-3-Flash (34.47% vs. 53.58%), placing the merged MEMORY model below the union-retrained MEMORY model but above every retrieval baseline. The full per-method sweep is reported in Tab. 12: TIES (ρ=0.3) and DARE-Linear (ρ=0.3) lead at 15.81% and 15.47% respectively, while SLERP (t=0.5) is the worst configuration at 7.85%. The pattern across families

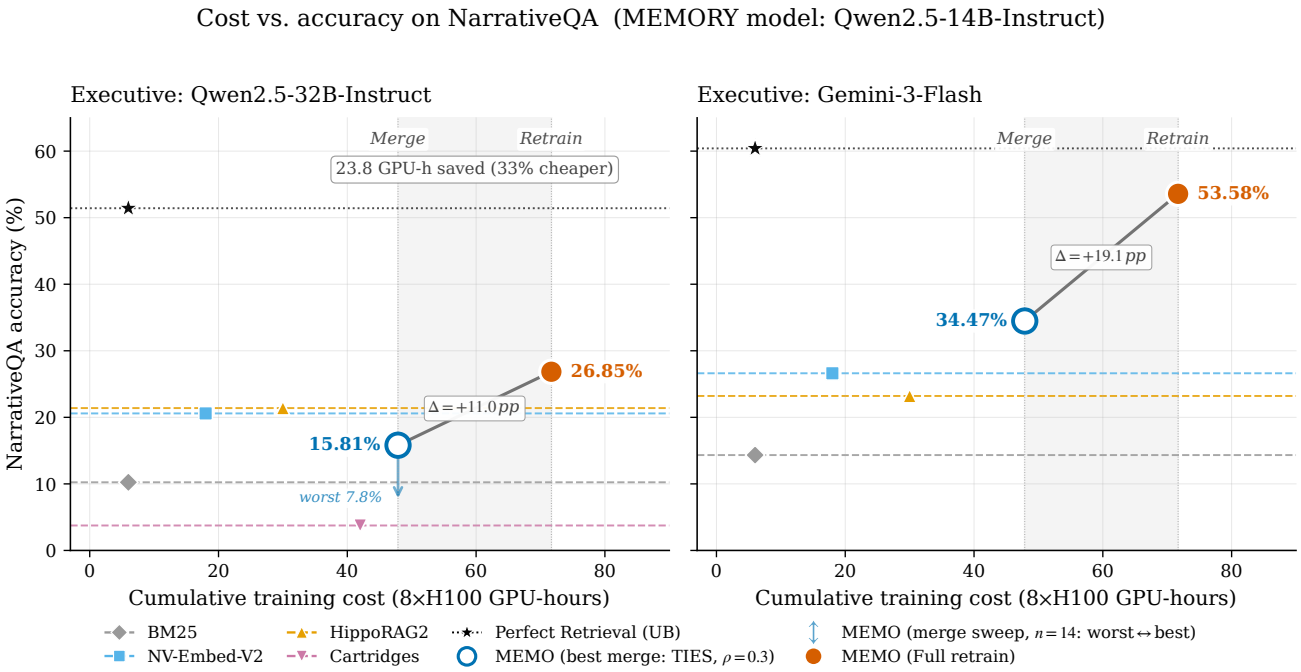

Figure 2. **Cost–accuracy trade-off on NarrativeQA when a second corpus arrives (K=2, MEMORY model = Qwen2.5-14B-Instruct, 8×H100).** Cumulative training cost is shown on the x-axis (one Qwen-14B SFT run takes $\approx$ 24 GPU-hours on a 640k-QA-pair corpus). Merging trains MEMORY model only on the new corpus, costing $X+Y \approx 48$ GPU-hours, while full retraining re-runs on the union, costing $X+(X+Y) \approx 72$ GPU-hours — a 33% saving. Merge-TIES (ρ=0.3) trails full retraining by 11.0% with Qwen2.5-32B-Instruct and 19.1% with Gemini-3-Flash as EXECUTIVE model, but still outperforms all retrieval baselines (BM25, NV-Embed-V2, HippoRAG2, Cartridges). The vertical ↕ at the merge cost shows the worst-to-best range across the 14 merge configurations swept (Tab. 12). Perfect Retrieval is shown as the upper bound.

suggests that aggressive sparsification at low ρ paired with sign-conflict resolution (TIES, DARE-Linear) is the most reliable merging recipe in this regime. These results confirm the predicted compute–accuracy trade-off: merging recovers most of MEMORY model's headroom over retrieval methods at substantially lower cumulative cost.

J. Validating evaluation dataset suitability

To assess the suitability of the evaluation datasets for EXECUTIVE model and whether EXECUTIVE model has memorized answers from training data, we evaluate performance both without any context (**No Context**) and with evidence documents (**Perfect Retrieval**), the latter serving as an empirical upper-bound that assumes perfect retrieval of relevant documents.

As shown in Tab. 13, the large disparity in performance between No Context and Perfect Retrieval confirms that these datasets require access to evidence documents to achieve correct answers, validating their suitability for evaluating MEMO.

Unsurprisingly, MuSiQue yields the highest No Context scores, as its Wikipedia-grounded questions fall within models' parametric knowledge. NarrativeQA proves most challenging as it achieves the lowest Perfect Retrieval scores across both EXECUTIVE models, reflecting the demand for careful reasoning over full-length books and movie scripts. BrowseComp-Plus yields the largest disparity between No Context and Perfect Retrieval, with near-zero No Context performance but strong recovery when evidence documents are provided.

These findings confirm that EXECUTIVE model heavily relies on evidence documents across all three datasets to perform well. MuSiQue tests multi-hop factual reasoning where parametric knowledge provides partial signals, NarrativeQA tests narrative comprehension that remains challenging even with perfect context, and BrowseComp-Plus tests the ability to exploit retrieved documents for facts otherwise entirely inaccessible to the model.

Table 12. **Sweep of all 14 merge configurations on NarrativeQA.** Two MEMORY models (Qwen2.5-14B-Instruct) are independently SFT-trained on the disjoint NarrativeQA.1 and NarrativeQA.2 subsets; each subset's best-performing checkpoint provides the task vector entering the merge. EXECUTIVE model = Qwen2.5-32B-Instruct; results are mean $\pm$ std. dev. over 3 runs. Best merge in **bold**; full-retrain accuracy (26.85 ± 0.39) is shown for reference. Hyperparameter conventions: $t \in [0, 1]$ is the SLERP interpolation factor along the unit sphere connecting the two task vectors (t=0 recovers MEMORY model on NarrativeQA.1, t=1 recovers MEMORY model on NarrativeQA.2, t=0.5 is the geodesic midpoint); $\rho \in (0, 1]$ is the sparsification density — the fraction of largest-magnitude task-vector entries kept (TIES) or the keep probability for random-drop sparsification (DARE, DARE-TIES). Linear and Task arithmetic merge with uniform weights ($\lambda_i = 1$) and have no hyperparameter.

Method family	Hyperparameter	Accuracy (%)
Linear	—	11.60 ± 1.02
Task arithmetic	—	12.74 ± 1.75
SLERP	$t = 0.3$	11.60 ± 2.24
	$t = 0.5$	7.85 ± 1.71
	$t = 0.7$	11.60 ± 2.13
TIES	$\rho = 0.3$	$\mathbf{15.81 \pm 0.39}$
	$\rho = 0.5$	12.17 ± 1.94
	$\rho = 0.7$	12.06 ± 2.58
DARE-Linear	$\rho = 0.3$	15.47 ± 0.79
	$\rho = 0.5$	9.78 ± 1.20
	$\rho = 0.7$	13.65 ± 2.08
DARE-TIES	$\rho = 0.3$	11.72 ± 0.52
	$\rho = 0.5$	12.97 ± 1.23
	$\rho = 0.7$	11.04 ± 1.20

Table 13. Performance gap between no context and perfect retrieval across datasets and EXECUTIVE models.

	Qwen2.5-32B-Instruct			Gemini-3-Flash		
	BrowseComp-Plus	**NarrativeQA**	**MuSiQue**	**BrowseComp-Plus**	**NarrativeQA**	**MuSiQue**
No Context	0.67 ± 0.00	5.01 ± 0.52	9.03 ± 0.06	1.33	27.30	30.80
Perfect Retrieval	79.67 ± 1.45	51.54 ± 0.34	51.53 ± 0.74	88.33	59.39	66.90

K. Evaluation details

K.1. Implementation details

The current temperature settings are described in Tab. 14. Stage 1 only has a budget of 1 interaction, Stage 2 has a budget of 7 interactions, Stage 3 has a budget of 8 interactions.

Beyond what is described in Sec. C.4, there are additional helper functions that help manage failure modes across Stage 2 and Stage 3. Within Stage 2, the uncertain answer streak tracker is called at the start of every entity-pinning interaction and its output is passed directly into the entity-pinning prompt, giving the EXECUTIVE model a live view of which candidates the MEMORY model has repeatedly failed to corroborate. This allows the EXECUTIVE model to continuously re-rank and prune the candidate pool as evidence accumulates. When Stage 2 concludes without a confirmed entity, either because the EXECUTIVE model explicitly exhausts its options or the interaction budget is reached, the best candidate selector acts as the bridge into Stage 3 by returning the top-ranked candidate. In cases where multiple candidates share the highest rank, the first candidate in the order produced by EXECUTIVE model is selected. In both cases, the downstream Stage 3 prompt is informed of whether the entity was formally confirmed or merely a best guess. Finally, if Stage 3 reveals that the Stage 2 entity was incorrect due to persistent MEMORY model failures, the entity pivot mechanism allows the EXECUTIVE model to nominate a replacement entity mid-stage. The confirmed entity is then overwritten and marked as unconfirmed, ensuring subsequent stages treat it with appropriate uncertainty rather than the confidence of a fully pinned entity.

Table 14. Temperature Configuration of each Stage from Sec. C.4

Stage	Model	Temperature Value	Intent
Evaluation Stage 1 – Grounding	EXECUTIVE model	0.4	Moderate exploration to generate diverse but focused sub-questions
Evaluation Stage 1 – Grounding	MEMORY model	0.1	Near-deterministic to ensure stable, consistent grounding answers
Evaluation Stage 2 – Entity identification	EXECUTIVE model	0.4	Moderate exploration to identify varied candidate entities without excess noise
Evaluation Stage 2 – Entity identification	MEMORY model	0.1	Near-deterministic to produce reliable entity-targeted answers
Evaluation Stage 3 – Answer Seeking	EXECUTIVE model	1.0	High exploration to maximally diversify sub-questions once the entity is confirmed
Evaluation Stage 3 – Answer Seeking	MEMORY model	0.3	Slightly relaxed determinism to allow nuanced answers while remaining consistent
Final Synthesis	EXECUTIVE model	0.3	Low temperature to produce a consistent final answer

Table 15. Helper Functions for Stage 2 and 3 of the Evaluation Pipeline

Function	Stage	Intent
Track uncertain answer streaks	Stage 2	Maintains a running tally of how many unanswerable questions each candidate entity has accumulated across Stage 2, allowing the EXECUTIVE model to progressively prioritize candidates that the MEMORY model consistently cannot corroborate
Select the best candidate	Stage 2	Fallback bridge from Stage 2 to Stage 3 when entity pinning ends without a confirmed entity. Selects the highest EXECUTIVE model-ranked candidate, with ties broken by the order in which the MEMORY model produced the candidates
Entity pivot correction	Stage 3	Allows the pipeline to self-correct mid Stage 3 if the Stage 2 entity proves incorrect. When the EXECUTIVE model nominates a different entity, the confirmed entity is overwritten and marked as unconfirmed so subsequent turns are aware it was not pinned through the full Stage 2 process

K.2. Ablations on evaluation setup

To justify our structured multi-turn evaluation design, we compare against two baselines: a single-turn setup and an *unstructured* multi-turn setup; in both cases, the same trained MEMORY model is used and EXECUTIVE model is held fixed. Results are reported in Tab. 16.

In a single-turn interaction, EXECUTIVE model first determines whether the question requires external memory retrieval, and if so, decomposes it into a set of sub-questions (Stage 1, Sec. C.4) and poses them all simultaneously to MEMORY model. MEMORY model responds to each sub-question independently, and responses indicating uncertainty are discarded before the remaining answers are passed to EXECUTIVE model for final synthesis. This design requires EXECUTIVE model to commit to its full sub-question set before observing any responses, preventing it from reformulating uninformative queries, following up on answers that introduce new candidate entities, or correcting retrievals that are incomplete, contradictory, or anchored to the wrong entity. This is a fundamental limitation that is reflected in its consistently lowest performance across all three datasets (Tab. 16).

A natural extension of the single-turn setting is an unstructured multi-turn interaction, where EXECUTIVE model examines the responses from MEMORY model and decides whether sufficient information has been gathered, or whether additional

Table 16. MEMO accuracy results with Qwen2.5-32B-Instruct as EXECUTIVE model and Qwen2.5-14B-Instruct as MEMORY model across evaluation setups. The best performing epoch was used in comparison across all 3 setups, with mean ± std. dev. reported across 3 runs. Bold results indicate best performing results in the column.

Evaluation Setup	BrowseComp-Plus Accuracy	NarrativeQA Accuracy	MuSiQue Accuracy
Single turn evaluation	32.56 ± 1.58	24.80 ± 0.20	37.57 ± 1.15
Unstructured multi-turn evaluation (15 turns)	47.33 ± 0.88	26.73 ± 2.17	40.13 ± 1.12
Unstructured multi-turn evaluation (50 turns)	48.67 ± 1.00	27.19 ± 0.71	40.57 ± 0.31
Structured multi-turn evaluation (7 Entity Identification turns + 8 Answer seeking turns)	$\mathbf{54.22 \pm 0.84}$	26.39 ± 1.75	$\mathbf{48.30 \pm 1.25}$
Structured multi-turn evaluation (7 Entity Identification turns + 15 Answer seeking turns)	51.44 ± 2.41	$\mathbf{27.76 \pm 0.20}$	47.57 ± 0.95

retrieval rounds are needed (Stage 3, Sec. C.4). In this setting, EXECUTIVE model is presented with the full history of question-answer pairs and prompted to either synthesize a final answer or generate a new batch of sub-questions targeting remaining gaps, repeating for up to T interactions. While iterative retrieval yields clear improvements over the single-turn baseline, performance plateaus quickly when increasing from 15 to 50 interactions (47.33 ± 0.88 to 48.67 ± 1.00 on BrowseComp-Plus, 26.73 ± 2.17 to 27.19 ± 0.71 on NarrativeQA, and 40.13 ± 1.12 to 40.57 ± 0.31 on MuSiQue), suggesting that iterative retrieval alone is insufficient.

The structured multi-turn setup (see Sec. C.4 and App. K.1) outperforms the unstructured multi-turn baseline, with 8 answer-seeking interactions achieving the strongest overall performance for BrowseComp-Plus and MuSiQue. This is consistent with the expectation that explicit entity identification is well-suited to the multi-hop reasoning demands of these datasets. NarrativeQA, which tests discourse understanding over long documents, has the unstructured 15- and 50-interaction baselines (26.73 ± 2.17 and 27.19 ± 0.71) initially outperform the structured setup with 8 answer-seeking interactions (26.39 ± 1.75). Inference logs indicate that EXECUTIVE model rarely utilizes the entity identification stage on NarrativeQA, likely because its questions are less reliant on resolving specific entities. Consequently, the fixed-entity identification budget effectively reduces the number of available answer-seeking interactions compared to unstructured baselines. Increasing the answer-seeking budget to 15 interactions recovers this gap, with NarrativeQA reaching 27.76 ± 0.20, surpassing both unstructured baselines. We hypothesize that this could be due to additional answer-seeking interactions continuing to surface useful signals without the risk of entity drift or state corruption that compound in open-domain multi-hop settings.

Unlike NarrativeQA, structured entity identification and state tracking in BrowseComp-Plus and MuSiQue introduce sensitivity to error accumulation as the number of answer-seeking interactions increases beyond the optimal budget. Additional interactions increase the risk of erroneous MEMORY model responses corrupting the known facts state, and provide more opportunities for EXECUTIVE model to commit to an incorrect intermediate entity via the entity pivot correction helper (Tab. 15). Furthermore, more interactions dilute the correct signal with potentially incorrect answers at the final synthesis stage. These failure modes are partly a function of the reasoning capability of EXECUTIVE model, as structured state maintenance demands strong in-context reasoning to accurately track entities and avoid premature entity commitment. Corroborating this, we observe in Tab. 1 that a stronger reasoning model as EXECUTIVE model yields improved performance when paired with the same MEMORY model, suggesting that these failure modes can be mitigated by scaling the reasoning capability of EXECUTIVE model.

The stage budget used in our experiments was selected without systematic tuning, and alternative settings may yield similar performance with greater token efficiency. We therefore leave a systematic study of the optimal interaction budget and EXECUTIVE model selection as future work.

L. Discussion on number of training epochs

From Figs. 3 to 5, we observe that additional training epochs do not consistently improve accuracy, as peak performance for MEMO mostly occurs at epoch 2 of training the MEMORY model, with marginal gains or mild regression thereafter.

We attribute the early saturation and subsequent regression to overfitting on the SFT corpus, which exhibits substantial lexical overlap across steps by design, as later steps are derived from earlier ones Alg. 1. To quantify this lexical overlap, we compute the lossless compression ratio of the combined QA text across all steps for each dataset by extracting all question and answer strings, concatenating them into a single text corpus, and applying gzip compression at maximum level (compression level 9), where the compression ratio is defined as the ratio of the original text size to the compressed size.

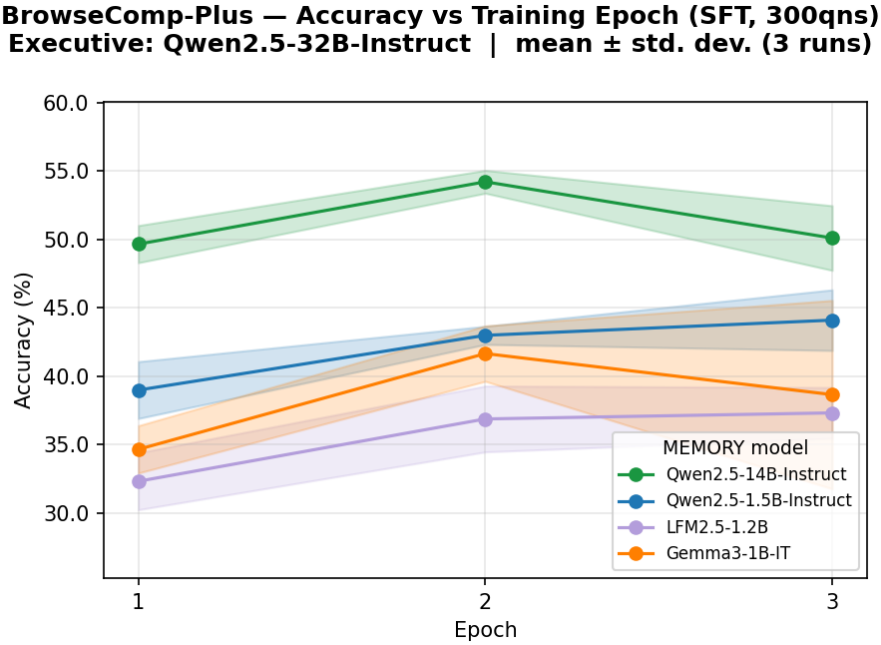

Figure 3. BrowseComp-Plus accuracy (%) vs. training epoch (Full SFT) for each MEMO model size and model family. Lines show the mean over 3 runs, and the shaded band shows ± std. dev. for Qwen2.5-32B-Instruct Runs.

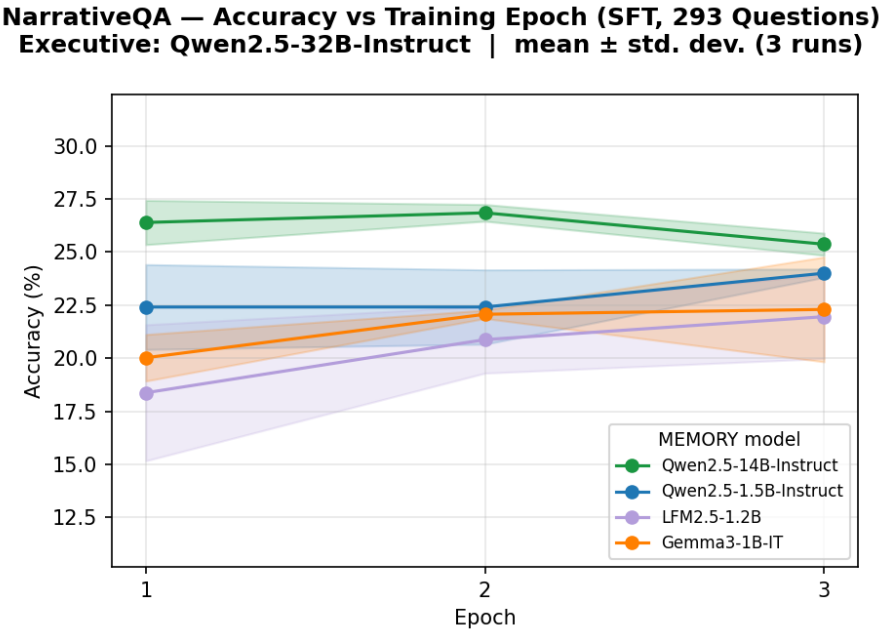

Figure 4. NarrativeQA accuracy (%) vs. training epoch (Full SFT) for each MEMO model size and model family. Lines show the mean over 3 runs, and the shaded band shows ± std. dev. for Qwen2.5-32B-Instruct Runs.

BrowseComp-Plus (1,639,995 pairs) achieves a ratio of $5.80\times$ (82.8% savings), MuSiQue (664,762 pairs) achieves $7.03\times$ (85.8% savings), and NarrativeQA (1,276,676 pairs) achieves $5.45\times$ (81.7% savings), indicating substantial lexical overlap

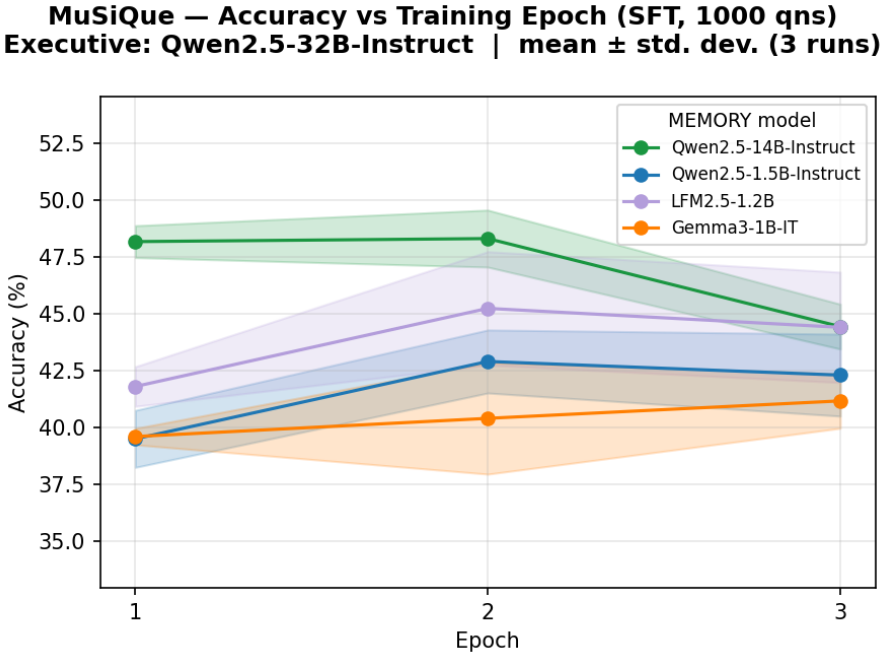

Figure 5. MuSiQue accuracy (%) vs. training epoch (Full SFT) for each MEMO model size and model family. Lines show the mean over 3 runs, and the shaded band shows ± std. dev. for Qwen2.5-32B-Instruct Runs.

within each dataset. We note that compression ratio captures lexical overlap only, and semantic diversity across QA pairs may remain higher, as each step targets distinct reasoning operations ranging from direct fact extraction to cross-document synthesis (see Alg. 1), which is consistent with the impact of removing Step 5 (see App. F).

M. Performance degradation of retrieval-based methods with increasing noise

Tab. 17 reports the performance of two retrieval-based baselines (NV-Embed-V2 and HippoRAG2) under increasing retrieval noise. Both methods degrade monotonically as noise increases, confirming their susceptibility to irrelevant documents. The degradation is most severe for HippoRAG2 on BrowseComp-Plus, which drops 11.55% from $0N$ to $2N$, and for NV-Embed-V2 on MuSiQue, which drops 9.27% over the same range. Notably, even a single negative document per evidence document ($1N$) causes substantial drops of up to 6.22% for both methods on BrowseComp-Plus, suggesting that retrieval-based methods are extremely sensitive to noisy retrieval settings.

Table 17. Accuracy (%) on BrowseComp-Plus and MuSiQue with Qwen2.5-32B-Instruct as EXECUTIVE model. MEMO results are based on Qwen2.5-14B-Instruct and reported at the best training epoch. N is the number of evidence documents present in the target corpus. Δ denotes accuracy difference (%) compared to $0N$.

Method	**Dataset**	$0N$	$1N$		$2N$	
		Acc. (%)	Acc. (%)	Δ	Acc. (%)	Δ
NV-Embed-V2	BrowseComp-Plus	56.89 ± 0.51	50.67 ± 0.33	$\downarrow 6.22$	49.44 ± 0.19	$\downarrow 7.45$
	MuSiQue	42.30 ± 0.53	37.47 ± 0.15	$\downarrow 4.83$	33.03 ± 1.10	$\downarrow 9.27$
HippoRAG2	BrowseComp-Plus	62.33 ± 1.15	56.11 ± 0.51	$\downarrow 6.22$	50.78 ± 1.35	$\downarrow 11.55$
	MuSiQue	47.33 ± 0.74	42.17 ± 0.12	$\downarrow 5.16$	41.70 ± 0.69	$\downarrow 5.63$

N. Ablation on MEMORY model size

Both Qwen2.5-1.5B-Instruct and Qwen2.5-14B-Instruct MEMORYs are trained on the same QA dataset generated by the GENERATOR model (Qwen2.5-32B-Instruct) under training settings described in App. G. Each MEMORY model is evaluated using Qwen2.5-32B-Instruct and Gemini-3-Flash as the EXECUTIVE model.

O. Ablation on MEMORY model family

Each MEMORY model is trained on the same QA dataset generated by GENERATOR model (Qwen2.5-32B-Instruct) and evaluated using Qwen2.5-32B-Instruct and Gemini-3-Flash as EXECUTIVE model. Notably, while Qwen2.5-1.5B-Instruct and Gemma3-1B-IT are based on standard transformer architectures, LFM2.5-1.2B-Instruct adopts a hybrid architecture combining state-space convolution with transformer attention blocks, thereby providing a broader test of MEMORY model across diverse model designs. These models are trained on the same training settings in App. G, with Gemma3-1B-IT using eager attention during training instead of Flash Attention 2.

P. Comparison between full SFT and LoRA

We train all models using LoRA (Hu et al., 2022) applied to the attention and feed-forward projection layers: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. The general LoRA configuration is summarised in Table 18, with model-specific rank and scaling settings reported in Table 19. All remaining training hyperparameters follow Table 10, and per-dataset batch sizes are given in Table 11.

Table 18. LoRA Specific Training Configuration. All other parameters are the same as those in Tab. 10.

Parameter	Value
Target modules	`q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj`
LoRA dropout	0.05
Bias	None
Learning rate	2×10^{-4}

Table 19. Model-Specific LoRA Configuration.

Model	Size	LoRA rank	LoRA alpha	Trainable params
LFM2.5-1.2B-Instruct	1.2B	8	16	6.1M (0.41%)
Gemma3-1B-IT	1B	8	16	6.6M (0.65%)
Qwen2.5-1.5B-Instruct	1.5B	8	16	9.2M (0.60%)
Qwen2.5-14B-Instruct	14B	16	32	68.8M (0.47%)

Table 20. Ablation on LoRA vs Full SFT training across all MEMORY models, evaluated with Qwen2.5-32B-Instruct as EXECUTIVE model. All results are mean ± std. dev. over 3 runs. Bold results indicate best performing results in the column.

	BrowseComp-Plus		NarrativeQA		MuSiQue	
MEMORY model	LoRA	Full SFT	LoRA	Full SFT	LoRA	Full SFT
Gemma3-1B-IT	25.22 ± 1.39	41.67 ± 2.03	21.62 ± 0.86	22.30 ± 2.47	26.17 ± 1.10	41.17 ± 1.20
LFM2.5-1.2B-Instruct	0.78 ± 0.19	37.33 ± 1.86	5.69 ± 0.71	21.96 ± 1.97	7.50 ± 0.26	45.23 ± 2.49
Qwen2.5-1.5B-Instruct	29.78 ± 0.51	44.11 ± 2.22	21.84 ± 0.34	24.00 ± 0.20	31.53 ± 0.55	42.90 ± 1.39
Qwen2.5-14B-Instruct	$\mathbf{48.78 \pm 1.02}$	$\mathbf{54.22 \pm 0.84}$	$\mathbf{23.78 \pm 0.52}$	$\mathbf{26.85 \pm 0.39}$	$\mathbf{43.94 \pm 0.97}$	$\mathbf{50.07 \pm 0.81}$

The notably poor LoRA performance of LFM2.5-1.2B-Instruct can be attributed to its hybrid convolution–attention architecture, which differs from the standard transformer models in our evaluation. Following the LFM2 architecture (Amini et al., 2025), LFM2.5-1.2B-Instruct consists of 16 layers — 6 grouped-query attention (GQA) blocks (at indices $\{2, 5, 8, 10, 12, 14\}$) interleaved with 10 short-range LIV convolution (ShortConv) blocks (Amini et al., 2025). Crucially, the LFM2 attention output projection is named `out_proj` (rather than `o_proj`) and its SwiGLU MLP uses `w1`/`w3`/`w2` (rather than `gate_proj`/`up_proj`/`down_proj`), while the ShortConv blocks expose their own `in_proj` and `out_proj` layers. A LoRA configuration targeting the standard Llama-family module names therefore adapts only a strict subset of the projections that exist in LFM2.5, leaving the remainder frozen. The result is 6.1M trainable parameters

(0.41% of total), disproportionately low given the model size and below our target of ∼0.5%. The rank $r = 8$ was kept fixed across all sub-2B models for a controlled comparison; in retrospect, this penalises LFM2.5-1.2B-Instruct due to its architectural mismatch with the standard Llama-style target set.

Furthermore, the 10 ShortConv blocks which handle the bulk of the model's local feature extraction and the SwiGLU MLPs attached to every block remain entirely unadapted under standard LoRA targeting, severely limiting the adapter's ability to shift the model's behaviour. As shown in Tab. 20, the large performance gap between LoRA and Full SFT confirms that the model is capable of learning the task when all parameters are updated.

Future work could explore LoRA configurations better suited to this architecture: targeting the LFM2-specific module names (`out_proj`, `w1`, `w3`, `w2`) alongside the ShortConv projections (`in_proj`, `out_proj`), as well as tuning the rank and learning rate per architecture rather than holding them fixed across families for the controlled comparison reported here.

Q. Impact statement

MEMO advances the ability of LLMs to internalize knowledge over large, domain-specific corpora without requiring access to model weights, lowering the barrier for deploying capable AI systems in knowledge-intensive domains such as law, medicine, and scientific research. By enabling plug-and-play integration with any LLM, including proprietary models, MEMO democratizes access to powerful knowledge integration capabilities that would otherwise require significant computational resources or white-box model access. At the same time, this accessibility introduces dual-use concerns, as the same capability that enables beneficial applications could be used to internalize misinformation, proprietary data without authorization, or harmful content at scale. Additionally, as MEMO reduces reliance on explicit retrieval, it may obscure the provenance of retrieved information, making it harder to attribute the sources underlying a model's responses. We encourage future work to investigate attribution mechanisms and access controls for memory-based systems, and urge practitioners to carefully consider the nature of the documents used to train MEMORY model.

R. Limitations

MEMO incurs an upfront training cost for each new corpus, and performance may vary across domains, document types, or LLM families beyond those covered in our experiments. Furthermore, the performance of MEMO is inherently bounded by the representational capacity of MEMORY model to internalize the target corpus. Although our experiments do not reveal clear signs that MEMORY model has reached its capacity limit, we hypothesize that sufficiently large or information-dense corpora will exceed what a fixed-size MEMORY model can correctly compress and represent.

S. Future work

We outline several directions for future work. The data generation pipeline is computationally expensive, with Step 5 in Alg. 1 scaling quadratically at $O(k \cdot C^2 \cdot Q^2)$, and reducing this cost remains an open problem. A systematic evaluation of chunking strategies and their associated tradeoffs (App. E) is likewise an open direction. On the training side, scaling MEMORY model with corpus size and developing more effective model merging strategies for reducing per-corpus training costs (App. D.3) are promising directions. Other post-training methods such as Reinforcement Learning (Sutton et al., 1998) have also shown to be effective in improving model task performance (Lambert et al., 2024), and applying such methods to MEMORY model training warrants future investigation.

LoRA configurations better suited to specific architectures, including per-architecture tuning of rank and learning rate, also warrant further investigation (App. P). Finally, a more systematic study of the interaction between EXECUTIVE model reasoning capability and MEMORY model model size (App. D.1), as well as the optimal interaction budget at each stage and EXECUTIVE model selection (App. K.2), are other promising future directions.