OMNI-SIMPLEMEM: AUTORESEARCH-GUIDED DISCOVERY OF LIFELONG MULTIMODAL AGENT MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

AI agents increasingly operate over extended time horizons, yet their ability to retain, organize, and recall multimodal experiences remains a critical bottleneck. Building effective lifelong memory requires navigating a vast design space spanning architecture, retrieval strategies, prompt engineering, and data pipelines; this space is too large and interconnected for manual exploration or traditional AutoML to explore effectively. We deploy an autonomous research pipeline to discover OMNI-SIMPLEMEM, a unified multimodal memory framework for lifelong AI agents. Starting from a naïve baseline (F1 = 0.117 on LoCoMo), the pipeline autonomously executes $\sim$50 experiments across two benchmarks, diagnosing failure modes, proposing architectural modifications, and repairing data pipeline bugs, all without human intervention in the inner loop. The resulting system achieves state-of-the-art (SOTA) on both benchmarks, improving F1 by +411% on LoCoMo and +214% on Mem-Gallery relative to the initial configurations. Critically, the most impactful discoveries are not hyperparameter adjustments: bug fixes (+175%), architectural changes (+44%), and prompt engineering (+188% on specific categories) each individually exceed the cumulative contribution of all hyperparameter tuning, demonstrating capabilities fundamentally beyond the reach of traditional AutoML. We provide a taxonomy of six discovery types and identify four properties that make multimodal memory particularly suited for autoresearch, offering guidance for applying autonomous research pipelines to other AI system domains.

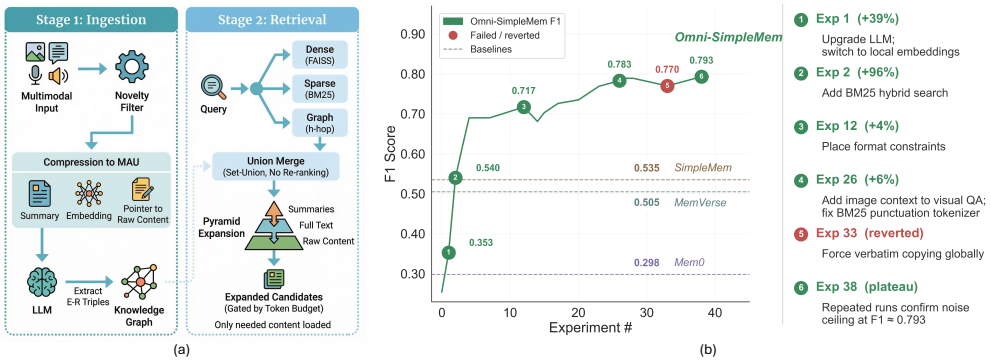


Figure 1: **Overview of the Discovery Process of OMNI-SIMPLEMEM. (a)** Discovered architecture: multimodal inputs are filtered for novelty, compressed into MAUs, and retrieved via hybrid dense-sparse-graph search with pyramid expansion. **(b)** Autonomous optimization trajectory on Mem-Gallery: 39 experiments improve F1 from 0.254 to 0.793 (+214%).

1 INTRODUCTION

Large language model agents increasingly combine tool use, multi-step reasoning, and cross-modal comprehension (Yao et al., 2023; Yan et al., 2025), yet retaining and recalling their accumulated experiences remains a bottleneck (Zhang et al., 2024; Xu et al., 2025b). Lifelong multimodal memory requires jointly designing storage, retrieval, prompts, and data pipelines. Raw-input retrieval (Lewis

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

et al., 2020; Borgeaud et al., 2022) faces storage bloat and retrieval noise, while structured memory systems (Packer et al., 2023; Park et al., 2023) typically focus on text. Both depend on manual research cycles that can miss interactions between tightly coupled components. Traditional AutoML (Hutter et al., 2019a) searches predefined spaces but does not directly address code comprehension, data pipeline repair, and cross-component diagnosis.

Motivated by autonomous scientific discovery (Lu et al., 2024; Romera-Paredes et al., 2024; Panfilov et al., 2026), we deploy AUTORESEARCHCLAW (Liu et al., 2026b), a 23-stage research pipeline, to discover OMNI-SIMPLEMEM, a unified multimodal memory framework. Across $\sim$50 experiments on two benchmarks, the pipeline proposes and implements architectural changes, repairs bugs, and validates improvements without human intervention in the inner loop. It improves F1 from 0.117 to 0.598 (+411%) on LoCoMo and from 0.254 to 0.797 (+214%) on Mem-Gallery, achieving state-of-the-art results on both. The largest discoveries extend beyond hyperparameter tuning: bug fixes (+175%), architectural changes (+44%), and prompt engineering (+188% on specific categories).

The resulting system combines **selective ingestion**, which filters redundant signals; **unified representation**, which separates searchable Multimodal Atomic Unit (MAU) metadata from raw content; and **progressive retrieval**, which expands summaries into details and raw evidence under token budgets. Retrieval combines dense vectors and sparse keywords through set-union merging. Multimodal memory supports this search through four properties: scalar evaluation metrics, modular components, fast experiments (1–2 hours), and version-controlled changes that allow failed experiments to be reverted.

Our contributions are the discovered multimodal memory framework, its evaluation on two benchmarks, and a taxonomy of six autonomous discovery types. We document optimization trajectories and analyze convergence, failures, and recovery, providing guidance for applying autonomous research pipelines to other multi-component systems.

2 RELATED WORK

Autonomous Scientific Discovery. The vision of AI-driven research has advanced rapidly. *The AI Scientist* (Lu et al., 2024) demonstrated end-to-end paper generation at $\sim$\$15 per paper across three ML domains, with its successor *AI Scientist v2* (Yamada et al., 2025) eliminating human-authored templates through agentic tree search. *FunSearch* (Romera-Paredes et al., 2024) pairs LLM creativity with programmatic evaluation to discover novel mathematical constructions. AUTORESEARCHCLAW (Liu et al., 2026b) introduces a 23-stage autonomous research pipeline with multi-agent debate and self-healing execution. *AI-Researcher* (Tang et al., 2025) introduces collaborative multi-agent frameworks with structured iterative refinement, and *Bilevel Autoresearch* (Qu & Lu, 2026) meta-optimizes the search strategy itself. A comprehensive survey by Tie et al. (2026) maps the evolution from foundational modules (2022–2023) through closed-loop systems (2024) to scalable human-AI collaboration (2025+). We apply the autoresearch paradigm to multi-component AI system optimization, where the challenge shifts from discovering isolated artifacts to diagnosing and improving interactions across tightly coupled modules.

Multimodal Memory Systems. Memory-augmented LLM agents have evolved from text-only systems, including MemGPT (Packer et al., 2023) with OS-inspired memory hierarchies, Generative Agents (Park et al., 2023) with recency-importance-relevance scoring, SimpleMem (Liu et al., 2026a) with efficient lifelong memory, and A-Mem (Xu et al., 2025a) with LLM-directed reorganization, to multimodal architectures. MemVerse (Liu et al., 2025) combines episodic-semantic memory with multimodal knowledge graphs but requires three LLM calls per ingested item. Mem0 (Chhikara et al., 2025) offers dynamic fact extraction with optional graph memory. VisRAG (Yu et al., 2025) indexes visual pages directly, avoiding text extraction losses. Claude-Mem (Anthropic, 2024) provides commercial embedding-based dialogue memory. These systems all require extensive manual tuning of retrieval strategies, ingestion pipelines, and prompt configurations, which is precisely the kind of optimization that autonomous research pipelines can accelerate.

Automated Machine Learning. Neural Architecture Search (Hutter et al., 2019b; Zoph & Le, 2017; Liu et al., 2019) automates model design but operates primarily on well-defined architectural search spaces with differentiable or reinforcement-learning-based objectives. Hyperparameter optimization methods (Bergstra et al., 2011; Falkner et al., 2018) efficiently navigate continuous and categorical

spaces, while systems like Auto-sklearn 2.0 (Feurer et al., 2022) automate full ML pipelines including preprocessing and model selection via meta-learning. More recently, LLM-based agents have been applied to ML tasks: MLAgentBench (Huang et al., 2024) benchmarks LLM agents on ML research tasks involving code modification, demonstrating the potential of language-guided optimization. Our setting differs: the "search space" includes not only hyperparameters and architectural choices, but also *prompt engineering*, *data pipeline bug detection and repair*, *evaluation format alignment*, and *cross-component interaction diagnosis*, all of which require natural language understanding and code modification capabilities beyond traditional AutoML.

3 AUTORESEARCH-GUIDED DISCOVERY OF OMNI-SIMPLEMEM

In this section, we describe the autonomous optimization process and the system it produces. We first overview the pipeline (§3.1), then present the discovered OMNI-SIMPLEMEM architecture (§3.2), followed by the benchmark-specific optimization strategy (§3.3).

3.1 PIPELINE OVERVIEW

As discussed in Section 1, the design space of multimodal memory systems is too large and interconnected for manual exploration to cover effectively. To address this, we deploy AUTORESEARCHCLAW (Liu et al., 2026b), a 23-stage autonomous research pipeline, to systematically optimize OMNI-SIMPLEMEM. The pipeline receives three inputs: (1) the SimpleMem (Liu et al., 2026a) codebase, a unimodal text-only lifelong memory framework, as a starting point, (2) two benchmark evaluation harnesses with quantitative metrics (F1), and (3) API access to LLM providers. It then enters an iterative loop: at each step, the pipeline analyzes prior results, generates a hypothesis for improvement, implements the change in code, evaluates on a benchmark, and decides whether to **proceed** (metric improved by $\geq 0.5\%$), **iterate** (ambiguous result; refine the current hypothesis), or **pivot** (two consecutive degradations; revert and try a new direction). Of $\sim$50 total experiments, most resulted in a proceed decision, with the remainder split between iterate and pivot. The full pipeline phases (scoping, literature discovery, multi-agent debate, experiment design, sandboxed execution, analysis, documentation, and finalization) are described in Liu et al. (2026b); here we focus on the discoveries it makes and the mechanisms that enable them.

3.2 THE AUTO-RESEARCH GUIDED ARCHITECTURE

The pipeline takes SimpleMem (Liu et al., 2026a), a unimodal text-only agent memory framework as its starting point. We provide AUTORESEARCHCLAW with the SimpleMem codebase and instruct it to extend the system from text-only memory to full multimodal support, autonomously designing the necessary architectural components for ingesting, storing, and retrieving heterogeneous signals (text, images, audio, video). Through iterative experimentation, the pipeline converges to an architecture organized around three principles: selective ingestion, progressive retrieval, and structured knowledge (Figure 2).

3.2.1 SELECTIVE INGESTION

The first principle is **selective ingestion**: the system first filters redundant inputs, then encapsulates the retained signals into a unified multimodal representation.

Novelty-Based Filtering. Before any data enters the memory store, lightweight perceptual encoders assess the novelty of incoming information and discard redundant content. For vision, CLIP embeddings are compared across consecutive frames to detect scene changes; for audio, VAD speech probability gates retention to reject silence; for text, Jaccard overlap with recent summaries filters near-duplicates. This filtering significantly reduces storage requirements without losing semantic content.

Multimodal Atomic Units (MAUs). Signals that pass the novelty filter are encapsulated as **(MAUs)**, $\mathcal{M} = \langle s, \mathbf{e}, p, \tau, m, \boldsymbol{\ell} \rangle$, which decouple compact searchable metadata from heavy raw content. Here s is a text summary, $\mathbf{e} \in \mathbb{R}^d$ is its embedding, p points to raw content in cold storage, τ is the timestamp, m is modality, and $\boldsymbol{\ell}$ stores structural links to other MAUs. This yields a two-tier design: *hot storage* $\mathcal{H} = \{\mathcal{M}_i\}_{i=1}^{N}$ keeps summaries, embeddings, and temporal/graph metadata for fast

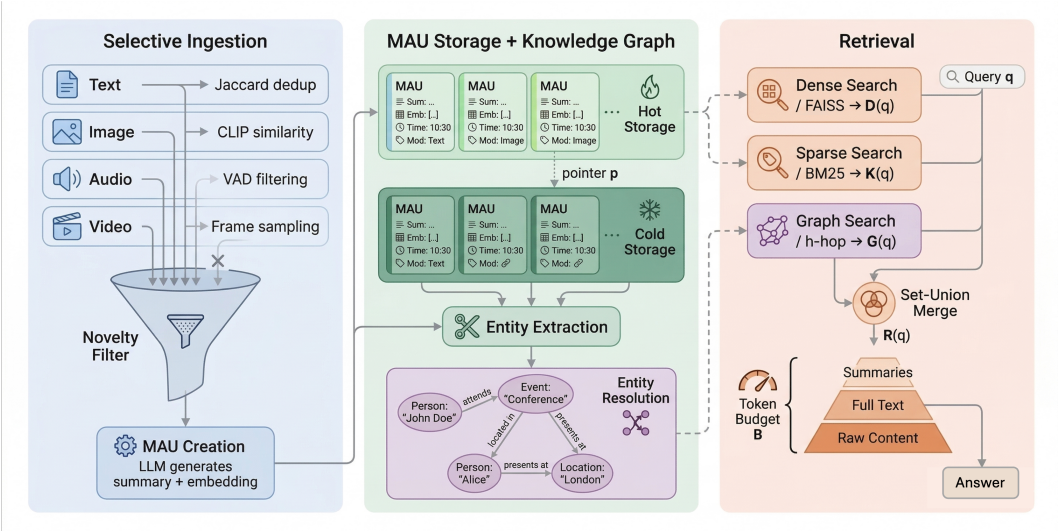


Figure 2: OMNI-SIMPLEMEM architecture overview. **Left:** Selective ingestion filters multimodal inputs (text, image, audio, video) via Novelty-Based Filtering and creates MAUs with LLM-generated summaries and embeddings. **Center:** Different parts of MAUs are stored in hot storage (summaries, embeddings, metadata) and cold storage (raw content), with entity extraction building a Knowledge Graph with entities and relations. **Right:** Retrieval combines dense (FAISS), sparse (BM25), and knowledge graph search via set-union merging, then progressively expands results through a pyramid mechanism (summaries $\rightarrow$ full text $\rightarrow$ raw content).

retrieval, while *cold storage* keeps large assets (images, audio, video) and is accessed lazily through p.

3.2.2 PROGRESSIVE RETRIEVAL WITH HYBRID SEARCH

Once memories are ingested and stored as MAUs, the next challenge is how to retrieve them efficiently at query time. The second principle is **progressive retrieval**: rather than loading all retrieved content into the LLM context at once, OMNI-SIMPLEMEM expands information in stages under explicit token budgets.

Hybrid Dense-Sparse Search. Given a user query q with embedding $\mathbf{e}_q \in \mathbb{R}^d$, dense retrieval via FAISS (Johnson et al., 2021), a library for efficient similarity search over high-dimensional vectors, yields semantically similar candidates $\mathcal{D}(q)$ by inner-product search over L2-normalized MAU embeddings, which is equivalent to ranking by cosine similarity:

$$\mathcal{D}(q) \;=\; \underset{\mathcal{M}_i \in \mathcal{H}}{\text{top-}k}\; \text{sim}(q, \mathcal{M}_i), \qquad \text{sim}(q, \mathcal{M}_i) \;=\; \mathbf{e}_q^\top \mathbf{e}_i \;=\; \cos(\mathbf{e}_q, \mathbf{e}_i). \tag{1}$$

In parallel, sparse retrieval scores the MAU summaries s_i with BM25 (Robertson & Zaragoza, 2009), yielding candidates $\mathcal{S}(q)$:

$$\mathcal{S}(q) \;=\; \underset{\mathcal{M}_i \in \mathcal{H}}{\text{top-}k}\; \text{BM25}(q, s_i), \tag{2}$$

A key discovery of the autonomous pipeline is set-union merging: empirically, score-based re-ranking (the standard approach) disrupts semantic ordering and degrades performance. Instead, each retriever keeps its own internal ordering and the lists are concatenated in turn, so that dense results always occupy the head of the candidate set and only previously unseen items are appended:

$$\mathcal{R}(q) \;=\; \mathcal{D}(q) \;\cup\; \big(\mathcal{S}(q) \setminus \mathcal{D}(q)\big) \tag{3}$$

Knowledge Graph-Augmented Retrieval While hybrid search handles queries that can be answered from individual MAUs, many real-world queries require reasoning over multiple connected facts (*e.g.*, "What gift did I give to the person I met at the conference in March?"). Therefore a complement is **structured knowledge**: OMNI-SIMPLEMEM maintains a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that captures entities and relationships across all MAUs.

During MAU creation, an LLM extracts entities and directed relations from each summary, producing entity-relation triples. Each entity carries a type label from 7 categories (Person, Location, Event, Concept, Time, Organization, Object) and is linked back to its MAU. As new MAUs are ingested, the same real-world entity may appear under different surface forms (*e.g.*, "Dr. Smith" vs. "John Smith"). To prevent node fragmentation, entity resolution merges any pair of entities (u, v) whose hybrid similarity exceeds a merge threshold:

$$\text{sim}(u, v) \;=\; \alpha \cdot \cos(\mathbf{e}_u, \mathbf{e}_v) \;+\; (1 - \alpha) \cdot \text{jw}(n_u, n_v) \tag{4}$$

where the $\alpha \in [0, 1]$ balances the weight of two parts: (1) cosine similarity over embeddings e_u, e_v , and (2) Jaro-Winkler string similarity $\text{jw}(\cdot, \cdot)$ over string names n_u, n_v.

At query time, the system identifies entities $\mathcal{V}_q \subset \mathcal{V}$ in the query and performs neighborhood expansion within h hops. Each reached entity is scored with relevance $r_{\mathcal{G}}(v)$, where $d(v, \mathcal{V}_q)$ is the shortest path distance to any entity. MAUs linked to high-scoring graph entities then form the candidate list:

$$\mathcal{G}(q) \;=\; \big\{\, \mathcal{M}_i \in \mathcal{H} \;\big|\; \mathcal{M}_i \in v \;\text{ and }\; r_{\mathcal{G}}(v) \geq \gamma \,\big\}, \tag{5}$$

v denotes the MAUs from which entity v was extracted, and γ is the threshold. KG-Augmented Retrieval provides relationally connected evidence for answer generation.

3.2.3 Pyramid Retrieval.

The retrieval stages above jointly produce the candidate set $\mathcal{R}(q)$ of Eq. (3), each candidate scored once by the dense similarity $\text{sim}(q, \mathcal{M}_i)$ of Eq. (1) during retrieval. The pyramid mechanism then progressively expands the *content* of these candidates across three levels, reusing this score to gate each transition:

▷ **Level 1** returns text summaries ($\sim$10 tokens each) for the top-k candidates by similarity;

▷ **Level 2** loads all used text for candidates whose similarity exceeds a threshold θ (default value 0.85);

▷ **Level 3** loads raw content from Cold Storage under an explicit token budget B (default value 2,000), expanding items in reversed similarity-per-token order.

All transitions are governed by deterministic rules rather than LLM judgment, avoiding additional latency and token cost, while ensuring the stability of processing.

3.3 Benchmark-Specific Optimization

Having described the multimodal architecture that the pipeline discovers from SimpleMem, we now turn to how it optimizes this architecture for each target benchmark. The pipeline employs a two-phase strategy: rapid iteration on a small training subset, followed by evaluation on the held-out test set.

Development subset for fast iteration. For each benchmark, the pipeline selects a small representative subset for rapid experimentation during the optimization loop. On LoCoMo, a small subset of conversations is used for iterative development, enabling each experiment to complete in under 2 hours. On Mem-Gallery, a small subset of datasets is used, with each experiment completing in minutes. This design enables the pipeline to explore dozens of hypotheses within days. After the optimization trajectory converges, the final configuration is evaluated on the complete benchmark to ensure generalization and to maintain consistency with the evaluation protocols used by prior memory systems.

Iterative diagnosis and repair. During each optimization cycle, the pipeline autonomously diagnoses and repairs failures at two levels. At the *execution level*, when an experiment fails or produces unexpected outputs, a self-healing module classifies the error (API error, dependency error, runtime exception, output format mismatch) and generates a targeted fix. For example, when the embedding service returned 403 errors due to an expired API key, the module detected the authentication failure pattern and switched to a local sentence-transformer backend without manual intervention. At the *semantic level*, when experiments succeed but produce unexpectedly poor metrics, the pipeline performs deeper analysis.

4 EXPERIMENTS

We evaluate OMNI-SIMPLEMEM along two dimensions: (1) the autonomous optimization process, specifically whether the pipeline discovers meaningful improvements across diverse benchmarks, and (2) final system quality, examining whether the discovered architecture achieves SOTA results and whether individual components contribute meaningfully.

4.1 EXPERIMENTAL SETUP

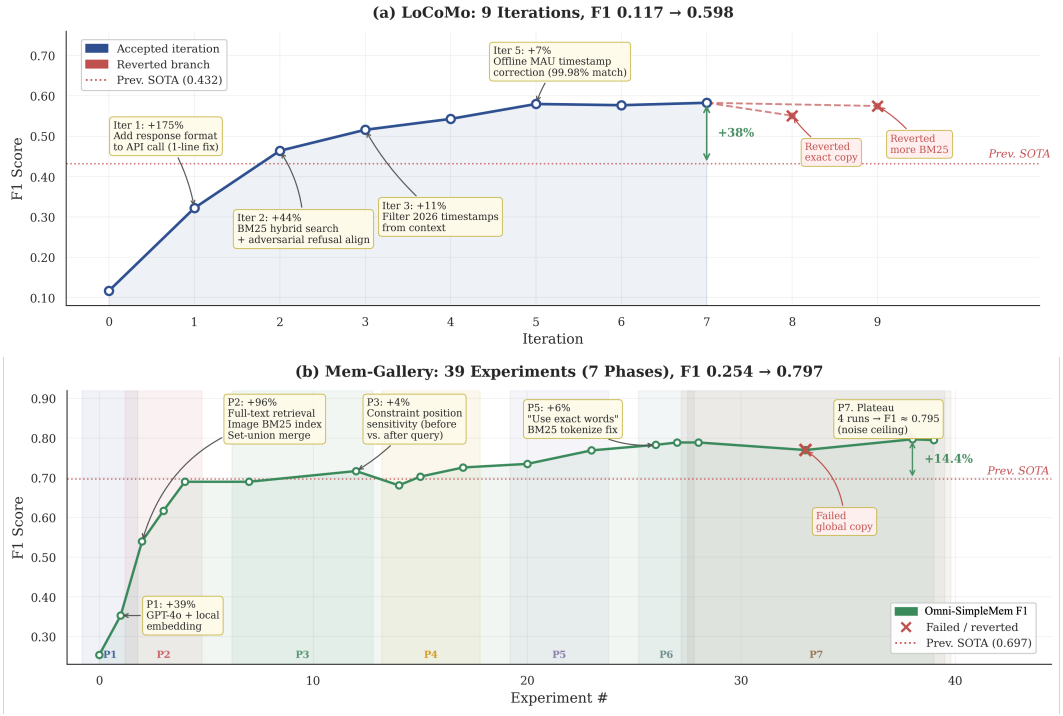


Figure 3: Optimization trajectories on LoCoMo (top, 9 iterations) and Mem-Gallery (bottom, 39 experiments across 7 phases). Solid lines denote accepted iterations; failed/reverted experiments are marked with $\times$. The dashed red line indicates the previous SOTA. Key discoveries at each stage are annotated with descriptions.

Benchmarks. We evaluate on two benchmarks spanning complementary types of memory-dependent reasoning: **LoCoMo** (Maharana et al., 2024) (1,986 QA pairs across multi-session dialogues, token-level F1) and **Mem-Gallery** (Bei et al., 2026) (1,711 QA pairs from 240 multimodal dialogues with 1,003 grounded images, F1). Detailed benchmark descriptions are provided in Appendix B.

Baselines. We compare against six memory systems representing diverse design philosophies: **MemVerse** (Liu et al., 2025) (hierarchical episodic-semantic memory with multimodal knowledge graph), **Mem0** (Chhikara et al., 2025) (dynamic fact extraction with optional graph memory), **Claude-Mem** (Anthropic, 2024) (commercial embedding-based dialogue memory), **A-MEM** (Xu et al., 2025a) (LLM-directed memory reorganization), **MemGPT** (Packer et al., 2023) (OS-inspired memory hierarchies), and **SimpleMem** (Liu et al., 2026a) (efficient lifelong memory with atomization and adaptive pruning). All are evaluated under identical splits and protocols.

Implementation. Dense retrieval uses FAISS with all-MiniLM-L6-v2 embeddings (384d); sparse retrieval uses BM25; visual novelty filtering uses frozen CLIP ViT-B/32. Knowledge graph extraction uses GPT-4o in JSON mode. Default configuration: top-k=20, θ=0.4, B=6,000 tokens. Full per-benchmark configuration details are provided in Appendix D.

Table 1: Experimental Results across five backbone LLMs on LoCoMo (left) and Mem-Gallery (right). LoCoMo columns: MH = multi-hop, SH = single-hop, Tmp = temporal, Open = open-domain, Adv = adversarial, All = overall F1. Mem-Gallery columns: F1, EM = exact match, B/B-1/B-2 = BLEU/BLEU-1/BLEU-2. The second best performance is marked underlined; the best is **bold**.

		LoCoMo						Mem-Gallery				
Backbone	**Method**	**MH**	**SH**	**Tmp**	**Open**	**Adv**	**All**	**F1**	**EM**	**B**	**B-1**	**B-2**
GPT-4o	MemVerse	0.260	0.157	0.196	0.192	**0.944**	0.365	0.505	0.330	0.270	0.440	0.355
	Mem0	0.309	0.156	0.217	0.295	0.857	0.397	0.298	0.192	0.182	0.268	0.224
	Claude-Mem	0.294	0.153	0.167	0.243	0.915	0.383	0.210	0.148	0.148	0.194	0.170
	A-MEM	0.295	0.174	0.200	0.266	0.898	0.394	0.370	0.252	0.240	0.332	0.285
	MemGPT	0.305	0.188	0.246	0.305	0.843	0.404	0.435	0.298	0.275	0.390	0.335
	SimpleMem	0.318	0.195	0.235	0.308	0.802	0.432	0.535	0.348	0.310	0.468	0.390
	OMNI-SIMPLEMEM	**0.556**	**0.365**	**0.255**	**0.641**	0.835	**0.598**	**0.797**	**0.449**	**0.366**	**0.627**	**0.505**
GPT-4o-mini	MemVerse	0.147	0.074	0.106	0.093	0.747	0.290	0.450	0.295	0.248	0.395	0.330
	Mem0	0.285	0.112	0.179	0.297	0.761	0.364	0.291	0.188	0.185	0.265	0.223
	Claude-Mem	0.245	0.102	0.122	0.215	0.845	0.338	0.272	0.175	0.172	0.245	0.210
	A-MEM	0.278	0.091	0.163	0.260	0.823	0.357	0.330	0.222	0.205	0.298	0.252
	MemGPT	0.283	0.113	**0.182**	0.289	0.776	0.364	0.398	0.262	0.242	0.355	0.298
	SimpleMem	0.300	0.128	0.178	0.312	**0.891**	0.404	0.498	0.318	0.290	0.435	0.368
	OMNI-SIMPLEMEM	**0.544**	**0.196**	0.177	**0.588**	0.779	**0.519**	**0.749**	**0.403**	**0.334**	**0.583**	**0.465**
GPT-4.1-nano	MemVerse	0.146	0.061	0.169	0.115	0.711	0.256	0.470	0.308	0.255	0.410	0.340
	Mem0	0.290	0.134	0.194	0.277	0.537	0.310	0.268	0.176	0.156	0.238	0.199
	Claude-Mem	0.087	0.029	0.119	0.047	0.705	0.246	0.303	0.194	0.172	0.268	0.223
	A-MEM	0.045	0.016	0.142	0.050	**0.747**	0.216	0.365	0.242	0.225	0.325	0.275
	MemGPT	0.287	0.130	0.234	0.279	0.556	0.316	0.360	0.238	0.218	0.318	0.268
	SimpleMem	0.298	0.145	0.210	0.285	0.648	0.342	0.518	0.338	0.300	0.452	0.380
	OMNI-SIMPLEMEM	**0.477**	**0.216**	**0.244**	**0.583**	0.722	**0.492**	**0.780**	**0.430**	**0.353**	**0.610**	**0.488**
GPT-5.1	MemVerse	0.287	0.173	0.277	0.297	0.780	0.383	0.478	0.312	0.262	0.418	0.345
	Mem0	0.292	0.160	0.261	0.298	0.819	0.390	0.270	0.175	0.157	0.240	0.200
	Claude-Mem	0.289	0.171	0.264	0.292	0.814	0.388	0.305	0.203	0.188	0.279	0.230
	A-MEM	0.287	0.164	0.246	0.284	**0.826**	0.385	0.408	0.268	0.242	0.365	0.302
	MemGPT	0.288	0.165	0.249	0.294	0.806	0.385	0.425	0.275	0.250	0.378	0.315
	SimpleMem	0.305	0.178	0.272	0.305	0.807	0.418	0.538	0.350	0.312	0.470	0.395
	OMNI-SIMPLEMEM	**0.598**	**0.367**	**0.307**	**0.676**	0.747	**0.613**	**0.810**	**0.460**	**0.374**	**0.639**	**0.515**
GPT-5-nano	MemVerse	0.208	0.203	0.168	0.252	0.741	0.366	0.478	0.315	0.262	0.420	0.345
	Mem0	0.264	0.143	0.237	0.270	0.737	0.352	0.283	0.176	0.165	0.250	0.210
	Claude-Mem	0.091	0.063	0.092	0.088	0.736	0.275	0.350	0.249	0.217	0.315	0.264
	A-MEM	0.260	0.149	0.223	0.257	0.745	0.348	0.505	0.332	0.290	0.445	0.368
	MemGPT	0.267	0.151	0.226	0.271	0.744	0.355	0.388	0.255	0.230	0.345	0.288
	SimpleMem	0.278	0.200	0.245	0.282	**0.824**	0.388	0.522	0.340	0.302	0.458	0.385
	OMNI-SIMPLEMEM	**0.357**	**0.371**	**0.253**	**0.561**	0.719	**0.522**	**0.787**	**0.437**	**0.357**	**0.617**	**0.494**

4.2 OPTIMIZATION TRAJECTORIES

Figure 3 visualizes the optimization trajectories. The pipeline completed $\sim$50 experiments across both benchmarks in $\sim$72 hours of wall-clock time, a coverage that would require approximately 4 weeks for a human researcher at $\sim$3 experiments per day. We highlight the most impactful discoveries; detailed per-iteration tables are provided in Appendix G.

LoCoMo (9 iterations, F1: 0.117→0.598). The pipeline executed 9 successful iterations over 48 hours, with 2 additional experiments automatically reverted (full trajectory in Appendix Table 7). The most impactful discovery (Iter 1, +175%) was identifying that the API call lacked a `response_format` parameter, a one-line bug causing 9× verbosity that destroyed F1 precision. In Iter 5, the pipeline discovered that all 4,277 MAU timestamps had been corrupted to the ingestion date and autonomously generated a keyword-matching script that corrected 99.98% of them without re-ingestion. The pipeline also discovered that set-union merging of FAISS and BM25 results (Iter 2) significantly outperforms score-based fusion, a finding confirmed by ablation (Section 4.4.1).

Mem-Gallery (39 experiments, F1: 0.254→0.797). Optimization spanned 7 phases (full trajectory in Appendix Table 8). The single largest improvement (+53%) came from discovering that returning full original dialogue text instead of LLM-generated summaries dramatically improves token-overlap F1, a non-obvious finding, since summaries are traditionally preferred for efficiency. The pipeline also found that prompt constraint *positioning* (before vs. after the question) matters more than constraint *content*, with one category improving +188% from this change alone. After Phase 7, four independent runs yielded F1 in [0.791, 0.797], confirming the performance ceiling and triggering the pipeline's decision to stop.

4.3 Main Results

To contextualize these results against existing memory systems, we conduct a controlled comparison of Omni-SimpleMem against six baselines across five LLM backbones (GPT-4o, GPT-4o-mini, GPT-4.1-nano, GPT-5.1, and GPT-5-nano). Table 1 reports per-category F1 on LoCoMo and five evaluation metrics on Mem-Gallery across all backbones.

On LoCoMo, Omni-SimpleMem achieves the highest overall F1 across all backbones, ranging from 0.492 (GPT-4.1-nano) to 0.613 (GPT-5.1), substantially outperforming SimpleMem (0.342–0.432), the current state-of-the-art on LoCoMo (Liu et al., 2026a). Omni-SimpleMem dominates on multi-hop, single-hop, and open-domain categories, with particularly large margins on open-domain questions.

On Mem-Gallery, Omni-SimpleMem achieves F1 ranging from 0.749 to 0.810, consistently outperforming all memory baselines by a wide margin. SimpleMem is again the strongest baseline (F1 up to 0.538 with GPT-5.1), but still trails Omni-SimpleMem by over 25 percentage points. These patterns confirm that Omni-SimpleMem's gains come from its architectural design (hybrid search, pyramid retrieval, knowledge graph augmentation) rather than from a single dominant component.

4.4 Detailed Analysis

4.4.1 Ablation Study

Table 2 presents an ablation study on LoCoMo that validates key design choices discovered by the pipeline. Specifically, we remove individual components and report the mean ΔF1 across 4 LLM backbones. Pyramid expansion (-17%) and BM25 hybrid search (-14%) are the most critical, and both were significantly refined by the autonomous pipeline during optimization. LLM summarization contributes -12%, confirming that compact MAU summaries are essential for retrieval quality. Reducing top-k from 20 to 5 costs -7%, while metadata context has a modest effect (-2%). Notably, the two most impactful components (pyramid expansion and hybrid search) are precisely those that received the most optimization iterations, suggesting the pipeline correctly allocated its search budget.

Table 2: Component ablation on LoCoMo (mean ΔF1$\times$100 across 4 backbones).

Component Removed	ΔF1	Rel.
w/o Pyramid Expansion	-10.2	-17%
w/o BM25 Hybrid	-8.5	-14%
w/o LLM Summarization	-7.3	-12%
Reduced top-k (5 vs 20)	-4.2	-7%
w/o Metadata Context	-1.4	-2%

4.4.2 Efficiency

Omni-SimpleMem achieves 5.81 queries/sec with 8 parallel workers (3.5$\times$ faster than the fastest baseline), enabled by read-only FAISS and BM25 indices supporting concurrent lookup (Figure 4, Table 3). All baselines are bottlenecked by sequential LLM generation (85–97% of per-query time), while Omni-SimpleMem parallelizes the retrieval-generation pipeline via thread-safe read-only indices.

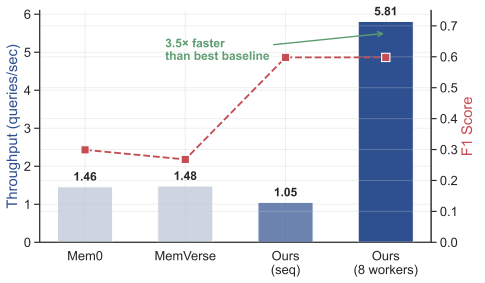


Figure 4: Throughput vs. F1. Omni-SimpleMem with 8 workers achieves 3.5$\times$ higher throughput.

Table 3: Efficiency breakdown.

Method	q/s	Ret.	Gen.
SimpleMem	1.68	45	550
MemVerse	1.48	70	596
Mem0	1.46	18	665
Omni-SimpleMem (w=1)	1.05	118	846
Omni-SimpleMem (w=8)	**5.81**	461	821

Ret./Gen. in milliseconds.

Across $\sim$50 experiments, the pipeline's discoveries fall into six categories (Table 4), several of which go beyond what traditional AutoML can achieve.

4.4.3 TAXONOMY OF AUTONOMOUS DISCOVERIES

Table 4: Taxonomy of discoveries made by the autonomous pipeline. The highest-impact categories (bug fix, architecture, prompt) all require code comprehension beyond traditional AutoML.

Category	Example	F1 Δ	AutoML?
Bug Fix	JSON response_format missing	+175%	✗
Data Repair	MAU timestamp correction (99.98% match)	+7%	✗
Architecture	Set-union hybrid search; full-text retrieval	+44%	Partial
Prompt Eng.	Constraint positioning; exact-citation instructions	+188%*	✗
Hyperparameter	top-k, token budget	+1–5%	✓
Format Align.	Output format to match evaluation rules	+5–63%	✗

*Category-specific improvement (KR on Mem-Gallery).

The most striking pattern is that the highest-impact discoveries are *not* hyperparameter adjustments. Bug fixes (+175%) and format alignment (+63%) individually exceed the cumulative contribution of all hyperparameter tuning experiments combined (+1–5% each). These categories all require code comprehension and cross-component reasoning that are unique to LLM-based research agents and fundamentally beyond the reach of traditional AutoML. A detailed case study illustrating the multi-hop retrieval pipeline on a real LoCoMo query is provided in Appendix A.

Why this domain supports autonomous optimization. Four properties of multimodal memory make these discoveries practical. First, each benchmark supplies a scalar metric after evaluation, providing feedback for the next hypothesis. Second, modular ingestion, retrieval, and generation components allow a proposed change to be examined through targeted ablations while retaining the rest of the system. Third, experiments complete within minutes to hours, making it feasible to evaluate dozens of hypotheses over a few days rather than selecting a single configuration in advance. Fourth, version-controlled changes permit unsuccessful modifications to be reverted before the next experiment. Together, these properties connect quantitative feedback to concrete code changes and recovery decisions. They also delimit the scope of our findings: transferring the pipeline to another domain requires suitable evaluation signals, tractable experiment costs, and mechanisms for isolating and reversing changes.

5 CONCLUSION

We have presented OMNI-SIMPLEMEM, a unified multimodal memory framework whose architecture and configuration are discovered through AUTORESEARCHCLAW, an autonomous research pipeline. Starting from a naïve baseline, the pipeline autonomously executed $\sim$50 experiments in $\sim$72 hours, achieving state-of-the-art on both LoCoMo and Mem-Gallery. The highest-impact discoveries, including bug fixes, architectural changes, and prompt engineering, require code comprehension and cross-component reasoning beyond the reach of traditional AutoML. Our taxonomy of six discovery types, together with the observation that multimodal memory is well-suited for autoresearch due to its scalar metrics, modular architecture, and fast iteration cycles, provides a roadmap for applying autonomous research pipelines to other complex AI system domains.

Limitations and future directions. Our evaluation covers two benchmarks with token-level F1 and exact match metrics; broader evaluation on open-ended generation tasks and user studies remains important future work. The pipeline's effectiveness depends on well-defined scalar metrics—domains with subjective or multi-dimensional evaluation criteria may require human-in-the-loop adaptations. The knowledge graph relies on LLM-based entity extraction, which can introduce resolution errors; specialized extraction models could improve robustness.

Ethics Statement

OMNI-SIMPLEMEM persistently retains multimodal personal information (text, images, audio, video) over extended time horizons, raising important considerations regarding data privacy and consent. Real-world deployment necessitates explicit user consent, transparent retention policies, and robust mechanisms to review, modify, or delete stored memories. The knowledge graph component, which extracts and links entities across memory units, could enable profiling if deployed without adequate access controls and encryption. Regarding the autonomous research pipeline, all AUTORESEARCH-CLAW experiments are conducted on established academic benchmarks with no real user data; deploying such pipelines on production systems would require additional oversight. We advocate for memory-augmented agents that prioritize user autonomy, data minimization, and the right to be forgotten.

Reproducibility Statement

We provide complete hyperparameters, per-benchmark configurations, and system architecture details in the appendix. Full optimization trajectories with per-iteration metrics and decisions are documented in Appendix F. Both benchmarks, LoCoMo (Maharana et al., 2024) and Mem-Gallery (Bei et al., 2026), are publicly available; we evaluate under identical splits and protocols as prior work using standard metrics (token-level F1, exact match, BLEU). We will open-source the complete framework, benchmark harnesses, and full experiment logs to enable independent replication.

AI Use Statement

We used Claude-Opus-4.7 to assist with code writing and to draft parts of the paper.

References

Anthropic. Claude 3 model card. https://www.anthropic.com, 2024.

Yuanchen Bei, Tianxin Wei, Xuying Ning, Yanjun Zhao, Zhining Liu, Xiao Lin, Yada Zhu, Hendrik Hamann, Jingrui He, and Hanghang Tong. Mem-gallery: Benchmarking multimodal long-term conversational memory for mllm agents. 2026. URL https://arxiv.org/abs/2601.03515.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, 2011.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162, pp. 2206–2240. PMLR, 2022.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. 2025. URL https://arxiv.org/abs/2504.19413.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. *International Conference on Machine Learning*, 2018.

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: Hands-free automl via meta-learning, 2022. URL https://arxiv.org/abs/2007.04074.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation, 2024. URL https://arxiv.org/abs/2310.03302.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019a.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019b.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474, 2020.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. 2026a. URL https://arxiv.org/abs/2601.02553.

Jiaqi Liu, Peng Xia, Siwei Han, Shi Qiu, Letian Zhang, Guiming Chen, Haoqin Tu, Xinyu Yang, Jiawei Zhou, Hongtu Zhu, Yun Li, Yuyin Zhou, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. AutoResearchClaw: Fully autonomous research from idea to paper, 2026b. URL https://github.com/aiming-lab/AutoResearchClaw.

Junming Liu, Yifei Sun, Weihua Cheng, Haodong Lei, Yirong Chen, Licheng Wen, Xuemeng Yang, Daocheng Fu, Pinlong Cai, Nianchen Deng, Yi Yu, Shuyue Hu, Botian Shi, and Ding Wang. Memverse: Multimodal memory for lifelong learning agents. 2025. URL https://arxiv.org/abs/2512.03627.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards llms as operating systems. 2023.

Alexander Panfilov, Peter Romov, Igor Shilov, Yves-Alexandre de Montjoye, Jonas Geiping, and Maksym Andriushchenko. Claudini: Autoresearch discovers state-of-the-art adversarial attack algorithms for llms. 2026. URL https://arxiv.org/abs/2603.24511.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. 2023. URL https://arxiv.org/abs/2304.03442.

Yaonan Qu and Meng Lu. Bilevel autoresearch: Meta-autoresearching itself. 2026. URL https://arxiv.org/abs/2603.23420.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. Ai-researcher: Autonomous scientific innovation. 2025. URL https://arxiv.org/abs/2505.18705.

Guiyao Tie, Pan Zhou, and Lichao Sun. A survey of ai scientists. 2026. URL https://arxiv.org/abs/2510.23045.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. 2025a. URL https://arxiv.org/abs/2502.12110.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. 2025b. URL https://arxiv.org/abs/2502.12110.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. 2025. URL https://arxiv.org/abs/2504.08066.

Yibo Yan, Shen Wang, Jiahao Huo, Jingheng Ye, Zhendong Chu, Xuming Hu, Philip S. Yu, Carla Gomes, Bart Selman, and Qingsong Wen. Position: Multimodal large language models can significantly advance scientific reasoning. 2025. URL https://arxiv.org/abs/2502.02871.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. 2025. URL https://arxiv.org/abs/2410.10594.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. 2024. URL https://arxiv.org/abs/2404.13501.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.

A Case Study: Multi-Hop Retrieval

We illustrate Omni-SimpleMem's retrieval pipeline on a real multi-hop query from LoCoMo that requires synthesizing facts across separate conversation sessions. The query asks: *"What subject have Caroline and Melanie both painted?"* The gold answer is "sunsets," but answering correctly requires retrieving each person's painting history from different sessions and identifying the overlap.

Hybrid search. Dense retrieval returns MAUs mentioning Caroline's paintings (*e.g.*, "Caroline painted a sunset") and Melanie's art projects (*e.g.*, "Mel and her kids painted a sunset with a tree"), but these appear in separate sessions with different surrounding context. BM25 recovers additional MAUs containing the keyword "paint" that rank lower in dense results. Set-union merging preserves the dense ordering and appends BM25-only matches.

Knowledge graph expansion. The query processor extracts seed entities Caroline (Person) and Melanie (Person). Neighborhood expansion links both entities to painting (Concept) and sunset (Concept) through separate relation paths, surfacing MAUs that mention each person's painting activities even when the surface text does not co-mention both names.

Pyramid retrieval and answer. Level 1 summaries from both relation paths are loaded; their similarity scores exceed θ, triggering Level 2 expansion of the full conversation text. The LLM identifies "sunsets" as the common subject and produces the correct answer (F1 = 1.0). In contrast, MemGPT, which lacks cross-session entity linking, hallucinates "Horses" (F1 = 0.0).

B Benchmark Details

LoCoMo. The Long-term Conversation Memory benchmark (Maharana et al., 2024) evaluates an agent's ability to recall and reason over extended multi-session dialogues. It comprises 10 conversations with 19–32 sessions each ($\sim$9K tokens per dialogue), generating 1,986 QA pairs across five categories: **Single-hop (SH)** requires retrieving a single fact; **Multi-hop (MH)** requires synthesizing information across multiple sessions; **Temporal (T)** tests reasoning about when events occurred; **Open-ended (O)** requires generating longer, contextual responses; and **Adversarial (A)** tests the ability to correctly refuse unanswerable questions. Evaluation uses token-level F1 with Porter stemming.

Mem-Gallery. Mem-Gallery (Bei et al., 2026) evaluates multimodal long-term memory in social interactions, comprising 1,711 QA pairs from 240 multi-session dialogues with 1,003 grounded images and 3,962 conversational rounds. Questions span 9 categories: Action Recognition (AR), Compound Decomposition (CD), Visual Search (VS), Timeline Learning (TTL), Temporal Reasoning (TR), Fact Retrieval (FR), Visual Reasoning (VR), Knowledge Reasoning (KR), and Multi-Entity Reasoning (MR). We use the public 20-dataset subset from HuggingFace.

C Baseline Descriptions

MemVerse (Liu et al., 2025) combines episodic and semantic memory layers with a multimodal knowledge graph. Each ingested item requires 3 LLM calls (summarization, entity extraction, relation linking), yielding an ingestion rate of 0.22 items/sec. Retrieval uses embedding-based similarity over the knowledge graph.

Mem0 (Chhikara et al., 2025) performs dynamic fact extraction from conversational inputs, storing structured facts with optional graph memory augmentation. It achieves 1.28 items/sec ingestion and 18ms search latency, but is fundamentally text-only.

SimpleMem (Liu et al., 2026a) introduces efficient lifelong memory management through three core mechanisms: memory atomization (decomposing inputs into fine-grained atomic units), adaptive consolidation (merging related memories to reduce redundancy), and context-aware pruning. It achieves state-of-the-art performance on LoCoMo with significantly reduced token cost ($\sim$45% less than Mem0) and 4$\times$ faster total processing time.

Claude-Mem (Anthropic, 2024) provides commercial embedding-based dialogue memory with per-turn image storage.

A-MEM (Xu et al., 2025a) introduces agentic memory management where the LLM itself decides when and how to reorganize stored memories, including operations such as merging, splitting, and abstracting memory entries to maintain a compact and relevant memory state.

MemGPT (Packer et al., 2023) draws an analogy to operating system memory hierarchies, implementing a main context (analogous to RAM) and external storage (analogous to disk) with explicit `push`/`pop` operations managed by the LLM agent. It achieves strong performance on conversational tasks but is fundamentally text-only and bottlenecked by sequential LLM calls for each memory management decision.

D OMNI-SIMPLEMEM TECHNICAL DETAILS

D.1 SYSTEM ARCHITECTURE

The complete OMNI-SIMPLEMEM implementation comprises 13,300 lines of Python across 11 subpackages:

- **Core** (MAU data structure, event hierarchy, unified configuration)
- **Processors** (modality-specific ingestion: text, image, audio, video)
- **Storage** (three-layer persistence: MAU store as JSON Lines, FAISS vector store, cold storage on filesystem/S3)
- **Retrieval** (pyramid retriever, query processor, expansion manager)
- **Knowledge** (entity extractor, in-memory knowledge graph, graph retriever)
- **Orchestrator** (central coordinator, 873 lines, unified API)

D.2 HYPERPARAMETERS AND BENCHMARK-SPECIFIC CONFIGURATIONS

Table 5 lists the complete hyperparameter specification. Table 6 summarizes the per-benchmark configurations discovered by the pipeline.

Table 5: Hyperparameter specification of OMNI-SIMPLEMEM.

Component	Parameter	Value
Visual Ingestion	CLIP threshold τ_{high}	0.9
	CLIP threshold τ_{low}	0.7
	Frame buffer size	3
Audio Ingestion	VAD threshold	0.5
Text Ingestion	Jaccard threshold τ_{dup}	0.8
Entity Resolution	Merge threshold τ_{res}	0.85
	Semantic weight α	0.5
	Graph decay β	0.7
	Expansion hops h	2
Pyramid Retrieval	Auto-expand threshold θ	0.4
	Token budget B	6,000
	BM25 parameters k_1, b	1.5, 0.75

D.3 KNOWLEDGE GRAPH SCHEMA

The knowledge graph uses 7 entity types (Person, Location, Object, Event, Concept, Time, Organization) and 7 relation types (located_in, part_of, interacts_with, owns, attended, created_by, related_to). Entity extraction is performed by GPT-4o in JSON mode during MAU creation.

Table 6: Per-benchmark configurations discovered by the autonomous pipeline (†varies per question category).

Parameter	LoCoMo	Mem-Gallery
Embedding	text-emb-3-large	MiniLM-L6-v2
Embedding dim	3072	384
top-k	20–30	20–40†
Token budget	6,000	2K–8K†
Per-doc memory	No	Yes
System prompt	Yes	Yes

E AutoResearchClaw Pipeline Details

The complete pipeline comprises 23 stages organized in 8 phases:

A. Research Scoping (Stages 1–2): Formulates SMART research goals; auto-detects hardware (GPU type, VRAM).
B. Literature Discovery (Stages 3–6): Queries OpenAlex, Semantic Scholar, arXiv; screens by relevance/quality (gate at Stage 5); extracts structured knowledge cards.
C. Knowledge Synthesis (Stages 7–8): Clusters findings; generates falsifiable hypotheses via multi-agent debate.
D. Experiment Design (Stages 9–11): Designs protocols (gate at Stage 9); generates hardware-aware Python code with AST validation; schedules resources.
E. Experiment Execution (Stages 12–13): Runs experiments in sandbox/Docker/SSH; self-healing loop (up to 10 retries) with structured deficiency classification.
F. Analysis & Decision (Stages 14–15): Statistical analysis (t-tests, bootstrap CI); autonomous PROCEED/PIVOT/ITERATE decision.
G. Documentation (Stages 16–19): Generates outline; writes draft; simulated peer review; revision.
H. Finalization (Stages 20–23): Quality gate (Stage 20); knowledge archival; LaTeX export; 4-layer citation verification (arXiv ID, DOI, title match, relevance score).

F Optimization Trajectory Details

Table 7: Optimization trajectory on LoCoMo. The pipeline autonomously discovers 7 successive improvements and correctly reverts 2 failed experiments.

Iter	Key Discovery	F1	Δ	Type
0	Naïve baseline	0.117	—	—
1	JSON `response_format` missing	0.322	+175%	Bug fix
2	BM25 hybrid	0.464	+44%	Architecture
3	Anti-hallucination prompting	0.516	+11%	Prompt
4b	Evaluation format alignment	0.543	+5%	Format
5	MAU timestamp correction	0.580	+7%	Data repair
6	top-k=30 + temporal hints	0.577	−0.5%	Hyperparam
7b	Adaptive top-k + metadata	0.583	+0.5%	Hyperparam
8	Forced exact-word copying	0.551	−5.5%	Reverted
9	Increased BM25 results	0.575	−1.4%	Reverted
benchmark validation		**0.598**	**+411%**	
SimpleMem SOTA (Liu et al., 2026a)		0.432		

G Full Iteration Logs

G.1 LoCoMo: Per-Category Performance Across Iterations

Table 9 shows how per-category F1 evolved across the optimization trajectory.

Table 8: Optimization trajectory on Mem-Gallery (39 experiments, 7 phases). Each phase represents a qualitative shift in the pipeline's optimization strategy.

Phase	Focus	F1 Range	Δ	Key Discovery
1	Environment setup	0.254→0.353	+39%	LLM upgrade + local embedding
2	Architecture	0.353→0.690	+96%	Full-text retrieval + image BM25
3	Fine-tuning	0.690→0.717	+4%	Constraint position sensitivity
4	Scale validation	0.717→0.726	+1%	Data completeness > algorithms
5	Exact citation	0.726→0.771	+6%	BM25 tokenization fix (+0.018)
6	Visual reasoning	0.771→0.789	+2%	Image catalog + context
7	Plateau exploration	0.789→0.793	+1%	Performance ceiling confirmed
Final (20 datasets)		**0.797**	**+214%**	
MuRAG SOTA (Bei et al., 2026)		0.697		

Table 9: LoCoMo per-category F1 across iterations (conv-26, 199 QA pairs).

Iter	Cat1 (MH)	Cat2 (T)	Cat3 (O)	Cat4 (SH)	Cat5 (A)
Baseline	0.093	0.047	0.116	0.219	0.000
Iter 1	0.206	0.117	0.229	0.381	0.447
Iter 2	0.292	0.117	0.282	0.418	1.000
Iter 3	0.351	0.335	0.346	0.444	1.000
Iter 4b	0.389	0.309	0.382	0.466	1.000
Iter 5	0.398	0.487	0.388	0.445	1.000
Iter 7b	0.404	0.543	0.398	0.440	1.000

Notable patterns: Cat 5 (Adversarial) jumps from 0.447 to 1.000 after alignment with evaluation-standard refusal phrases in Iter 2. Cat 2 (Temporal) shows the most dramatic improvement (+0.496 total), driven primarily by the timestamp correction in Iter 5. Cat 3 (Open-ended) is our strongest category vs. prior SOTA, exceeding SimpleMem by +0.200.

G.2 Mem-Gallery: Detailed Phase-by-Phase Log

Phase 1: Environment Setup (Exp-000 to 001). Starting from F1 = 0.254 with gpt-4.1-nano and CLIP-B/32 512-dim embeddings, the pipeline first upgraded the LLM to gpt-4o and switched to local all-MiniLM-L6-v2 embeddings (384d) after encountering API proxy errors. Result: F1 = 0.353 (+39%).

Phase 2: Architecture Breakthrough (Exp-002 to 004). The largest single improvement (+53%) came from returning full original dialogue text instead of LLM summaries, which is counter-intuitive since summaries are traditionally preferred. The pipeline further added BM25 hybrid search (set-union merging) and created a dedicated image-caption BM25 index for visual question categories. Combined result: F1 = 0.690 (+96%).

Phase 3: Fine-Tuning (Exp-004b to 012). The pipeline discovered that format constraint *position* (before vs. after the question) matters more than content at temperature=0. KR category improved +188% from repositioning alone. Several prompt-tuning experiments (Exp-008 to 011) failed and were reverted. Net result: F1 = 0.717 (+4%).

Phase 4: Scale Validation (Exp-014 to 018). Validating on 14 datasets exposed data completeness issues (store_only not completed before qa_only). The pipeline learned that *data completeness > algorithmic optimization*. Result: F1 = 0.726 (+1%).

Phase 5: Exact Citation (Exp-020 to 023). Two key discoveries: (1) "Use the exact words and phrases from the conversation" instruction improved 12/15 datasets (+0.031 F1); (2) a simple BM25 tokenization fix (stripping punctuation: "sushi." → "sushi") yielded +0.018 F1, which is more than 10 rounds of prompt engineering. Result: F1 = 0.771 (+6%).

Phase 6: Visual Reasoning Enhancement (Exp-026 to 027). Augmenting the Image Catalog with dialogue context (full_text[:300]) for each image improved VR category by +0.087. Adding temporal ordering for TR/CD categories contributed another +0.006. Result: F1 = 0.789 (+2%).

Phase 7: Plateau Exploration (Exp-028 to 039b). The pipeline explored model comparison (gpt-4.1 $\approx$ gpt-4o), per-category prompt refinements, and various BM25 configurations. After 4 independent runs yielding F1 in [0.791, 0.797], the pipeline correctly identified the performance ceiling ($\sim$0.795 due to random noise) and terminated. Final peak: F1 = 0.797.

H Prompt Catalog

This section documents representative prompts used throughout Omni-SimpleMem, organized by system component. All prompts are reproduced from the source code; minor formatting adjustments are made for readability.

H.1 Core System Prompts

H.1.1 Answer Generation

The orchestrator uses a two-part prompt structure for answer generation: a system prompt that defines the assistant's role, and a user prompt that injects retrieved context with structured output requirements. This design enforces JSON-mode output for reliable answer extraction.

Answer Generation – System Prompt

```
You are a professional Q&A assistant. Your task is to extract
concise, accurate answers from the provided memory context. You
should make reasonable inferences from the context when possible.
You must output valid JSON format.
```

Answer Generation – User Prompt Template

```
Based on these memories:
{context}
Question: {question}
Requirements:
1. First, think through the reasoning process
2. Provide a CONCISE answer (short phrase, ideally under 10 words).
Use exact words and phrases from the context whenever possible
rather than paraphrasing.
3. Answer based on the provided context. You may make reasonable
inferences from the information given.
4. Try your best to answer. Only respond with 'unknown' if the
context contains absolutely NO relevant information about the topic
asked.
5. For counting questions, answer with just the number (e.g., '2'
not 'twice').
6. For yes/no questions, start with 'Yes', 'No', 'Likely yes', or
'Likely no'.
7. When listing multiple items, separate them with commas.
8. Return your response in JSON format.
Output Format:
{"reasoning": "Brief explanation", "answer": "Concise answer"}
```

The LLM is called with `temperature=0.1` and `response_format=json_object` to ensure deterministic, parseable output. The structured JSON output was a key discovery of the autonomous pipeline: without explicit format enforcement, the model produced verbose natural-language answers that degraded F1 by 175% (see Iteration 1, Table 7).

H.1.2 Entity and Relation Extraction

The knowledge graph module extracts typed entities and directed relations from each MAU summary. The prompt enforces a structured JSON schema with confidence scores, enabling downstream entity resolution to filter low-confidence extractions.

Entity Extraction – System Prompt

```
You are an expert knowledge extraction system. Extract entities
and relations from the given text.

Output JSON format:

{
  "entities": [
    {"name": "entity name",
     "type": "PERSON|OBJECT|LOCATION|EVENT|
              CONCEPT|TIME|ORGANIZATION",
     "attributes": {"key": "value"},
     "confidence": 0.0-1.0}
  ],
  "relations": [
    {"subject": "entity name",
     "predicate": "relation type",
     "object": "entity name",
     "confidence": 0.0-1.0}
  ]
}

Guidelines:
• Extract all meaningful entities mentioned
• Identify relationships between entities
• Use standardized entity types
• Assign confidence scores based on clarity
• For visual descriptions, pay attention to spatial relationships
  and object attributes
```

Entity Extraction – User Prompt

```
Extract entities and relations from this text:
{text}
```

The 7 entity types (Person, Location, Object, Event, Concept, Time, Organization) were selected to cover the dominant categories observed across both benchmarks. The multimodal guideline ("for visual descriptions, pay attention to spatial relationships") enables the same extraction prompt to operate uniformly over text summaries generated from any input modality.

H.1.3 Query Intent Analysis

Before retrieval, the query processor optionally analyzes query intent to extract structured metadata. This analysis informs retrieval strategy selection (e.g., enabling temporal sorting for temporal queries, or activating visual search for image-related queries).

Query Analysis – System Prompt

```
You are a query analysis assistant. Output valid JSON only.
```

Query Analysis – User Prompt Template

```
Analyze this memory retrieval query and extract:
1. intent_type: one of [factual, temporal, comparative,
exploratory, verification]
```

```
2. entities: list of named entities (people, places, things)
3. time_references: any temporal expressions (yesterday, last
week, etc.)
4. modality_hints: likely content types [visual, audio, text,
video]
5. reformulated_query: optimized search query
Query: {query}
Respond in JSON format only, no explanation.
```

H.2 Modality Processor Prompts

Omni-SimpleMem employs modality-specific prompts during ingestion to convert heterogeneous inputs into the unified MAU text summary. A key design principle is prompt minimalism: concise instructions yield shorter, more retrieval-friendly summaries at ingestion time, while detailed prompts are reserved for on-demand expansion during answer generation.

H.2.1 Image Captioning

Two levels of image captioning serve different stages of the pyramid retrieval pipeline:

Concise Image Caption (Ingestion – Level 1)

```
Describe this image in one concise sentence. Focus on key objects,
actions, and context.
```

Detailed Image Caption (Expansion – Level 3)

```
Provide a detailed description of this image including:
1. Main subjects and their appearance
2. Actions or events taking place
3. Setting and environment
4. Notable objects
5. Any text visible
```

The concise variant is used during ingestion (`max_tokens=150`), producing $\sim$10-token summaries stored in hot storage. The detailed variant is invoked only during Level 3 pyramid expansion (`max_tokens=500`), loading richer descriptions from cold storage on demand. This two-tier design reduces storage cost while preserving access to fine-grained visual detail when needed for complex queries.

H.2.2 Audio and Video Summarization

Audio content is first transcribed via speech-to-text, then summarized:

Audio Transcript Summary

```
Summarize this audio transcript in one concise sentence:
{transcript[:2000]}
Summary:
```

Video MAUs combine visual frame descriptions with audio transcripts into a unified summary:

Video Content Summary

```
Summarize this video content in 1-2 sentences:
{frame_descriptions}
{audio_transcript}
Video summary:
```

Both prompts follow the minimalism principle: single-sentence summaries are sufficient for embedding-based retrieval in the first pyramid level, while the original transcripts and frame descriptions remain accessible in cold storage for deeper expansion.

H.3 Mem-Gallery Benchmark Prompts

Mem-Gallery evaluation employs a multi-level prompt architecture discovered through the autonomous optimization pipeline. A global system prompt defines the task context, category-specific format constraints control output formatting, and a dialogue agent prompt frames each question. The pipeline's key discovery was that constraint *positioning* (before vs. after the question) significantly affects performance at low temperature, with per-category optimization yielding up to +188% improvement on individual categories (Section 3.3).

H.3.1 System Prompt

Mem-Gallery System Prompt

```
You are an AI assistant evaluated on multimodal long-term
conversational memory. For the given question-answering task, your
responses must be concise, yet complete enough to accurately answer
the questions. If multiple pieces of information about the same
event appear in the conversation, always rely on the most recent
information.

The question-answering evaluation will contain several multimodal
task types:

Factual Retrieval: Retrieve explicit facts mentioned in the
conversation for the answer.
Multi-entity Reasoning: Combine the retrieved information to reason
and infer an answer.
Temporal Reasoning: Resolve time-dependent questions.
Visual-centric Reasoning: Besides textual information, answer
questions using visual images in the conversation.
Test-time Learning: Learn new visual knowledge from provided images
within historical dialogue and use it in question-answering.
Visual-centric Search: Find the image(s) that match the information
in a given query and return their image ID(s).
Conflict Detection: Detect contradictions between the conversation
history and the information provided in the question.
Knowledge Resolution: Resolve knowledge conflicts or updates by
prioritizing the most recent information.
Answer Refusal: Decline to answer when the information does not
exist in the conversation history.

Follow all instructions strictly. Only answer using information
contained within the multimodal conversation. Do not hallucinate.
Always remain consistent and grounded in the dialogue history.
```

H.3.2 Category-Specific Format Constraints

Three of the nine categories receive explicit format constraints appended to the question. These constraints were iteratively refined by the pipeline to align model output with evaluation metrics:

Answer Refusal (AR) Constraint

```
Provide your answer based on the information in the conversation.
Only if the information about the question is not present in the
conversation, reply with: "Not mentioned."
```

Conflict Detection (CD) Constraint

```
Please check whether this information conflicts with the
conversation, and reply strictly with either "Yes." or "No."
```

Visual Search (VS) Constraint

```
Return the image_id of the image(s). If there are multiple images,
sort them in ascending order and separate them by commas. Format
example: "D2:IMG_003, D2:IMG_010, D10:IMG_002" (for format
reference only).
```

The remaining six categories (FR, MR, TR, VR, TTL, KR) use the default system prompt without additional format constraints. The pipeline discovered that adding constraints to these categories either had no effect or degraded performance due to over-specification.

H.3.3 Dialogue Agent Prompt

Each question is framed by a dialogue agent prompt that is appended after the retrieved memory context:

Mem-Gallery Dialogue Agent Prompt

```
Your task is to answer the question about the conversation
between {speaker_a} and {speaker_b} in a concise manner with the
help of memory content. Please only provide the content of the
answer, without including introductory phrases like 'answer:'.
For questions that require answering a date or time, strictly
follow the format and provide a specific date or time whenever
possible. Generate answers primarily concise, yet complete enough
to accurately answer the questions.

The current question is as follows:
{observation} {format_constraint}
```

H.3.4 Reasoning Model Adaptation

When using reasoning-class models (e.g., o1, o3) that tend to produce verbose chain-of-thought outputs, the pipeline appends a conciseness boost to the system prompt. This adaptation was discovered during Phase 7 when testing alternative backbones:

Reasoning Model Conciseness Boost

```
CRITICAL RULE -- Answer brevity:
You are being evaluated by token-level F1 against a short reference
answer. Verbose answers DESTROY your score. Follow these rules
WITHOUT exception:

1. Output ONLY the essential answer -- no reasoning, no
justification, no restating the question.
2. For Yes/No questions → answer "Yes." or "No." ONLY. Do NOT
add explanations.
3. For factual questions → answer with the bare fact.
✓ "Maltese" ✗ "The dog in the image appears to be a Maltese."
4. For list questions → comma-separated items, nothing else.
5. NEVER use bullet points, numbered lists, or multi-line
formatting.
6. NEVER start with "The answer is", "Based on the conversation",
etc.
```

H.3.5 MEMORY CONTEXT INJECTION

Retrieved memories are formatted and injected into the prompt using the following template. For multimodal memories, image content is interleaved with textual context as base64-encoded content parts:

Memory Context Template

```
The retrieved memory contents are as follows:
{memory_context}
```

Multimodal Memory Template

```
{textual_context}
image:
image_id:  {image_id}
image_content:  [base64-encoded image]
```