VizoMem: A Visual-Textual Memory Framework for Efficient Long-Horizon Reasoning

Anonymous ACL submission

Abstract

Agentic systems built upon large language models (LLMs) increasingly depend on long-context modeling to support document understanding, long-term memory recall, and multi-step reasoning. However, extending context windows incurs substantial computational and memory overhead, significantly limiting the scalability and practicality of long-context LLM-based agents. Recent studies suggest that visual representations can serve as an effective medium for compressing and organizing long textual content. Motivated by this insight, we propose **VizoMem**, a novel visual memory framework for agentic systems. In this framework, textual memories are pre-rendered into structured images and stored as visual notes, enabling compact and persistent memory representations. Moving beyond standard vision-language models like Glyph, we pioneer a specialized retrieval system designed for large-scale visual memory. Our innovation lies in the construction of a dedicated dataset and the development of a highly efficient retrieval model that repurposes foundational vision-language encoders to navigate complex, text-heavy visual environments. Experiments on public datasets demonstrate that our approach significantly reduces token consumption while preserving effective long-term memory recall, highlighting its potential as a scalable alternative to conventional long-context modeling.

1 Introduction

With the rapid advancement of Large Language Models (LLMs), agentic systems have demonstrated unprecedented capabilities across a diverse range of complex tasks. By incorporating perception–reasoning–action loops, these agents can interact deeply with both physical and digital environments (Liu et al., 2024; Feng et al., 2025). However, as these agents operate, they generate massive volumes of interaction history. Due to the underlying Transformer architecture, extending the context window to millions of tokens incurs prohibitively high computation costs (Vaswani et al., 2017), making the development of efficient Long-term Memory a critical bottleneck (Jiang et al., 2024).

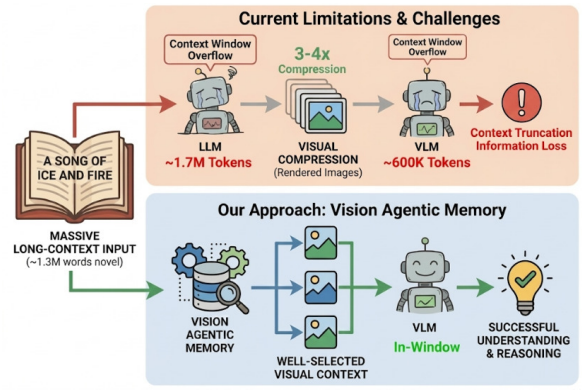

Figure 1: Converting text into visual representations can achieve 3-4x compression. For extremely long documents (e.g., a 1.3M-words book requiring almost 600K vision tokens), this still exceeds a VLM's context window. **VizoMem** integrates visual representations into agentic memory, enabling scalable handling of arbitrarily long background knowledge with high compression ratios while preserving accuracy.

Beyond internal model modifications like context window extension (Peng et al., 2024; Jin et al., 2024) or attention optimization (Chen et al., 2025), the challenge of long-context scaling has converged toward two frontiers: agentic memory system (Zhang et al., 2025) and visual compression techniques. Yet, a fundamental gap remains: current paradigms typically treat visual compression as a static preprocessing tool, which caps its utility in open-ended environments and fails to leverage the structured retrieval capabilities of agentic systems.

This research bridges that gap by drawing inspiration from cognitive psychology, where humans condense complex logical constructs into mental "visual sketches" (Braisby and Gellatly, 2012). We propose a paradigm shift that re-imagines the image as the fundamental unit of a structured, agentic

memory system. By converting textual experiences into "visual notes", we empower agents to manage, organize, and retrieve information within a dense visual manifold. This transition allows visual compression to function not merely as a reduction technique, but as the core representational layer of a scalable memory system (see Figure 1).

Building on this principle, we introduce **VizoMem**, a novel visual agent memory framework centered on a Visual Memory Subsystem. In VizoMem, all memory notes are rendered as images directly from raw textual content using Vision-Language Model (VLMs) like Glyph (Cheng et al., 2025). During reasoning, the agent issues a textual query that triggers retrieval over the image-based memory, enabling it to "see" the global context without reconstructing long textual sequences. This unifies visual compression and agentic memory into a single scalable system. While Glyph alone compresses roughly 240K tokens into 80K, extremely long or continuously growing information still presents challenges. By combining visual compression with agentic memory, VizoMem can organize and retrieve much longer contexts.

More specifically, to support efficient retrieval and reasoning, we construct the GraphMARCO dataset based on MS MARCO (Nguyen et al., 2017) and train a dedicated embedding model for both rendered text images and textual queries. We evaluate our method on two public datasets, LoCoMo (Maharana et al., 2024) and LongMemEval (Wu et al., 2025), performing two main steps for each dataset. First, we directly render raw dialogues into images for retrieval and QA. Second, we design our memory integration by drawing inspiration from the respective state-of-the-art framework. Managing and retrieving memories entirely through rendered images achieves substantial token reduction while maintaining high accuracy.

Our contributions can be summarized as follows:

- We propose **VizoMem**, a visual agent memory framework where textual memories are rendered as images and processed entirely via visual retrieval and question answering. This provides a new perspective on solving long-horizon problems for current agent systems.
- We build the **GraphMARCO** dataset and train a specialized embedding model **ColGlyph** to support efficient retrieval, along with an alignment mechanism to mitigate modality discrepancies.
- We empirically demonstrate that using images as the fundamental memory unit is highly effective. When integrated with advanced memory architectures, it achieves significant token reduction with only minor accuracy loss.

2 Related Work

Agentic Memory. Unlike static and passive RAG systems (Lewis et al., 2020), Agentic memory structures information across multiple memory types—factual, episodic, and working (Hu et al., 2025)—and employs a dedicated agent to orchestrate retrieval, integration, and interlinking of these memories. Such management mechanisms can be broadly instantiated in token-level (Chhikara et al., 2025; Rasmussen et al., 2025), parametric (Xiao et al., 2024), or latent forms (Behrouz et al., 2025), with token-level approaches currently being the most widely adopted. For example, A-MEM (Xu et al., 2025) improves recall accuracy through the temporal evolution and interlinking of multiple discrete memory notes, while MIRIX (Wang and Chen, 2025) categorizes knowledge into six distinct memory types and employs an agent to manage their updates and interactions. Furthermore, MemOS (Li et al., 2025b) stores "memory" in the form of KV-cache, elevating memory to a core AI capability on par with CPU and RAM.

Visual Compression. Recent work has explored transforming textual information into visual representations, enabling more compact and structured encodings of long text. Text or Pixels (Li et al., 2025a) demonstrates that visualized text representations constitute an effective input compression strategy for decoder-only language models. DeepSeek-OCR (Wei et al., 2025) introduces the concept of Contextual Optical Compression, showing that rendering text into images can tightly couple visual understanding with language generation, while Glyph demonstrates that visualized text can substantially reduce token consumption—by up to 3–4×—while preserving strong question-answering accuracy. Taken together, these recent and nearly concurrent works collectively demonstrate the efficiency and practicality of visual text representations for long-context modeling.

We are the first to empirically demonstrate that using images as the fundamental memory unit for agent systems is highly effective. When integrated with advanced memory architectures like MIRIX, it achieves significant token reduction with

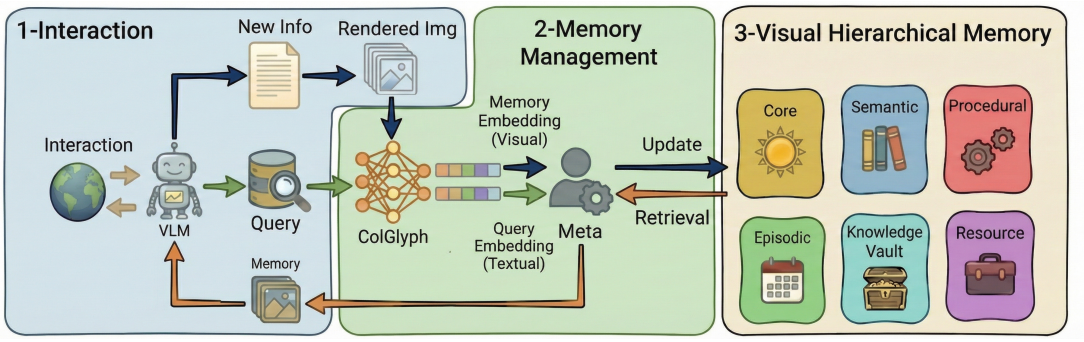

Figure 2: Comprehensive architecture of the VizoMem framework. (1) **Interaction**: The agent interacts with the environment and autonomously decides whether to convert newly acquired information into visual notes or issue a memory query. (2) **Memory Management**: New information is encoded into visual embeddings by the ColGlyph encoder, while queries are encoded into textual embeddings; a Meta Manager coordinates memory updates and retrievals. (3) **Visual Hierarchical Memory**: The memory subsystem follows a MIRIX-inspired hierarchical organization, structuring visual memories into specialized components for efficient long-term storage and retrieval.

only minor accuracy loss. This paradigm offers a novel frontier for addressing the challenges of long-horizon agent reasoning, enabling more scalable and efficient memory management.

3 Method

3.1 Overall Framework

As shown in Figure 2, VizoMem is a novel visual memory framework designed to empower agentic systems with long-horizon context modeling capabilities. Addressing the prohibitive computational and memory overhead inherent in conventional long-context Large Language Models (LLMs), VizoMem introduces a paradigm shift by transitioning from token-heavy textual histories to compact, structured visual representations.

The framework operates through three synergistic pillars:

- **Interaction**: The agent continuously interacts with the environment, producing either *new information* that is rendered and stored as image-based "Visual Notes", or a *textual query* that triggers memory retrieval.
- **Memory Management**: Both visual notes and textual queries are encoded by ColGlyph, a embedding model optimized for text-heavy visual representations. The resulting embeddings are then forwarded to a meta memory manager, which communicates with the corresponding memory agents to carry out memory updates or retrieval operations.
- **Visual Hierarchical Memory**: The system maintains a visual-based hierarchical agentic memory, where image-form visual notes serve as fundamental memory units. Following advanced memory organization designs, this structure supports compact storage, efficient updates, and accurate retrieval for reasoning and VLM-based question answering.

3.2 Retrieval Embedder

Task Definition Images rendered from text constitute a specialized visual representation that requires dedicated embedding models for retrieval. Existing models such as the ColPali (Faysse et al., 2025) series can handle general image embeddings—including mixed content of text, layout, tables, and charts—but their performance on text-rendered images with specific rendering parameters is suboptimal. Inspired by ColPali's approach of creating detailed multi-vector embeddings and performing retrieving via pairwise late-interaction similarity scores, we propose a specialized model, **ColGlyph**, specifically designed for text-rendered images.

Formally, given a batch of query embeddings $Q \in \mathbb{R}^{B\times N_q\times d}$ and passage embeddings $P \in \mathbb{R}^{B\times N_p\times d}$, we first apply L_2 normalization to each vector along the feature dimension:

$$\hat{Q}_i = \frac{Q_i}{|Q_i|_2}, \quad \hat{P}j = \frac{P_j}{|P_j|_2}. \tag{1}$$

This normalization stabilizes training and retrieval across different embedding magnitudes.

The retrieval scores between queries and a passages are then computed using a late-interaction MaxSim operator:

$$\text{score}(Q,P) = \sum_{i=1}^{N_q} \max_{j=1,\ldots,N_p} \hat{Q}_i \cdot \hat{P}_j \tag{2}$$

Here, a *passage* refers to one candidate text-rendered image for retrieval. The hard max operation effectively measures the passage's strongest response to each individual query token, while also mitigating bias introduced by passage length.

In-Batch Negative Sampling During training, we adopt an in-batch negative sampling strategy. For each query in a batch, its corresponding passage serves as the positive sample, while the passages associated with other queries in the same batch are treated as negative samples. Denoting the scores between all queries and all passages in the batch as $S \in \mathbb{R}^{B\times B}$, where

$$S_{mn} = \text{score}(Q_m, P_n), \tag{3}$$

thus, the training objective is a standard cross-entropy loss over the batch:

$$\mathcal{L}_{\text{Train}} = -\frac{1}{B}\sum_{m=1}^{B}\log\frac{\exp(S_{mm})}{\sum_{n=1}^{B}\exp(S_{mn})}. \tag{4}$$

This naturally leverages all other passages in the batch as hard negatives, improving retrieval robustness without additional negative sampling.

Model Design Leveraging Glyph as the backbone, we adopt its vision and language towers to construct our retrieval model for text-rendered images. On top of the backbone, a single-layer MLP projection head maps token-level hidden states to a low-dimensional embedding space, enabling accurate and efficient retrieval. Training details are provided in Appendix B.5.

3.3 Alignment Mechanism for Retrieval

To improve retrieval robustness across modalities, we introduce an alignment-based confidence mechanism that quantifies the consistency between visual and textual representations of same memory.

When rendering a textual passage into an image, we obtain: a textual embedding $L = E_{\text{text}}(p_{\text{l}})$ and a visual embedding $V = E_{\text{vision}}(p_{\text{v}})$.

Using the same late-interaction multi-vector scoring function defined in Eq. (2), we compute two directional self-similarity scores:

$$s_{\text{L}\to\text{V}} = \text{score}(L, V), \tag{5}$$

$$s_{\text{V}\to\text{L}} = \text{score}(V, L). \tag{6}$$

We then define an *alignment score* as a symmetric consistency measure between the visual and textual representations of the same memory:

$$\text{Align}(p) = \frac{2\cdot\min(s_{\text{L}\to\text{V}}, s_{\text{V}\to\text{L}})}{s_{\text{L}\to\text{V}} + s_{\text{V}\to\text{L}} + \epsilon}, \tag{7}$$

where ϵ is a small constant for numerical stability.

During retrieval, given a query q with textual embedding $Q = E_{\text{text}}(q)$, we compute the similarity $\text{score}(Q, V)$. The precomputed alignment score is then used as a memory-level confidence term to modulate the retrieval score:

$$\text{score}'(q, p) = \text{score}(Q, V)\cdot\text{Align}(p)^{\alpha}, \tag{8}$$

where α controls the strength of the alignment adjustment; smaller values of α impose a stronger penalty. In our experiments, we set $\alpha = 0.5$.

Intuitively, memories with consistent visual and textual embeddings receive higher confidence during retrieval, while inconsistent ones are suppressed; a detailed analysis is provided in Appendix A.

3.4 Agentic Memory Designs

To systematically investigate the effects of visual representations within agentic memory systems, we design two agent variants at different levels of sophistication.

3.4.1 Direct Visual Retrieval (DVR)

The first agent adopts a naive implementation strategy. Following the dataset-defined session segmentation, each session is rendered as a single image. During evaluation, a fixed number of session images are retrieved and fed into a VQA model.

3.4.2 Visual Hierarchical Memory (ViHM)

Due to significant differences in architecture performance across datasets, we first evaluated various baselines on both LoCoMo and LongMemEval. For each dataset, we selected the best-performing method for adaptation. Taking LoCoMo as an example, the second agent follows a hierarchical memory organization inspired by the corresponding SOTA design (MIRIX), consisting of **Core Memory** ($\mathcal{C}$), **Episodic Memory** ($\mathcal{E}$), **Resource Memory** ($\mathcal{R}$), **Procedural Memory** ($\mathcal{P}$), **Semantic Memory** ($\mathcal{S}$), **Knowledge Vault** ($\mathcal{K}$) and a **Meta** ($\mathcal{M}$) manages the aforementioned memories. See Appendix B.4 for a detailed description.

We adopt this memory hierarchy while enhancing it with textual memories rendered as visual representations, which are leveraged during retrieval and reasoning.

Formally, let $\mathcal{A} = \{\alpha_{\mathcal{C}}, \alpha_{\mathcal{E}}, \alpha_{\mathcal{R}}, \alpha_{\mathcal{P}}, \alpha_{\mathcal{S}}, \alpha_{\mathcal{K}}\}$ denote the set of specialized memory agents. The system dynamics map an input stimulus x to an updated memory state through a hierarchical process driven by LLM-based reasoning.

Meta-Cognitive Dispatching The meta-agent $\mathcal{M}$ functions as a semantic router. Under the guidance of the meta-prompt $\rho_{\mathcal{M}}$, it maps the input x to a relevant subset of agents $\mathcal{A}^*$:

$$x \xrightarrow{\rho_{\mathcal{M}}} \mathcal{A}^* \subseteq \mathcal{A} \tag{9}$$

This mapping represents the autonomous decision process to distribute information to the appropriate memory domains based on the prompt $\rho_{\mathcal{M}}$.

Memory Evolution For each active agent $\alpha_i \in \mathcal{A}^*$, the evolution of its memory state $S_{i,t}$ is governed by a unified *Update Mechanism*. This mechanism is polymorphic, with execution determined by each agent's unique prompt ρ_i. The process involves two sequential steps:

Step I: Content Extraction. The agent interprets the raw input x to generate a structured memory entity e:

$$e = \text{Extraction}(x \mid \rho_i) \tag{10}$$

Step II: Unified Update Operation. The memory state of agent α_i at time $t+1$ is obtained by applying a general update function $\mathcal{U}$ to the current state $S_{i,t}$ and new input e:

$$S_{i,t+1} = \mathcal{U}(S_{i,t}, e, op) \tag{11}$$

where the atomic operation op is selected from a predefined action set:

$$\Omega = \{\textsc{Insert}, \textsc{Replace}, \textsc{Merge}, \textsc{Skip}\}.$$

The operation is determined by the agent-specific prompt ρ_i via a decision function:

$$op \leftarrow \text{Decide}(S_{i,t}, e \mid \rho_i), \tag{12}$$

and the update function $\mathcal{U}$ executes the selected op. In short, the meaning and application of op, including any fusion of existing and new content, are fully defined by ρ_i, making $\mathcal{U}$ prompt-driven and polymorphic across agents.

4 Theoretical Analysis

4.1 Information Theory Perspective

Let X denote a textual input sequence and Y the corresponding target variable. We contrast the representational efficiency of language and vision processors from an information-theoretic perspective.

Language Processor: Atomic Discretization. Standard language processor maps input X into a sequence Z_{text} of "atomic units" t_i ($i \in [1, L_t]$ and L_t is the sequence length) via a deterministic tokenizer τ_L:

$$\tau_L : X \to Z_{\text{text}} = (t_1, \ldots, t_{L_t}). \tag{13}$$

To quantify the information transfer, we examine the mutual information $I(Z_{\text{text}}; X)$ between the input X and the discrete representation Z_{text}:

$$I(Z_{\text{text}}; X) = H(Z_{\text{text}}) - H(Z_{\text{text}} \mid X). \tag{14}$$

Since the tokenization process τ_L is deterministic, the conditional entropy vanishes. Consequently, the mutual information is strictly limited by the representational entropy: $I(Z_{\text{text}}; X) = H(Z_{\text{text}})$.

Crucially, each token t_i points to a mutually exclusive entry in a fixed vocabulary $\mathcal{V}$. By the **Maximum Entropy Principle**, the capacity of such an atomic slot is bounded:

$$H(t_i) \leq \log |\mathcal{V}|. \tag{15}$$

Since semantic aggregation is deferred to deep attention layers, the processor is restricted to atomic discretization at the input level, implying that total capacity scales linearly with sequence length:

$$I(Z_{\text{text}}; X) = H(Z_{\text{text}}) \leq \sum_{i=1}^{L_t} H(t_i) \leq L_t \cdot \log |\mathcal{V}|. \tag{16}$$

Thus, semantic content exceeding $\log |\mathcal{V}|$ bits requires a linear expansion of L_t to be preserved.

Vision Processor: Composite Aggregation. The rendering process $\phi : X \to \mathbb{R}^{H \times W}$ first aggregates symbols into a unified signal. Leveraging spatial locality, multiple related *glyphs* are physically grouped into local regions $\mathcal{R}_j$. A patch-based tokenizer τ_V then maps these regions into a sequence Z_{visual} of visual embeddings $\mathbf{v}_j$ ($j \in [1, L_v]$ and L_v is sequence length):

$$\tau_V : \mathbb{R}^{H \times W} \to Z_{\text{visual}} = (\mathbf{v}_1, \ldots, \mathbf{v}_{L_v}), L_v \ll L_t. \tag{17}$$

Unlike language tokens, each visual unit $\mathbf{v}_j$ is a composite representation that aggregates features from all glyphs within its spatial region $\mathcal{R}_j$. The resulting reduction in sequence length imposes a structural information bottleneck:

$$I(Z_{\text{visual}}; X) \leq \mathcal{H}(Z_{\text{visual}}) \sim \mathcal{O}(L_v) \ll \mathcal{O}(L_t). \tag{18}$$

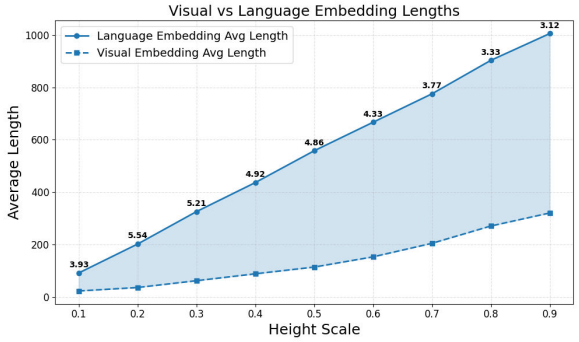

Figure 3: Comparison of average embedding lengths between original text and vision representations under different image sizes (Height Scale).

where the notation $\mathcal{O}(\cdot)$ signifies that the maximum information capacity scales linearly with the sequence length. This severe constraint compels the model to capture highly abstract semantic features.

Empirical Validation and Implication. The validity of this aggregation-first paradigm is empirically substantiated by Glyph (Cheng et al., 2025). Experimental results demonstrate that task performance is preserved ($I(Z_{\text{visual}}; Y) \approx I(Z_{\text{text}}; Y)$) even with a compression ratio of $k \in [3, 4]$.

Finally, the 2D spatial structure S stabilizes this aggregation (i.e., $I(Z_{\text{visual}}; S) > I(Z_{\text{text}}; S)$). For instance, a sequence "Hello World" can be rendered into a single visual patch, where relative pixel coordinates (left-to-right) implicitly encode the sequential order within the fused representation.

Given that the total mutual information with the target Y remains unchanged, the information density per token must shift proportionally:

$$I(\mathbf{v}; Y) \approx k \cdot I(t; Y), \tag{19}$$

where $I(\cdot; Y)$ denotes the average mutual information with the target variable per unit. This confirms that each visual token aggregates the joint information of approximately k textual tokens in a single processing step.

4.2 Retrieval Perspective

Retrieval is performed using a multi-vector similarity function, as defined in Eq. (2). For a textual passage $P = \{p_1, \ldots, p_{L_t}\}$, the contribution of a query token q_i to the retrieval score is

$$\max_{1 \le j \le L_t} \langle q_i, p_j \rangle. \tag{20}$$

Under visual memory compression, the passage is represented as $M = \{\mathbf{v}_1, \ldots, \mathbf{v}_{L_v}\}$, where each visual token aggregates information from approximately k textual tokens. Accordingly, the MaxSim operation becomes

$$\max_{1 \le j \le L_v} \langle q_i, \mathbf{v}_j \rangle, \tag{21}$$

where $\mathbf{v}_j \sim \phi\big(\{p_\ell\}_{\ell \in \mathcal{I}_j}\big), \quad |\mathcal{I}_j| \approx k$. Here, $\phi(\cdot)$ is used as an abstract notation indicating that a single visual token $\mathbf{v}_j$ aggregates information from approximately k textual tokens. As shown in Figure 3, this aggregation is supported by the difference in embedding lengths between visual and textual tokens. This token-level aggregation reduces fragmentation in similarity matching and enables queries to align with semantically richer units, yielding more reliable retrieval scores under the same scoring function.

From both information compression and retrieval efficiency perspective, the above-mentioned analysis and results indicate that integrating visual representations into agentic memory is not only feasible but also highly effective.

5 Experiments

5.1 Experimental Setup

Datasets. Following prior evaluations of agentic memory systems, we adopt the LoCoMo dataset (Maharana et al., 2024) for comparison. LoCoMo features long-term conversations spanning up to 35 sessions, with each conversation averaging hundreds of turns and ~26K tokens. Each conversation is paired with ~200 questions, making it ideal for assessing long-term memory, dependencies, and consistency in extended dialogues.

In addition, we evaluate our method on another long-dialogue dataset, LongMemEval (Wu et al., 2025). Compared to LoCoMo, LongMemEval has much longer average contexts (~115K tokens) with each context associated with a single question, totaling 500 questions. However, the overall textual coherence is lower, with weaker relationships between target and distractor statements. Experimental details and evaluation results on LongMemEval, are provided in Appendix C.2.

Baselines. We compare our method against a range of representative systems, including **Mem0** (and its knowledge-graph variant) (Chhikara et al., 2025), **A-Mem** (Xu et al., 2025), **OpenAI**, **LangMem**, **ZEP** (Rasmussen et al., 2025), **MemOS** (Li et al., 2025b), **MIRIX** (Wang and Chen, 2025), and a **Full-Context** baseline; see Appendix B.2 for detailed descriptions of each baseline.

Method	Single Hop	Multi-Hop	Open Domain	Temporal	Overall
Comparison with Ours (DVR)					
Mem0	68.97	51.42	**72.92**	56.07	63.31
Mem0^g	69.32	51.77	73.96	59.19	64.29
A-Mem	39.24	26.60	54.17	49.84	40.06
LangMem	70.27	57.09	55.21	56.39	64.03
OpenAI	61.24	58.87	62.50	24.92	53.31
Ours (DVR)	**85.10**	**59.69**	52.08	**59.50**	**73.05**
Comparison with Ours (ViHM)					
Zep	80.26	71.99	**70.83**	82.87	78.70
MemOS	78.12	67.02	63.51	80.06	75.58
MIRIX	**83.95**	**82.27**	63.54	**86.92**	**82.99**
Ours (ViHM)	83.63†	76.24	65.62†	86.09†	81.67†
Full-Context	88.59	78.01	71.88	92.83	86.49

Table 1: **LLM-as-a-Judge score** (%, higher is better) comparison under different graph-based agentic memory systems on LoCoMo. † indicates performance that is within a small margin of, or exceeds, the MIRIX baseline, showing no significant degradation despite **substantially reduced token usage**. Since the average sequence length can still fit within the context window, the *Full-Context* setting can be regarded as an approximate upper bound.

Evaluation Metrics. Although early agentic memory systems are often evaluated using automatic metrics such as F1 and BLEU-1, these metrics are sensitive to the surface form of answers and assume strong alignment between outputs and references, which can lead to biased or unreliable scores even when the answers are semantically correct. In this paper, we primarily adopt an LLM-as-a-Judge evaluation protocol, while the automatic metrics are still reported in the ablation studies; see Appendix B.3 for detailed implementation and comparison of the evaluation metrics.

Implementation Details. For fair comparison, we standardize all candidate methods by replacing their backbone models with GPT-4.1-mini. Specifically, we also use GPT-4.1-mini as the judge to evaluate whether a model response successfully addresses the question, based on the question, the ground-truth answer, and the generated response. The prompt for judging and building memories follow the templates provided in Appendix D.1.

The questions include single-hop, multi-hop, temporal reasoning, and open-domain types. LoCoMo also contains an adversarial category for unanswerable questions, but this can mislead evaluation: a model without memory may appear correct if the question happens to be unanswerable (Chhikara et al., 2025). We therefore exclude this category to ensure a fair comparison.

5.2 Results on Performance

For the baselines, we follow the official implementations and conduct a single evaluation on the dataset. For our method, we perform three independent runs and report the averaged results. The results are shown in Tables 1 (performance) and 2 (token consumption).

Method	Token Consumption	Overall
LangMem	185	67.86
OpenAI	4,324	53.31
Mem0	1,291	63.31
Ours (DVR)	2,683	73.80
MemOS	1,731	75.58
ZEP	2,231	78.70
MIRIX	18,803	82.99
Ours (ViHM)	**4,612**	81.17
Full-Context	23,587	86.49

Table 2: Comparison of token consumption and overall LLM-as-a-Judge scores across different memory systems on LoCoMo.

Direct Visual Retrieval Performance. Without applying any structured memory or agentic memory mechanisms, the Naive method simply renders the original information as images and performs retrieval. Despite its simplicity, it achieves significantly higher overall scores under the LLM-as-a-Judge metric compared to the first group of earlier methods. In particular, the Direct method outperforms all methods in the single-hop category, and even ranks highest across all baseline comparisons. This can be attributed to the challenge of generating accurate embeddings for retrieval from extremely long texts: effectively organizing information is crucial for memory-based systems. By representing information visually, the embeddings become

Method	LLM-Judge	BLEU-1	F1	ROUGE-L	METEOR	SBERT
ColGlyph + Alignment (Ours)	**0.738**	**0.433**	**0.513**	**0.518**	**0.382**	**0.698**
w/o Alignment	0.719	0.420	0.497	0.497	0.369	0.671
w/o ColGlyph (ColQwen2) & Alignment	0.693	0.403	0.489	0.494	0.355	0.673

Table 3: Ablation study on the impact of ColGlyph and alignment method.

more compact and robust, leading to superior performance on tasks that rely on Single Hop retrieval.

Visual Hierarchical Memory Performance. Since this paper explores the integration of visual representations into agentic memory systems, we focus on comparing with MIRIX, which currently achieves state-of-the-art performance on LoCoMo. While the visual representation leads to a noticeable drop in the multi-hop category, overall accuracy is largely preserved and token consumption is substantially reduced, with open-domain performance even improving compared to the original MIRIX implementation. This highlights the main advantage of visual memory: maintaining task-relevant accuracy while improving efficiency.

5.3 Ablation Study

Table 3 presents an ablation study on the memory embedder and alignment mechanism, evaluated with LLM-as-a-Judge.

Replacing ColGlyph with the general-purpose ColQwen2.5 (Faysse et al., 2025) and removing alignment consistently degrades performance, notably LLM-Judge (0.738 $\rightarrow$ 0.693) and SBERT similarity, highlighting the effectiveness of a specialized embedder for text-rendered visual memory.

Removing the alignment module while keeping ColGlyph also lowers LLM-Judge (0.738 $\rightarrow$ 0.719), indicating that alignment reduces cross-modal mismatch and improves retrieval accuracy.

In summary, ColGlyph drives the main performance gain, and alignment further enhances precision by mitigating modality discrepancies.

5.4 Effect of Image Size and DPI

In this part, we fix all other image parameters and vary the height scale to adjust image size, considering both high and low DPI settings. We render and encode a large number of texts, then compute $s_{\mathrm{L}\rightarrow\mathrm{V}}$ and $s_{\mathrm{V}\rightarrow\mathrm{L}}$ following Eqs. (5) and (6).

As shown in Fig. 4, increasing image size generally increases the discrepancy between vision and text embeddings, with LVScore showing larger variance. The Alignment Score decreases with larger images, and this effect is more pronounced and less stable at lower DPI settings. These results suggest that for text-rendered images, post-processing based on the Alignment Score is crucial for reliable retrieval similarity, particularly when using lower-resolution images.

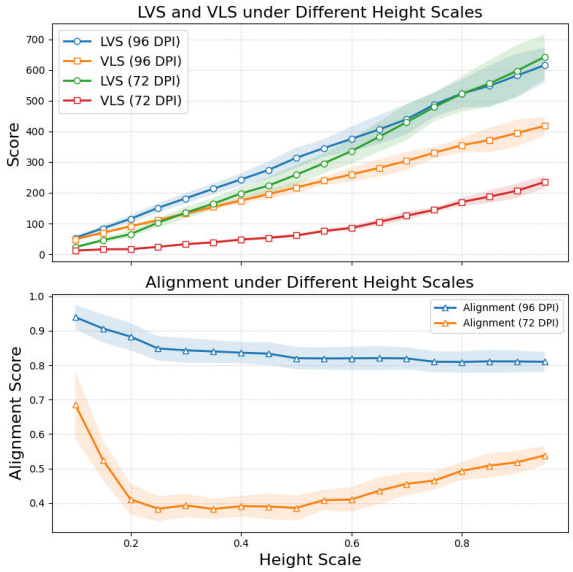

Figure 4: (Upper) Mean and standard deviation of $s_{\mathrm{L}\rightarrow\mathrm{V}}$ (LVScore) and $s_{\mathrm{V}\rightarrow\mathrm{L}}$ (VLScore) under different height scales and DPIs. (Lower) Mean and standard deviation of the Alignment Score under different height scales and DPIs.

6 Conclusion

In this work, we present VizoMem , a visual-textual memory framework for agentic systems. By rendering textual information into compact visual notes and integrating them into a structured agentic memory, VizoMem reduces token usage while preserving high-fidelity recall and reasoning. A specialized retrieval model, ColGlyph, trained on the GraphMARCO dataset, enables precise search over large-scale image-based memory banks, overcoming limitations of standard vision-language encoders in text-heavy settings. Empirical results on the LoCoMo and LongMemEval benchmark show that VizoMem effectively compresses long conversational histories and scales with advanced memory hierarchies, making visual compression a practical enabler for long-horizon agent memory.

Limitations

Integration with Structured Memory Operations. While VizoMem effectively compresses and retrieves long-term memories, purely visual representations cannot fully replace structured data formats in agentic memory systems. Core operations such as organizing, updating, and coordinating different memory types (e.g., Core, Episodic, and Semantic memory) still rely on structured representations, including text-based schemas or LLM-invoked tools. Integrating vision-based memories into these processes introduces additional system complexity and remains an open challenge for future work.

Scope Limited to Text-Derived Visual Representations. The current work primarily explores the feasibility of incorporating visual representations rendered from pure text into agentic memory systems and therefore remains fundamentally text-centric. A significant direction for future research lies in extending this approach to a general-purpose multimodal architecture that inherently supports heterogeneous sensory inputs, including images, audio, and video.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. 2025. Titans: Learning to memorize at test time. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Nick Braisby and Angus Gellatly. 2012. *Cognitive psychology*. Oxford University Press.

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, Chengjun Xiao, Chengyu Du, Chi Zhang, Chu Qiao, Chunhao Zhang, Chunhui Du, Congchao Guo, Da Chen, Deming Ding, and 80 others. 2025. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *CoRR*, abs/2506.13585.

Jiale Cheng, Yusen Liu, Xinyu Zhang, Yulin Fei, Wenyi Hong, Ruiliang Lyu, Weihan Wang, Zhe Su, Xiaotao Gu, Xiao Liu, and 1 others. 2025. Glyph: Scaling context windows via visual-text compression. *arXiv preprint arXiv:2510.17800*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, CELINE HUDELOT, and Pierre Colombo. 2025. Colpali: Efficient document retrieval with vision language models. In *The Thirteenth International Conference on Learning Representations*.

Yiyang Feng, Yichen Wang, Shaobo Cui, Boi Faltings, Mina Lee, and Jiawei Zhou. 2025. Unraveling misinformation propagation in LLM reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11683–11707, Suzhou, China. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, and 28 others. 2025. Memory in the age of ai agents. *Preprint*, arXiv:2512.13564.

Xun Jiang, Feng Li, Han Zhao, Jiahao Qiu, Jiaying Wang, Jun Shao, Shihao Xu, Shu Zhang, Weiling Chen, Xavier Tang, and 1 others. 2024. Long term memory: The foundation of ai self-evolution. *arXiv preprint arXiv:2410.15665*.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. Llm maybe longlm: Self-extend llm context window without tuning. *Preprint*, arXiv:2401.01325.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Yanhong Li, Zixuan Lan, and Jiawei Zhou. 2025a. Text or pixels? it takes half: On the token efficiency of visual text inputs in multimodal llms. *arXiv preprint arXiv:2510.18279*.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, and 20 others. 2025b. Memos: A memory os for ai system. *Preprint*, arXiv:2507.03724.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2017. MS MARCO: A human-generated MAchine reading COmprehension dataset.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yu Wang and Xi Chen. 2025. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*.

Haoran Wei, Yaofeng Sun, and Yukun Li. 2025. Deepseek-ocr: Contexts optical compression. *Preprint*, arXiv:2510.18234.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47.

A Alignment Score

Given a late-interaction multi-vector similarity function, the similarity between two representations $A = \{a_i\}_{i=1}^{|A|}$ and $B = \{b_j\}_{j=1}^{|B|}$ is defined as

$$\text{score}(A, B) = \sum_{i=1}^{|A|} \max_j \; a_i^\top b_j. \tag{22}$$

When a memory passage p is rendered into an image, two representations of the same content are obtained: a textual embedding $L = E_{\text{text}}(p_{\text{l}})$ and a visual embedding $V = E_{\text{vision}}(p_{\text{v}})$. Although text-rendered images ideally induce equivalent retrieval scores across modalities, practical factors such as encoder asymmetry and rendering artifacts often result in misalignment.

Given a query q with embedding $Q = E_{\text{text}}(q)$, retrieval is performed by matching Q against the visual representation V of each memory passage.

Under ideal cross-modal consistency, for each visual token $v_j \in V$ there exists a corresponding textual token $l_{\pi(j)} \in L$ such that their interactions with any query token $q_i \in Q$ are similar:

$$q_i^\top v_j \approx q_i^\top l_{\pi(j)}, \quad \forall q_i \in Q. \tag{23}$$

Substituting Eq. (23) into Eq. (22) yields

$$\text{score}(Q, V) \approx \text{score}(Q, L), \tag{24}$$

indicating that, for a well-aligned memory, the query should induce similar relevance scores regardless of whether the memory is represented in the visual or textual modality.

However, the late-interaction score is not symmetric. In particular, we define two directional similarities:

$$s_{\text{Q}\to\text{V}} = \text{score}(Q, V), \tag{25}$$

$$s_{\text{V}\to\text{Q}} = \text{score}(V, Q), \tag{26}$$

which generally satisfy $s_{\text{Q}\to\text{V}} \neq s_{\text{V}\to\text{Q}}$ in practice.

To capture the internal cross-modal consistency of a memory passage, we define an alignment score as

$$\text{Align}(p) = \frac{2 \cdot \min(s_{\text{Q}\to\text{V}}, s_{\text{V}\to\text{Q}})}{s_{\text{Q}\to\text{V}} + s_{\text{V}\to\text{Q}} + \epsilon}, \tag{27}$$

where ϵ is a small constant for numerical stability.

The alignment score is used to modulate the original query-to-vision similarity:

$$\text{score}'(q, p) = \text{score}(Q, V) \cdot \text{Align}(p)^\alpha, \tag{28}$$

where α controls the strength of the alignment correction.

B More Implementation Details

B.1 Data Construction

Following the optimal rendering configuration identified in the Glyph, we render a new image-based dataset from MS MARCO. The detailed rendering parameters are provided in Appendix B.5.

For each pair $(\text{query}, \text{passage})$, we concatenate all text in the passage associated with the query and render it into a single text-rendered image, which serves as the positive passage during training. As discussed earlier, no explicit negative passage rendering is required.

B.2 Baselines

- **Mem0** (Chhikara et al., 2025): A system that extracts salient information using LLM-based summarization and incrementally updates memory through an extraction–update pipeline, optionally organizing the stored knowledge in a graph-based representation.
- **A-Mem** (Xu et al., 2025): A Zettelkasten-inspired memory system that forms an interconnected knowledge network through dynamic indexing and linking, enabling semantically related memories to cluster for efficient retrieval.
- **LangMem** [1]: LangChain's open-source memory module uses an LLM to extract information from conversations and update episodic, semantic, and procedural memories.
- **OpenAI** [2]: This method uses a dedicated prompt to inject information into ChatGPT, generating timestamped memories that emulate OpenAI's Memory feature.
- **Zep** (Rasmussen et al., 2025): A system that integrates a time-aware knowledge graph (Graphiti) to synthesize unstructured conversational data with structured metadata, maintaining historical relationships for efficient semantic querying.
- **MemOS** (Li et al., 2025b): An operating-system–inspired system that constructs **plaintext memory** for independently stored external information, **activation memory** for intermediate reasoning states, and **parameter**

[1] https://langchain-ai.github.io/langmem/

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

memory, which refers to knowledge and capabilities encoded in the model's fixed weights, with **MemCube** managing and transforming memory across these types.

- **MIRIX** (Wang and Chen, 2025): A memory system that partitions memory into six types—core, episodic, resource, procedural, semantic, and knowledge vault—managed by a meta-agent with an active retrieval mechanism for efficient access.

- **Full-Context**: Delegating all information to the LLM for answering; under the assumption that the document length does not exceed the context window, this can be regarded as a theoretical upper bound.

We follow the official implementations provided in the Mem0[3] and MemOS[4] repositories (which also include all other baseline methods).

B.3 Evaluation Metrics

- **F1 Score:** The F1 score quantifies the balance between precision and recall, combining them into a single measure:

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{29}$$

with

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}, \quad \text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}. \tag{30}$$

- **BLEU-1:** BLEU-1(Papineni et al., 2002) evaluates the unigram overlap between the generated output and the reference text:

$$\text{BLEU-1} = \text{BP} \cdot \exp\Big(\sum_{n=1}^{1} w_n \log p_n\Big), \tag{31}$$

$$\text{BP} = \begin{cases} 1, & c > r \\ e^{1-r/c}, & c \le r \end{cases}, \tag{32}$$

$$p_n = \frac{\sum_i \sum_k \min(h_{ik}, m_{ik})}{\sum_i \sum_k h_{ik}} \tag{33}$$

where c is the candidate length, r is the reference length, h_{ik} is the count of n-gram i in candidate k, and m_{ik} is the maximum count in references.

- **ROUGE-L:** ROUGE-L(Lin, 2004) captures the longest common subsequence between reference and candidate:

$$\text{ROUGE-L} = \frac{(1+\beta^2) R_l P_l}{R_l + \beta^2 P_l}, \tag{34}$$

$$R_l = \frac{\text{LCS}(X, Y)}{|X|}, \tag{35}$$

$$P_l = \frac{\text{LCS}(X, Y)}{|Y|} \tag{36}$$

where X and Y denote reference and candidate texts, respectively, and LCS stands for Longest Common Subsequence.

- **METEOR:** METEOR (Banerjee and Lavie, 2005) accounts for unigram matches, alignment, and paraphrasing:

$$\text{METEOR} = F_{\text{mean}} \cdot (1 - \text{Penalty}), \tag{37}$$

$$F_{\text{mean}} = \frac{10P \cdot R}{R + 9P}, \tag{38}$$

$$\text{Penalty} = 0.5 \cdot \left(\frac{ch}{m}\right)^3 \tag{39}$$

where P and R are precision and recall, ch is the number of chunks, and m is the total number of matched unigrams.

- **SBERT Similarity:** SBERT(Reimers and Gurevych, 2019) similarity evaluates semantic closeness using sentence embeddings:

$$\text{SBERT\_Sim}(x, y) = \cos(\text{SBERT}(x), \text{SBERT}(y)), \tag{40}$$

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \tag{41}$$

where $\text{SBERT}(x)$ denotes the embedding of sentence x.

- **LLM-as-a-judge score:** Automatic metrics based on lexical overlap suffer from inherent limitations in evaluating factual correctness in conversational settings. For instance, when the ground truth is *"The meeting is scheduled for Monday morning"* but the model outputs *"The meeting is scheduled for Friday morning"*, metrics such as F1 or BLEU-1 may still assign

[3] https://github.com/mem0ai/mem0/tree/main/evaluation

[4] https://github.com/MemTensor/MemOS/tree/main/evaluation

high scores due to shared tokens, despite the critical factual error. This limitation often results in misleading evaluations that fail to reflect semantic and factual validity.

To mitigate this issue, *LLM-as-a-Judge* is commonly used as a complementary evaluation metric (Packer et al., 2023). The judge evaluates responses by jointly considering the question, reference answer, and generated output, assessing factors such as factual accuracy, relevance, and completeness, and producing judgments that better align with human evaluation.

B.4 Memory Details

Follows MIRIX (simulating human-like memory characteristics), we construct six memory modules: **Episodic Memory**, **Procedural Memory**, **Resource Memory**, **Semantic Memory**, **Core Memory**, and **Knowledge Vault Memory**, managed by a **Meta Manager**. Each module serves a specific function and is implemented via its corresponding prompt; prompts are detailed in D.2.

Meta Manager Upon receiving new information, the Meta Manager determines which memory modules should handle it next, issues update instructions to the relevant modules, and prefers to include multiple memory types rather than risk omitting potentially important information.

Episodic Memory Stores event-based information in chronological order, effectively serving as an "interaction log". Each event contains event_type, summary, details, actor, and tree_path.

Procedural Memory Identifies step-by-step instructions, operational guides, workflows, or other procedural knowledge, containing `description` and step-wise `steps`.

Resource Memory Holds user-specific or task-related documents, files, or reference materials. If a document already exists in the resource memory, the existing entry should be updated.

Semantic Memory Stores concepts, definitions, facts, and linguistic elements, serving as an abstract understanding of the world—for example, new software names, new concepts, or entities such as people or locations. Only novel concepts are saved; common knowledge or well-known entities are excluded.

Core Memory Analyzes messages in depth, identifying user behavior patterns, preferences, personal details, and any information potentially useful in future interactions. Even if some information exists in other memory modules, it should be updated in Core Memory.

Knowledge Vault Stores retrievable structured factual data for "quick lookup." Stored content includes discrete data points, credentials, identifiers, contact information, URLs, configuration values, and similar structured information.

B.5 Implementation Details

Parameter	Value
Page size	595 × 842
DPI	96
Margin (x)	10
Margin (y)	10
Font path	`Verdana.ttf`
Font size	9
Line height	10
Font color	#000000
Text alignment	LEFT
Horizontal scale	1.0
First-line indent	0
Left indent	0
Right indent	0
Space before	0
Space after	0
Border width	0
Border padding	0
Page background	#FFFFFF
Paragraph background	#FFFFFF
Auto-crop width	True
Auto-crop last page	True
Newline markup	` `

Table 4: Rendering parameters for text-to-image generation.

Rendering Parameters Table 4 summarizes the rendering parameters used to generate images. All parameters follow the reference configuration provided in the Glyph (Cheng et al., 2025).

All training datasets are rendered using a fixed DPI of 96 as specified above. While the newline-markup option can be replaced with alternative form, we observe that different newline encodings may affect both retrieval performance and downstream reasoning behavior.

Method	Single Hop	Multi-Hop	Open Domain	Temporal	Overall
Comparison with Ours(DVR)					
Mem0	68.97	51.42	**72.92**	56.07	63.31
Mem0^g	69.32	51.77	73.96	59.19	64.29
A-Mem	39.24	26.60	54.17	49.84	40.06
LangMem	70.27	57.09	55.21	56.39	64.03
OpenAI	61.24	58.87	62.50	24.92	53.31
Ours(DVR)-Run1	85.26	58.51	53.12	59.50	72.99
Ours(DVR)-Run2	85.61	59.57	53.12	61.06	73.70
Ours(DVR)-Run3	84.42	60.99	50.00	57.94	72.47
Comparison with Ours(ViHM)					
Zep	80.26	71.99	70.83	82.87	78.70
MemOS	78.12	67.02	63.51	80.06	75.58
MIRIX	83.95	82.27	63.54	86.92	82.99
Ours(ViHM)-Run1	83.23	74.82	66.67	85.67	81.17
Ours(ViHM)-Run2	84.78	78.01	67.71	85.98	82.73
Ours(ViHM)-Run3	82.88	75.89	62.50	86.60	81.10
Full-Context	88.59	78.01	71.88	92.83	86.49

Table 5: **LLM-as-a-Judge score** (%, higher is better) comparison under different graph-based agentic memory systems on LoCoMo. The table reports results from three independent runs of *Ours (DVR)* and *Ours (ViHM)*, showing no substantial performance variation overall.

Training Details During training, we fine-tune the language tower using LoRA adapter (Hu et al., 2022) applied to the q_{proj}, k_{proj}, v_{proj}, and o_{proj} layers, while keeping the vision tower frozen. The LoRA rank is set to 16. On top of the backbone, a single-layer MLP projection head reduces the 4096-dimensional token features to 128 dimensions. This architecture is trained for one epoch on the custom dataset, effectively adapting the base generative model into a token-level embedding model.

C More Experimental Results

C.1 Full Experimental Results with Different Runs on LoCoMo

We conduct multiple runs of *DVR* and *ViHM* on LoCoMo, as shown in Table 5. Across different runs, the results exhibit no significant fluctuation, demonstrating the robustness of the visual agentic memory system. Results reported in the main text are averaged over these runs.

C.2 Experimental Results on LongMemEval

We further evaluate our approach on LongMemEval, a comprehensive benchmark consisting of 500 carefully designed questions for assessing long-term memory in conversational assistants. Each question corresponds to a single ultra-long conversation with an average length of approximately 115k tokens, which is significantly longer than LoCoMo and thus poses a greater challenge for memory agents to effectively compress information. However, the tasks in this benchmark are relatively less complex and the corpus structure is less realistic: target statements are split and inserted into a large amount of distracting content, and the lack of semantic coherence among contexts reduces the difficulty of organizing memory.

On LongMemEval, we additionally evaluate the *Supermemory*[5] baseline, which manages stored memories along multiple dimensions, including tag-based categorization, project-level grouping, and associative recommendation.

Similar to our protocol on LoCoMo, we first evaluate multiple memory frameworks and then follow the best-performing architecture to construct *ViHM*, while also evaluating *DVR*. Although MemOS achieves the best overall performance, it relies on complex mechanisms such as KV cache manipulation and parameter-level memory, which are difficult to realize through visual representations. Therefore, we adopt the second-best frame-

[5] https://github.com/supermemoryai/supermemory

method	content tokens	single-session preference ↑	single-session assistant ↑	temporal reasoning ↑	multi-session ↑	knowledge update ↑	single-session user ↑	overall ↑
MIRIX	-	46.7	60.7	23.3	24.8	61.5	72.9	42.2
ZEP	1.7k	60.0	**73.2**	54.9	44.4	71.8	87.1	61.8
Supermemory	1.5k	86.7	51.8	50.4	57.1	62.8	81.4	60.8
MemOS	1.4K	**93.3**	66.1	**71.4**	69.9	73.1	94.3	**75.2**
Ours(DVR)	**0.9K**	63.3	26.8	58.7	66.9	**79.5**	**98.6**	66.4
Ours(ViHM)	**0.4K**	90.0†	44.6†	68.4†	69.2†	75.6†	90.0†	71.4†
Mem0	1.1K	90.0	46.4	69.2	**70.7**	73.1	88.6	71.6

Table 6: **LLM-as-a-Judge** Performance Comparison across different memory systems on LongMemEval Dataset. † indicates performance that is nearly identical to the Mem0 baseline, showing no significant degradation despite reduced token usage.

work, *mem0*, as the foundation. Experimental results are reported in Table 6.

Experimental results show substantial performance differences among memory systems across different question types on this dataset. *DVR*, which does not employ explicit memory management and instead converts raw conversations into visual representations for retrieval, surpasses several baselines with only **0.9k tokens** consumed. It even achieves the best performance on *knowledge update* and *single-session user* categories, demonstrating strong retrieval capability.

ViHM follows the design of *mem0*, where salient information is first extracted from conversations before being stored. The prompts used for information extraction are detailed in D.3. Given the relatively small number of questions in this dataset, *ViHM* achieves performance nearly identical to *mem0*, while requiring only **0.4k tokens**. This further highlights the potential of combining visual representations with agentic memory systems.

C.3 Question Answering over Ultra-Long Documents

To evaluate our model's retrieval and question answering capabilities under extremely long contexts, we consider a full-length novel, *A Song of Ice and Fire*, containing approximately 1.3M words and about 1.7M textual tokens. Even after visual compression, the document still yields nearly 600K visual tokens, which exceeds the maximum context window of current VLMs (128K).

We select several questions for qualitative QA evaluation, with representative examples shown in Table 7, including model responses and comparisons against the ground truth. Across all three cases, the framework demonstrates its ability to effectively retrieve the most relevant information from ultra-long documents, reducing the context to a scale that is acceptable for VLMs.

For Questions 1 and 2, the model correctly answers almost all key points, with only minor omissions and some non-critical content. For Question 3, only one of the two ground-truth answers is correctly retrieved, while the other exhibits a clear case of hallucinated content through erroneous information composition.

Additionally, in Question 2, the model answers "black claws" as "black paws," which indicates that converting text into visual representations may still lead to information loss due to rendering parameters or limitations in the model's encoding capacity.

C.4 Case Study

Table 8 presents a case study on LoCoMo, comparing two variants of our approach: *DVR* and *ViHM*. Since *ViHM* achieves a significantly higher overall score than *DVR*, we analyze their differences in information processing across four representative categories: *single-hop*, *multi-hop*, *temporal reasoning*, and *open-domain* questions.

The key distinction lies in their memory handling strategies. *DVR* does not perform any additional processing on raw conversational content; instead, it directly renders the original information chunks into visual representations and retrieves them at query time. In contrast, *ViHM* incrementally constructs and updates a hierarchical memory during the conversation. We examine representative questions where the two methods exhibit notable performance differences and closely inspect the specific memories retrieved by each system.

Single-hop Reasoning In this example, *ViHM* stores two original statements ("pottery" and "painting") in core memory and gradually merges them into a unified concept representing Melanie's

Question	Response	GroundTruthComparison
List all the titles, honorifics, and positions held by Daenerys Targaryen.	**First of Her Name**, **Khaleesi of the Great Grass Sea**, **Daenerys Stormborn**, **the Unburnt**, **Mother of Dragons**, sole surviving heir of Aerys II Targaryen, widow of Khal Drogo, **Queen of Meereen**, **Queen of the Andals and the Rhoynar and the First Men**, **Lord of the Seven Kingdoms**, **Protector of the Realm**.	**Matches most key titles**; missed **Breaker of Chains**, order differs slightly.
Describe Jon Snow's direwolf, including its name and distinguishing characteristics.	**Jon Snow's direwolf is named Ghost**. It is a large direwolf with **white fur** and **black paws**. **Ghost is known for its fierce protectiveness and is a loyal companion to Jon Snow.**	**Correctly retrieved name and fur color**; Should be **black claws, but answered black paws**; silent nature not mentioned.
Which characters were directly responsible for the death of Eddard Stark.	****Joffrey Baratheon**** ordered Eddard Stark's murder as part of his scheme to consolidate power and eliminate threats. - ****Lady Olenna Tyrell** (Queen of the Andals and the First Men)** threatened Joffrey with death unless he killed Eddard.	**Correctly mentions Joffrey Baratheon** , but **is completely unrelated to Lady Olenna Tyrell. Furthermore, the description of Olenna is derived from fictional splicing/fabrication.**

Legend: **Green** = Correctly retrieved/matched Ground Truth; **Red** = Severe factual errors or missing; Yellow = extra or non-critical content.

Table 7: Long-Context QA Case Studies on A Song of Ice and Fire. The table shows model responses, ground truth comparison, and color-coded fact correctness for three representative questions.

art personality. This abstraction leads to hallucination. Additionally, the phrase "To catch the eye and make people smile" is deemed unimportant and discarded during memory evolution, resulting in missing information. In contrast, *DVR* achieves the best performance among all methods in this category (85.10), outperforming the sota method, MIRIX (83.95). Since single-hop questions only require recalling directly relevant facts, this result indicates that rendering raw information into images and retrieving them via multi-vector scoring can yield unexpectedly high accuracy.

Multi-hop Reasoning This question requires integrating two dispersed pieces of evidence with no explicit linkage. *DVR* accurately retrieves the relevant statement, "for 4 years, since I moved from my home country," but fails to recover the fact that the home country is Sweden. In contrast, *ViHM* successfully binds "home country" and "Sweden" together through memory evolution, storing them across core memory and knowledge vault memory, which enables correct multi-hop reasoning.

Open-domain Questions The open-domain category reveals a clear limitation of advanced memory designs: overuse of abstraction. Although more sophisticated memory systems achieve higher overall performance, they may degrade on open-domain questions. In the examined case, *ViHM* correctly retains the key concept "policymaking," but, in an attempt to reconcile additional contextual signals such as education and infrastructure across multiple dialogue turns, it incorrectly synthesizes the answer "mechanical engineering." This example illustrates how excessive summarization and forced semantic alignment can introduce erroneous conclusions.

Temporal Reasoning This example highlights another drawback of complex memory evolution: information loss. The original conversation only contains a vague temporal cue, "Got the tattoo a few years ago." During memory evolution, this imprecise time reference is considered low-importance and is either merged or discarded. As a result, *ViHM* fails to preserve the temporal information needed for correct reasoning.

Question	Evidence	Ours (DVR)	Ours (ViHM)
Single-Hop Why did Melanie choose to use colors and patterns in her pottery project?	"I'm obsessed with those, so I made something to catch the eye and make people smile. Painting helps me express my feelings and be creative."	**To catch the eye and make people smile.**	**To express her feelings and creativity. She finds pottery a therapeutic, comforting, and meaningful form of self-expression.**
Multi-Hop Where did Caroline move from 4 years ago?	"I've known these friends for 4 years, since I moved from my home country." "A gift from my grandma in my home country, Sweden."	**Caroline moved from her home country 4 years ago.**	**Her home country Sweden**
Open-Domain What might John's degree be in?	"I'm considering going into policymaking because of my degree and my passion for making a positive impact."	**Public Policy.**	**Mechanical engineering, given his interest in policymaking focused on education and infrastructure improvements in community.**
Temporal When did Gina get her tattoo?	"Got the tattoo a few years ago."	**A few years ago (as mentioned in the 8 February 2023 conversation).**	**There is no specific date recorded for when Gina got her tattoo.**

Table 8: Case studies across different question categories. **Green** = Correct; **Red** = Incorrect

D Prompts

D.1 Prompt for LLM-as-a-Judge Score

Prompt for LLM-as-a-Judge Score

```
Your task is to label an answer to a question as 'CORRECT' or 'WRONG'. You
    will be given the following data:
    (1) a question (posed by one user to another user),
    (2) a 'gold' (ground truth) answer,
    (3) a generated answer
which you will score as CORRECT/WRONG.
The point of the question is to ask about something one user should know about
     the other user based on their prior conversations.
The gold answer will usually be a concise and short answer that includes the
    referenced topic, for example:
Question: Do you remember what I got the last time I went to Hawaii?
Gold answer: A shell necklace
The generated answer might be much longer, but you should be generous with
    your grading - as long as it touches on the same topic as the gold answer,
     it should be counted as CORRECT.
For time related questions, the gold answer will be a specific date, month,
    year, etc. The generated answer might be much longer or use relative time
    references (like "last Tuesday" or "next month"), but you should be
    generous with your grading - as long as it refers to the same date or time
     period as the gold answer, it should be counted as CORRECT. Even if the
    format differs (e.g., "May 7th" vs "7 May"), consider it CORRECT if it's
    the same date.
Now it's time for the real question:
Question: {question}
Gold answer: {gold_answer}
Generated answer: {generated_answer}

First, provide a short (one sentence) explanation of your reasoning, then
    finish with CORRECT or WRONG.
Do NOT include both CORRECT and WRONG in your response, or it will break the
    evaluation script.
Just return the CORRECT or WRONG in a json format with the key as "label".
```

D.2 Prompt MIRIX-Inspired Memory

For readability, only the core parts of the prompt are shown; omitted sections are indicated in parentheses with a brief description.

Prompt for Meta Memory Agent

```
You are the Meta Memory Manager.

[The Meta Manager coordinates the system consisting of itself and six memory
    agents; they collectively manage the same memory set.]

The details of all memory components, and how content should be classified
    into different memories, are summarized below:
[Detailed characteristics of each memory module are omitted for brevity]

DECISION-MAKING FRAMEWORK:

When processing messages, think step by step, evaluating each memory type in
    the following order:

1. Core Memory: Does this reveal something about WHO the user is or HOW they
    prefer to communicate?
   - User identity, preferences, personality traits, relationships
   - Communication style preferences, behavioral patterns
```

```
2. Episodic Memory: Does this describe WHAT happened WHEN?
   - Events, activities, experiences with time context
   - User's interactions and personal timeline
   - Update for almost all user activities and experiences

3. Procedural Memory: Does this explain HOW TO do something?
   - Step-by-step instructions, workflows, processes
   - Sequential procedures or tutorials

4. Resource Memory: Does this involve WHAT files or documents?
   - Shared files, documents, images, reference materials

5. Knowledge Vault: Does this contain static REFERENCE DATA?
   - Contact info, passwords, IDs, addresses, credentials
   - Static facts that don't change often but might be needed later

6. Semantic Memory: Does this mention NEW general knowledge or concepts?
   - New concepts, people, places, organizations not previously known
   - Definitions and abstract knowledge about the world

[Tool invocation workflow is omitted for brevity]
```

Prompt for Episodic Memory Agent

```
You are the Episodic Memory Manager, part of the personal assistant system.

[The system also includes other agents; you do not directly see or interact
    with them, but you share the same memory repository.]

[Information stored by other agents helps you decide whether to update your
    memory; details omitted for brevity.]

Episodic Memory:
     - Definition: Stores time-ordered, event-based information from
    interactions-essentially, the "diary" of user and assistant events.
     - Example: If the user shared a voice message about having dinner with a
    friend, record this event with title "having dinner with xxx", timestamp (
    e.g., "2025-03-05 10:15") and details (e.g., background, discussions
    during the dinner, etc.).
     - Each item in Episodic Memory has the following attributes:
        (a) event_type: Type/category of the episodic event (e.g.,
    user_message, inferred results from user message, system_notification)
        (b) summary: Short textual summary of the event, e.g. (suppose you
    already know that the user's name is John), "John shared a photo of his
    vacation in Paris." Make sure the summary is concise and informative.
        (c) details: Detailed description or text for the event, e.g. (suppose
     you already know that the user's name is John and suppose the user sent a
     photo with caption), "John sent an image showing the Eiffel Tower with
    the caption 'Amazing sunset view from our hotel room'. The photo shows
    John and his partner enjoying the evening light. He mentioned in the
    accompanying voice message that this is their first trip to Europe
    together." The key idea is to record as many details as possible here.
        (d) actor: The actor who generated the event (user or assistant)
        (e) tree_path: Required hierarchical categorization path for
    organizing events (e.g., ["personal", "travel", "vacation"] or ["work", "
    meetings", "team"]). Use this to create logical groupings and enable
    better organization of memories.

[Definitions of functions, tools, the calling pipeline and some specific
    important notes, are omitted for brevity]
```

Prompt for Procedural Memory Agent

```
You are the Procedural Memory Manager, part of the personal assistant system.

[The system also includes other agents; you do not directly see or interact
    with them, but you share the same memory repository.]

[Information stored by other agents helps you decide whether to update your
    memory; details omitted for brevity.]

Procedural Memory:
Definition: Contains how-to guides, step-by-step instructions, or processes
    the assistant or user might follow.
Example: "How to reset the router."
Each entry in Procedural Memory has:
   (a) entry_type (e.g., 'workflow', 'guide', 'script')
   (b) description (short descriptive text)
   (c) steps (the procedure in a structured or JSON format)
   (d) tree_path: Required hierarchical categorization path for organizing
    procedures (e.g., ["technology", "networking", "troubleshooting"] for
    router reset guides, or ["cooking", "baking", "desserts"] for recipe
    instructions). Use this to create logical groupings and enable better
    organization of procedural knowledge.

[Definitions of functions, tools, the calling pipeline and some specific
    important notes, are omitted for brevity]
```

Prompt for Resource Memory Agent

```
You are the Resource Memory Manager, part of the personal assistant system.

[The system also includes other agents; you do not directly see or interact
    with them, but you share the same memory repository.]

[Information stored by other agents helps you decide whether to update your
    memory; details omitted for brevity.]

Resource Memory:
Holds documents, files, or reference materials for the user's personal or work
     tasks.
Example: "VacationPlans.docx" with the creation date, tags like "travel," and
    partial or full file content.
Each Resource Memory item has:
   (a) title: Short name/title of the resource
   (b) summary: A brief description or summary of the resource, including
    which project it is from, which context it came from, what the content is
    about, what does the code do, or what is the doc about, etc.
   (c) resource_type: File type or format (e.g. 'doc', 'markdown', 'pdf_text',
     'image', 'voice_transcript')
   (d) content: Full or partial text content of the resource
   (e) tree_path: Required hierarchical categorization path for organizing
    resources (e.g., ["work", "projects", "ai-assistant"] for project
    documents, or ["personal", "finance", "taxes"] for financial documents).
    Use this to create logical groupings and enable better organization of
    documents and files.

[Definitions of functions, tools, the calling pipeline and some specific
    important notes, are omitted for brevity]
```

Prompt for Semantic Memory Agent

```
You are the Semantic Memory Manager, part of the personal assistant system.

[The system also includes other agents; you do not directly see or interact
    with them, but you share the same memory repository.]

[Information stored by other agents helps you decide whether to update your
    memory; details omitted for brevity.]
 Semantic Memory (your primary domain):
   - Definition: Semantic Memory holds general knowledge, concepts,
    definitions, facts, and language elements. It is the storehouse of
    abstract understanding about the world, such as a new software name, a new
     concept, or an object (e.g., a person, a place).
   - ONLY save new concepts that are NEW to you. DO NOT save the commensense
    knowledge or person such as "VS Code", "Google Chrome", "ChatGPT", "Albert
     Einstein", "numpy", "scipy" (because you know them) unless they mean
    something different to the user.
   - You should save NEW concepts, NEW knowledges, and NEW persons.
   - Each entry in Semantic Memory should include:
    (a) Name: The name of the new concept or an object. For instance, "
    MemoryLLM", or "Jane" (These are just examples).
    (b) Summary: A concise explanation or summary of the concept or the object
    . For example, "MemoryLLM is a type of memory architecture designed for
    large language models." or "Jane is a friend who is a computer scientist."
    (c) Details: An extended description that may include context, examples,
    or deeper insights. For example, elaborate on the details of "MemoryLLM" (
    what, how, etc.) or "Jane" (What kind of person is her).
    (d) Source: A reference to where this general knowledge originates (e.g.,
    user message, voice recording, image description).
    (e) Tree Path: A required hierarchical categorization path as an array of
    strings for organizing concepts (e.g., ['favorites', 'pets', 'dog'] or ['
    work', 'projects', 'ai-research']). This helps structure and navigate the
    semantic memory efficiently.

[Definitions of functions, tools, the calling pipeline and some specific
    important notes, are omitted for brevity]
```

Prompt for Core Memory Agent

```
You are the Core Memory Manager, part of the personal assistant system.
[The system also includes other agents; you do not directly see or interact
    with them, but you share the same memory repository.]
[Information stored by other agents helps you decide whether to update your
    memory; details omitted for brevity.]

Core Memory:
Contains fundamental information about the user, such as name, personality,
    and simple details that help communication.
Example: Anything related to the user can go in the `human` block, including
    personal info and preferences. Examples: "Is a software engineer", "Loves
    to play Cyberpunk", "Has publications: 1. ... 2. ...", etc.
It can be as detailed as possible.
If a block exceeds 90% full, call `core_memory_rewrite` to rewrite it. When
    rewriting, keep the most important information and reduce the block to
    roughly half its current size (~45% full). Include the block label in
    every function call.

The memory is organized in multiple blocks. Each block has line indicators
    such as "Line 1:", "Line 2:". (for reference only; not part of content).
[Definitions of functions, tools, the calling pipeline and some specific
    important notes, are omitted for brevity]
```

Prompt for Knowledge Vault Memory Agent

```
You are the Knowledge Vault Memory Manager, part of the personal assistant
    system.

[The system also includes other agents; you do not directly see or interact
    with them, but you share the same memory repository.]

[Information stored by other agents helps you decide whether to update your
    memory; details omitted for brevity.]

Knowledge Vault:
Definition: A repository for structured, retrievable factual data that serves
    as "quick lookup" information.
**STORE HERE:** Discrete data points, credentials, identifiers, contact info,
    URLs, configuration values.
Examples: Phone numbers ("123-456-7890"), API keys, passwords, email addresses
    , specific URLs, account usernames, addresses, database connection strings
    .
Each Knowledge Vault item has:
   (a) entry_type (e.g., 'credential', 'bookmark', 'api_key')
   (b) source (e.g., 'github', 'google', 'user_provided')
   (c) sensitivity (e.g., 'low', 'medium', 'high')
   (d) secret_value (the actual data to store securely)

[Definitions of functions, tools, the calling pipeline and some specific
    important notes, are omitted for brevity]
```

D.3 Prompt for Mem0-Inspired Memory

Prompt for Extracting Information from Raw Content

```
Generate personal memories that follow these guidelines:

1. Each memory should be self-contained with complete context, including:
   - The person's name, do not use "user" while creating memories
   - Personal details (career aspirations, hobbies, life circumstances)
   - Emotional states and reactions
   - Ongoing journeys or future plans
   - Specific dates when events occurred

2. Include meaningful personal narratives focusing on:
   - Identity and self-acceptance journeys
   - Family planning and parenting
   - Creative outlets and hobbies
   - Mental health and self-care activities
   - Career aspirations and education goals
   - Important life events and milestones

3. Make each memory rich with specific details rather than general statements
   - Include timeframes (exact dates when possible)
   - Name specific activities (e.g., "charity race for mental health" rather
    than just "exercise")
   - Include emotional context and personal growth elements

4. Extract memories only from user messages, not incorporating assistant
    responses

5. Format each memory as a paragraph with a clear narrative structure that
    captures the person's experience, challenges, and aspirations
```