MEMO: Memory as a Model

Anonymous Authors[1]

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 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 that enabling plug-and-play integration with both open and proprietary LLMs, and **(e)** its retrieval cost is independent of corpus size at inference time. Our experiments on three benchmarks, BrowseComp-Plus, NarrativeQA, and MuSiQue, show that MEMO achieves strong performance compared to existing methods across diverse settings.

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after pretraining (Xu et al., 2024) until subsequent updates, causing their pretrained knowledge to become increasingly outdated as the world evolves.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) compress knowledge into soft tokens or other model-specific representations, but suffer from representation coupling 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 that encodes new knowledge into a dedicated trained MEMORY model, from which EXECUTIVE model, responsible for reasoning and responding to user queries, retrieves relevant information via targeted sub-queries at inference time and reasons over it to produce a final answer. MEMO combines the complementary strengths of the three paradigms above while mitigating their individual limitations. Specifically, it inherits the plug-and-play flexibility of non-parametric methods while leaving the LLM unchanged, the ability to internalize knowledge in model parameters from parametric methods, and the compact, queryable memory artifact of latent memory methods. As a result, MEMO offers the following advantages: **(a)** it captures complex cross-document relationships, **(b)** it is robust to retrieval

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the ICML 2026 Workshop "Continual Adaptation at Scale: Towards Sustainable AI". Do not distribute.

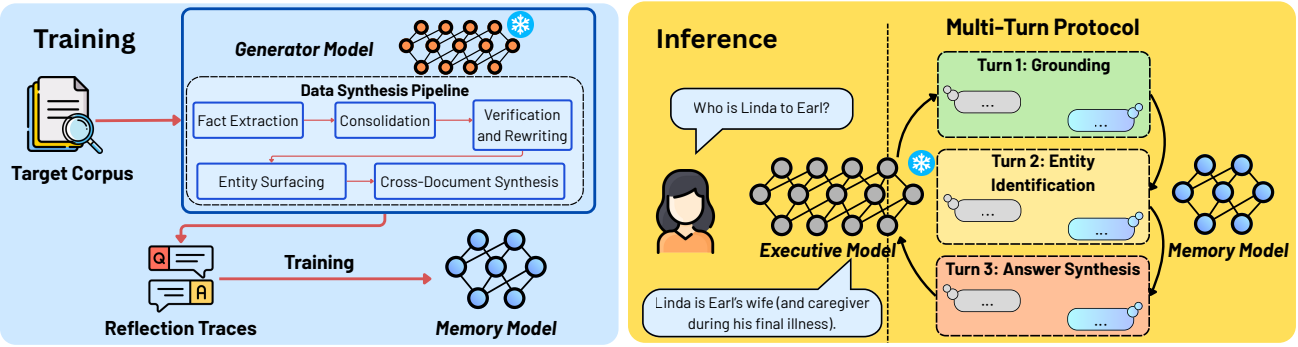

Figure 1. **Overview of MEMO.** During MEMORY model training (left), a frozen GENERATOR model transforms a target corpus into a reflection QA dataset via fact extraction, consolidation, verification, entity surfacing, and cross-document synthesis, which is then used to train a dedicated MEMORY model.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

noise, **(c)** it avoids catastrophic forgetting in the LLM, **(d)** it does not require access to the base LLM'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 training 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 inference-time challenge in mind, we design a novel *five-stage data synthesis pipeline* (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 detailed 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-step inference pipeline* in which EXECUTIVE model, responsible for responding to user queries, interacts with MEMORY model via a structured multi-turn protocol, decomposing complex queries into targeted sub-queries that align with the shared reflection interface (illustrated in Fig. 1 (right) and detailed in Sec. C.3). Unlike retrieval-based methods, this approach incurs retrieval cost 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 both open and proprietary closed-source models.

Our methods is guided by a single design principle: *reflections*, distilled from the target corpus, serve as a shared interface that consolidates relationships into compositional representations. During training, 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-stage data synthesis pipeline that distills a target corpus into reflections, enabling a dedicated MEMORY model to internalize knowledge in compositional forms that capture more complex cross-document relationships and generalize robustly to diverse, unseen query variations at inference time (see Sections C.1 and C.2).
- **Structured multi-turn protocol.** We introduce a *struc-*

tured 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 LLM, including proprietary models, and has retrieval cost independent of corpus size (see Sec. C.3).

- **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 for multi-hop, multi-document retrieval and reasoning. We filter non-English instances using LangDetect (Danilák, 2021) and sample 300 questions; the target corpus is constructed by taking each question's evidence documents alongside an equal number of negative documents[1], yielding 3,541 documents in total. **NarrativeQA** (Kočiskỳ et al., 2018) tests discourse understanding over long documents such as books and movie scripts. We use 293 questions across 10 documents[2]. **MuSiQue** (Trivedi et al., 2022) is a multi-hop question-answering benchmark that requires composing 2–4 reasoning steps across multiple Wikipedia paragraphs. We use 1,000 questions and construct the target corpus in the same manner as BrowseComp-Plus, yielding 5,296 documents in total. Further details are provided in App. D.

Baselines. We compare MEMO against four baselines spanning distinct knowledge-integration paradigms: **BM25** (Robertson & Walker, 1994) (lexical retrieval), **NV-Embed-V2** (Lee et al., 2024) (dense retrieval), **HippoRAG2** (Gutiérrez et al., 2025) (graph-based RAG), and **Cartridges** (Eyuboglu et al., 2025) (a trained KV-cache loaded onto EXECUTIVE model at inference time). We additionally include **Perfect Retrieval** as an *empirical upper bound*, where EXECUTIVE model receives only gold documents in context (Brown et al., 2020), requiring no document retrieval or processing. Retrieval-based baselines use top-k=9 retrieval with adaptive backoff, where k is dynamically reduced until the retrieved context fits within EXECUTIVE model's context window size.

Implementation and evaluation. *(a) Data generation.* We use Qwen2.5-32B-Instruct (Yang et al., 2025) as GENERATOR model, a strong open-weight model with long-context handling capabilities, served via vLLM (Kwon et al., 2023) with YaRN RoPE scaling (Su et al., 2023) to support a 131K-token context. *(b) Training.* We train MEMORY model from Qwen2.5-14B-Instruct for 3 epochs using the fused AdamW (Loshchilov & Hutter, 2017) optimizer and DeepSpeed 2 (Rajbhandari et al., 2020), with a learning rate of 2×10^{-5}. Full hyperparameters are detailed in App. E. *(c) Evaluation.* At inference time, we instantiate EXECUTIVE model with Qwen2.5-32B-Instruct and Gemini-3.0-Flash (Google DeepMind, 2025) to evaluate performance across models with varying reasoning capabilities when augmented with the same trained MEMORY model. We further verify that both models have minimal prior knowledge of the evaluation datasets (App. F). EXECUTIVE model interacts with MEMORY model through the multi-turn protocol described in Sec. C.3. We report binary accuracy, evaluated by Gemini-2.5-Flash-Lite (Comanici et al., 2025) via the DeepEval framework (Ip & Vongthongsri, 2025). Results with Qwen2.5-32B-Instruct are reported as mean $\pm$ standard deviation over three runs, while results with Gemini-3.0-Flash are based on a single run due to API cost constraints.

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, comprising full-length books and movie scripts—MEMO achieves 26.85% with Qwen2.5-32B-Instruct and 53.58% with Gemini-3-Flash, substantially surpassing all baselines. This is particularly notable as NarrativeQA requires reasoning over long documents with complex connections, a setting where retrieval-based methods are fundamentally constrained by context window limits and their inability to model relationships. By contrast, MEMO captures these complex connections via reflections during training and retrieves them precisely through a structured multi-turn protocol during inference. This trend is further supported on MuSiQue, where MEMO achieves 48.30% with Qwen2.5-32B-Instruct and 58.70% with Gemini-3-Flash, outperforming retrieval-based baselines that struggle to compose multi-hop reasoning steps. On BrowseComp-Plus, MEMO achieves 66.67% with Gemini-3-Flash, outperforming all baselines, and remains competitive with Qwen2.5-32B-Instruct (54.22%), narrowly trailing HippoRAG2 (56.11%).

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). This demonstrates that MEMO can be trained once with a weaker GENERATOR model, and seamlessly combined with *any* LLM at inference time, with-

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

[2] We follow the setting of HippoRAG2 and evaluate on 10 documents from the NarrativeQA validation split, which comprises 294 questions; one duplicate is removed for consistency.

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	51.42 ± 0.52	60.41	62.83 ± 0.90	73.00
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	0.00 ± 0.00	-	3.75 ± 0.11	-	8.57 ± 0.40	-
MEMO	54.22 ± 0.84	**66.67**	$\mathbf{26.85 \pm 0.39}$	**53.58**	$\mathbf{48.30 \pm 1.25}$	**58.70**

Table 2. Accuracy (%) on BrowseComp-Plus and MuSiQue with Qwen2.5-32B-Instruct as EXECUTIVE model. Red values denote absolute drop in accuracy (percentage points) relative to the noiseless setting ($0N$). MEMO results are based on Qwen2.5-14B-Instruct and reported at the best training epoch. N = Number of evidence documents present in the target corpus.

Method	Dataset	Number of negative documents per evidence document	
		$0N$	$1N$
NV-Embed-V2	BrowseComp-Plus	56.89 ± 0.51	50.67 ± 0.33 (↓ 6.22)
	MuSiQue	42.30 ± 0.53	37.47 ± 0.15 (↓ 4.83)
HippoRAG2	BrowseComp-Plus	62.33 ± 1.15	56.11 ± 0.51 (↓ 6.22)
	MuSiQue	47.33 ± 0.74	42.17 ± 0.12 (↓ 5.16)
MEMO	BrowseComp-Plus	53.67 ± 1.15	54.22 ± 0.84 (↑ 0.55)
	MuSiQue	50.07 ± 0.81	48.30 ± 1.25 (↓ 1.77)

out retraining. This *plug-and-play* capability allows MEMO to directly leverage state-of-the-art proprietary models without any additional overhead.

2.2. Ablation on the amount of noise for the dataset

We investigate the robustness of MEMO against two strong, retrieval-based methods to increasing retrieval noise, where noise is controlled by the number of negative documents added to the target corpus ($N = 1775$ for BrowseComp-Plus and $N = 2648$ for MuSiQue). As shown in Tab. 2, performance significantly degrades for retrieval-based methods (drops of up to 6.22 pp) as noise increases, confirming that these retrieval systems are highly susceptible to irrelevant documents and struggle in noisy settings. In contrast, MEMO's performance remains relatively consistent for both benchmarks: an improvement of 0.55 pp on BrowseComp-Plus and a negligible drop of 1.77 pp on MuSiQue, both within standard deviation bounds. We attribute this robustness to MEMO's design: rather than retrieving raw documents that may include distractors, MEMORY model internalizes synthesized reflections that consolidate cross-document connections, and responds to EXECUTIVE model's sub-queries with precise, relevant knowledge.

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 weak 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 desired information retrieval from the memory model. 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 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.

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.

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. *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*, 2025.

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, 2024. URL https://arxiv.org/abs/2403.12958.

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, 2025. URL https://arxiv.org/abs/2507.06261.

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.

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, 2023. URL `https://arxiv.org/abs/2211.08411`.

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?, 2024. URL `https://arxiv.org/abs/2207.13332`.

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*, 6:317–328, 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, 2023. URL `https://arxiv.org/abs/2309.06180`.

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, 2021. URL `https://arxiv.org/abs/2005.11401`.

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*, 40(12):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*, 12:157–173, 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.

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. IEEE, 2020.

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

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, 2022. URL https://arxiv.org/abs/2212.13138.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

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.

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

Trivedi, H., Balasubramanian, N., Khot, T., and Sabharwal, A. Musique: Multihop questions via single-hop question composition, 2022. URL https://arxiv.org/abs/2108.00573.

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.

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, 2023. URL https://arxiv.org/abs/2303.17564.

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.

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

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.

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 rather 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). 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 erodes safety alignment (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 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., 2025) 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. Here, we summarize how MEMO differs from the above methods across key architectural and methodological dimensions, highlighting its unique combination of modular memory construction and retrieval-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)	✓	×	✓	✓	×	✓
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 $\mathcal{D} = \{d_1, \ldots, d_N\}$ denote a *target corpus of N documents* containing knowledge largely absent from the pretrained knowledge ($\mathcal{D}_{\text{pre}}$) of $\mathcal{M}_\theta$. Let $\mathcal{Q}$

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

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 target corpus $\mathcal{D}$, find a mechanism (Φ, f) such that, without modifying θ, for all $q \in \mathcal{Q}$, $\mathbb{P}\{f(\mathcal{M}_\theta, \Phi(\mathcal{D}), q) = a^\star(q)\} = 1$.*

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 reflection QA dataset derived from $\mathcal{D}$, and queried by a frozen EXECUTIVE model $\mathcal{M}_\theta$ at inference time.

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 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 MEMORY model to resolve knowledge-intensive questions (see Sections C.1 to C.3).

C.1. Data Generation 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 stages, 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 stage, preventing MEMORY model from exploiting shortcut signals during evaluation.

Stage 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.

Stage 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 $\mathcal{Q}_{\text{con}} = \mathcal{Q}_{\text{dir}} \cup \mathcal{Q}_{\text{indir}} \cup \mathcal{Q}_{\text{mrg}}$.

Stage 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 $\mathcal{Q}_{\text{ver}}$, self-contained QA pairs suitable for standalone supervision.

Stage 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 $\mathcal{Q}_{\text{ent}}$, mitigate the reversal curse (Berglund

Algorithm 1 Reflection QA Dataset Generation Pipeline from Target Corpus

Require: Corpus $\mathcal{D}$, generator $\mathcal{M}_{\text{gen}}$, document groups $\mathcal{G} = \{G_1, \ldots, G_k\}$ 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**
    $\mathcal{Q}_{\text{dir}},\ \mathcal{Q}_{\text{indir}} \leftarrow \mathcal{M}_{\text{gen}}(c)$ ▷ Stage 1: Direct and indirect extraction
    $\mathcal{Q}_{\text{raw}} \leftarrow \mathcal{Q}_{\text{dir}} \cup \mathcal{Q}_{\text{indir}}$ ▷ Stage 2a: Merge direct and indirect
    $\mathcal{Q}_{\text{mrg}} \leftarrow \mathcal{M}_{\text{gen}}(\mathcal{Q}_{\text{raw}})$ ▷ Stage 2b: Consolidate related pairs
    $\mathcal{Q}_{\text{con}} \leftarrow \mathcal{Q}_{\text{raw}} \cup \mathcal{Q}_{\text{mrg}}$ ▷ Stage 2c: Full merge set
    $\mathcal{Q}_{\text{ver}} \leftarrow \mathcal{M}_{\text{gen}}(\mathcal{Q}_{\text{con}},\ c)$ ▷ Stage 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
  $\mathcal{Q}^d_{\text{ent}} \leftarrow \mathcal{M}_{\text{gen}}(\mathcal{Q}^d_{\text{ver}})$ ▷ Stage 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**
  $\mathcal{Q}_{\text{cross}} \leftarrow \mathcal{M}_{\text{gen}}\Big(\bigcup_{d \in G_i} \big(\mathcal{Q}^d_{\text{ver}} \cup \mathcal{Q}^d_{\text{ent}}\big)\Big)$ ▷ Stage 5: Cross-document synthesis
  $\mathcal{Q}_{\text{final}} \leftarrow \mathcal{Q}_{\text{final}} \cup \mathcal{Q}_{\text{cross}}$
end for
return $\mathcal{Q}_{\text{final}}$

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.3).

Stage 5: Cross-document synthesis. The final stage operates over pre-defined document groups $\mathcal{G} = \{G_1, \ldots, G_k\}$, 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 the entity-surfacing pairs $\mathcal{Q}^d_{\text{ent}} : d \in G_i$ 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 $s(q) > 1$ (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 reasoning signals for training MEMORY model.

C.2. Training the MEMORY model

Given $\mathcal{Q}_{\text{final}}$, MEMORY model is trained via supervised fine-tuning to map questions directly to answers *without* access to source documents at inference time. MEMORY model is initialized from a small pretrained language model, substantially 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 knowledge.

C.3. Inference-Time Integration

At inference time, EXECUTIVE model and MEMORY model interact through a structured multi-turn protocol, with EXECUTIVE model treating MEMORY model as an external knowledge oracle. Rather than a single query–response exchange, the interaction proceeds through three sequential turns, each designed to progressively improve the likelihood of producing a correct final answer, as illustrated in Fig. 1 (right). Each turn employs distinct prompts and sampling temperatures, with independent per-turn budgets controlling the number of queries. We have provided full details in App. G.

Turn 1: Grounding. Given a query q, EXECUTIVE model decomposes it into a set of atomic, clue-probing sub-questions $\{q'_1, \ldots, q'_K\}$, where each sub-question targets a single identifying constraint in q, and K is adaptively determined by EXECUTIVE model. The MEMORY model answers each sub-question independently, without shared context, producing grounding responses $\{m_1, \ldots, m_K\}$. These responses provide additional contextual grounding for subsequent turns, drawing on MEMORY model's parametric knowledge rather than relying solely on that of EXECUTIVE model.

Turn 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 turns. This process continues until EXECUTIVE model converges on a single entity $e^\star$ or the per-turn budget is exhausted. If no candidates are identified, Turn 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).

Turn 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 turn budget is exhausted, EXECUTIVE model synthesizes the accumulated responses into a final answer: $\hat{a} = \mathcal{M}_\theta\big(q,\ \{m_k\}_{k=1}^{K},\ e^\star,\ m_{\text{seek}}\big)$.

Notably, the MEMORY model responses m_k 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.

D. Preparation of Datasets

Corpus construction. Extending from our description in Sec. 2, we distinguish between two types of documents[4]: 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 in total. For MuSiQue, we used $2{,}648$ documents for each of the evidence and negative documents, yielding $5{,}296$ documents in total. NarrativeQA does not have negative documents.

Chunking strategy. As shown in Tab. 4, 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 Stage 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. 5. Unlike NarrativeQA, MuSiQue documents are compact with 99.70% falling below 512 tokens, and each MuSiQue document is treated as a single chunk.

BrowseComp-Plus documents are also treated as a single chunk. The time complexity of Stage 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 Stage 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

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

Table 4. 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

Table 5. 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

MuSiQue due to computational constraints arising from the quadratic scaling of Stage 5. As reported in Tab. 5, 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 Stage 5, this would result in a prohibitive increase in pipeline cost for BrowseComp-Plus ($k = 300$) and MuSiQue ($k = 1{,}000$). Hence, we opted to only include up to $N_{\text{evidence}}^{\text{dataset}}$ number of negative documents for each question in the corpus.

E. MEMORY model hyperparameter settings

Training was conducted on H100 and H200 GPUs using the hyperparameter settings reported in Tab. 6. The effective batch size was selected for each subset based on training stability and downstream evaluation performance, with the full configuration summarized in Tab. 7.

F. Validating evaluation dataset suitability for EXECUTIVE model

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

As shown in Tab. 8, the near-zero No Context scores across both EXECUTIVE models confirm that all three datasets require access to retrieved evidence to answer correctly, validating their suitability for evaluating MEMO. Furthermore, the low

Table 6. 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 7. Effective batch sizes and number of QA pairs used per dataset. NarrativeQA.1 and NarrativeQA.2 are independent subsets partitioned from the original.

Dataset	Target Num of Questions	Num of QA Pairs	Effective Batch Size
NarrativeQA	293	1,276,676	512
MuSiQue	1,000	664,762	256
BrowseComp-Plus	300	1,639,995	512

baseline performance suggests that data contamination during pretraining does not inflate the reported results.

Table 8. Gap between no context and in-context learning across datasets and evaluator models. The mean $\pm$ std. dev. is reported across 3 runs for Qwen2.5-32B-Instruct. Only a single run was carried out for Gemini-3-Flash due to API costs.

	Qwen2.5-32B-Instruct			Gemini-3-Flash		
	BrowseComp-Plus	**NarrativeQA**	**MuSiQue**	**BrowseComp-Plus**	**NarrativeQA**	**MuSiQue**
No Context	0.00 ± 0.00	5.35 ± 0.20	17.03 ± 0.40	1.33	26.62	41.80
In-Context Learning	79.67 ± 1.45	51.42 ± 0.52	62.83 ± 0.90	88.33	60.41	73.00

Comparing datasets under the oracle upper bound, MuSiQue yields the highest No Context scores (17.03 for Qwen2.5-32B-Instruct and 41.80 for Gemini-3-Flash), likely because its questions are grounded in Wikipedia-derived factual relationships that fall within models' parametric knowledge. In contrast, NarrativeQA requires reasoning over specific narrative documents and is unlikely to have been seen during pretraining. BrowseComp-Plus is explicitly constructed to contain obscure and hard-to-retrieve facts, driving its No Context performance near chance by design.

NarrativeQA proves to be the most challenging dataset even under perfect retrieval conditions, achieving the lowest In-Context Learning scores (51.42 and 60.41 respectively), suggesting that its questions demand careful reasoning over long narrative passages rather than simple fact retrieval. BrowseComp-Plus presents a strikingly different profile: near-zero performance without context yet strong recovery under oracle retrieval (79.67 and 88.33), indicating that its answers are not stored in model weights but are readily recoverable once the relevant documents are provided. This contrast highlights that NarrativeQA and BrowseComp-Plus probe distinct failure modes, namely reasoning difficulty and knowledge absence respectively, rather than a single axis of retrieval dependence.

Together, these findings confirm that EXECUTIVE model relies on retrieved evidence rather than prior exposure across all three datasets, and that each benchmark stresses a different capability: MuSiQue tests multi-hop factual reasoning where parametric knowledge provides a partial signal, NarrativeQA tests deep narrative comprehension that remains challenging even with perfect context, and BrowseComp-Plus tests the ability to exploit retrieved documents for facts that are otherwise entirely inaccessible to the model.

G. Eval process and prompt

Beyond what is described in Sec. C.3, there are additional helper functions which help manage failure modes across Turn 2 and Turn 3. Within Turn 2, the uncertain answer streak tracker is called at the start of every entity-pinning turn 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 Turn 2 concludes without a confirmed entity, either because the EXECUTIVE model explicitly exhausts its options or the turn budget is reached, the best candidate selector acts as the bridge into Turn 3 by returning the top-ranked candidate. In cases where multiple candidates share the highest rank, the first candidate in the order produced by the EXECUTIVE model is selected. In both cases, the downstream Turn 3 prompt is informed of whether the entity was formally confirmed or merely a best guess. Finally, if Turn 3 reveals that the Turn 2 entity was incorrect due to persistent MEMORY model failures, the entity pivot mechanism allows the EXECUTIVE model to nominate a replacement entity mid-turn. The confirmed entity is then overwritten and marked as unconfirmed, ensuring subsequent turns treat it with appropriate uncertainty rather than the confidence of a fully pinned entity. The current temperature settings are described in Tab. 9. Turn 1 has 1 interaction, Turn 2 has 7 interactions, Turn 3 has 8 interactions.

Table 9. Temperature Configuration of each Stage from Sec. C.3

Turn	Model	Temperature Value	Intent
Evaluation Turn 1 – Grounding	EXECUTIVE model	0.4	Moderate exploration to generate diverse but focused sub-questions
Evaluation Turn 1 – Grounding	MEMORY model	0.1	Near-deterministic to ensure stable, consistent grounding answers
Evaluation Turn 2 – Entity identification	EXECUTIVE model	0.4	Moderate exploration to identify varied candidate entities without excess noise
Evaluation Turn 2 – Entity identification	MEMORY model	0.1	Near-deterministic to produce reliable entity-targeted answers
Evaluation Turn 3 – Answer Seeking	EXECUTIVE model	1.0	High exploration to maximally diversify sub-questions once the entity is confirmed
Evaluation Turn 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 10. Helper Functions for Turn 2 and 3 of the Evaluation Pipeline

Function	Turn	Intent
Track uncertain answer streaks	Turn 2	Maintains a running tally of how many unanswerable questions each candidate entity has accumulated across Turn 2, allowing the EXECUTIVE model to progressively prioritize candidates that the MEMORY model consistently cannot corroborate
Select best candidate	Turn 2	Fallback bridge from Turn 2 to Turn 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	Turn 3	Allows the pipeline to self-correct mid Turn 3 if the Turn 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 Turn 2 process

The turn budget used in our experiments was selected without systematic tuning, and alternative settings may yield similar performance with greater token efficiency. We leave a principled study of turn budget optimization to future work.