

OmniMem: 自研究引导的终身多模态智能体记忆发现

Jiaqi Liu¹, Zipeng Ling², Shi Qiu¹, Yanqing Liu³, Siwei Han¹, Peng Xia¹, Haoqin Tu³, Zeyu Zheng⁴, Cihang Xie³, Charles Fleming⁵, Mingyu Ding¹, Huaxiu Yao¹

¹UNC-Chapel Hill ²University of Pennsylvania ³University of California, Santa Cruz

⁴University of California Berkeley ⁵Cisco

jqliu@cs.unc.edu huaxiu@cs.unc.edu

Abstract

智能体在越来越长的时间跨度上运行，但其保留、组织和回忆多模态经验的能力仍是一个关键瓶颈。构建有效的终身记忆需要在架构、检索策略、提示工程和数据流水线等多个方面进行探索；这一设计空间过于庞大且高度耦合，难以通过人工探索或传统自动机器学习 (AutoML) 有效覆盖。我们部署了一个自主研究流水线，以发现 OmniMem——一种面向终身智能体的统一多模态记忆框架。从一个初始基准 (在 LoCoMo 上 $F1 = 0.117$) 出发，该流水线自主执行 ~ 50 次实验，跨越两个基准测试，诊断失败模式，提出架构修改，并修复数据流水线中的缺陷，整个过程在内部循环中无需人工干预。最终系统在两个基准上均达到当前最优表现，相较于初始配置，LoCoMo 上的 F1 提升了 +411% (从 0.117 \rightarrow 0.598)，Mem-Gallery 上提升了 +214% (从 0.254 \rightarrow 0.797)。关键的是，最具影响力的发现并非超参数调整：错误修复 (+175%)、架构变更 (+44%) 以及提示工程 (特定类别上 +188%) 各自单独带来的提升，已超过所有超参数调参的累计贡献，展现出远超传统 AutoML 的能力。我们提出了六种发现类型的分类体系，并识别出多模态记忆特别适合自主研究的四个特性，为将自主研究流水线应用于其他人工智能系统领域提供了指导。代码已公开于 this <https://github.com/aiming-lab/OmniMem>。

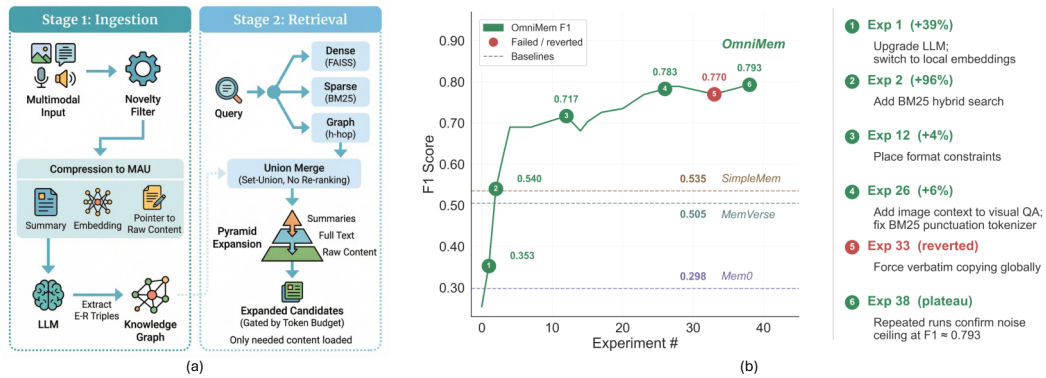


图 1: OmniMem 发现过程概述。(a) 发现的架构：多模态输入经过新颖性过滤，压缩为 MAU，通过金字塔扩展的混合稠密-稀疏图搜索进行检索。(b) 在 Mem-Gallery 上的自主最优化轨迹：39 次实验使 F1 从 0.254 提升至 0.793 (提升 214%)。

1 引言

大规模语言模型的最新进展催生了能够使用工具、进行多步推理和跨模态理解的智能体 (Yao et al., 2023; Yan et al., 2025)。这些智能体在较长的时间范围内与用户交互，其运行过程中持续积累来自文本、图像、音频和视频等多种模态的数据流。然而，它们在保留、组织和回忆过往经验方面的能力仍是一个关键瓶颈 (Zhang et al., 2024; Xu et al., 2025b)。构建有效的终身多模态记忆需要在广阔的设计空间中进行权衡，涵盖架构选择（如何组织存储）、检索策略（如何查找相关信息）、提示工程（如何向大语言模型呈现上下文）以及数据流水线配置（如何摄入和处理异构输入）。

现有的智能体记忆方法主要分为两大类，每类均有显著的局限性。第一类存储原始输入，并通过嵌入相似度进行检索 (Lewis et al., 2020; Borgeaud et al., 2022)，随着记忆的增长，存在存储膨胀和检索噪声的问题。第二类引入了结构化记忆管理并采用显式操作 (Packer et al., 2023; Park et al., 2023)，但通常仅处理文本信息，忽略了丰富的视觉和听觉信号。关键的是，这两类方法均源于人工研究周期：研究人员提出改进假设，实现该方案，在基准上评估，再迭代优化。单个研究人员每天可能仅能探索少量配置，且容易忽略紧密耦合组件之间的关键交互。传统 AutoML 方法 (Hutter et al., 2019a) 可在预定义的数值超参数空间中进行搜索，但无法完成代码理解、错误诊断、架构重构以及跨组件推理，而这些恰恰是复杂系统性能提升的主要来源。因此，现有的记忆系统继承了设计者的盲点——这些局限性本可通过更系统的搜索加以避免。

近期关于自主科学发现的工作 (Lu et al., 2024; Romera-Paredes et al., 2024; Panfilov et al., 2026) 表明，只要目标领域具备明确且可量化的评估信号，大型语言模型智能体便可自主发现性能优于人工设计基准的新型算法。我们探讨该范式是否适用于复杂、多组件的人工智能系统，并给出了肯定回答。我们部署了 AutoResearchClaw (Liu et al., 2026b)——一个 23 阶段的自主研究流水线，用于发现 OmniMem，一种面向终身学习智能体的统一多模态记忆框架。从一个初始基线（在 LoCoMo 上 $F1 = 0.117$ ）出发，该流水线在两个基准上自主执行 ~ 50 次实验，通过迭代诊断失败模式、提出架构修改、修复数据流水线中的缺陷并验证改进效果，整个内环过程无需人工干预。最终系统在两个基准上均达到当前最优水平，相较于初始配置，在 LoCoMo 上的 F1 提升 +411% ($0.117 \rightarrow 0.598$)，在 Mem-Gallery 上的 F1 提升 +214% ($0.254 \rightarrow 0.797$)。关键的是，最具影响力的发现并非超参数调整：错误修复 (+175%)、架构变更 (+44%) 以及提示工程（特定类别上 +188%）各自单独带来的提升均超过所有超参数调参的累计贡献，展现出远超传统 AutoML 能力的根本性突破。

流水线最重要的发现包括定义 OmniMem 的三条架构原则。首先，**选择性摄入**：轻量级感知编码器衡量每个输入信号的信息新颖性，并在存储前丢弃冗余内容，显著降低存储需求。其次，**统一表示**：所有记忆，无论模态如何，均以多模态原子单元 (MAUs) 表示，将轻量级元数据与重型原始数据分离，从而实现基于紧凑元数据的快速搜索，同时按需保留完整内容访问能力。第三，**渐进式检索**：一种金字塔机制分三个阶段扩展信息（摘要、细节、原始证据），每一阶段由 token 预算控制，依托于结合稠密向量检索与稀疏关键词匹配的混合搜索策略，通过集合并集实现，该策略由流水线自主发现。我们的关键观察是，多模态记忆特别适合自主研究流水线，原因在于四个特性：即时标量评估指标支持紧密的最优化环，模块化架构允许组件独立修改，快速迭代周期（每次实验 1-2 小时）可在数日内支持数十个假设，以及版本控制的代码修改使得失败实验可被干净地回滚。

总而言之，我们的主要贡献是 OmniMem，这是一个统一的多模态记忆框架，其架构和配置通过 AutoResearchClaw 发现，并在两个评估基准上取得了最先进的性能。除了系统本身，我们还提供了对 ~ 50 次实验中自主发现的全面分类，揭示了最具影响力的改进超出了传统

AutoML 的范围，同时刻画了流水线的收敛行为、失败模式以及自动恢复模式。我们的分析进一步识别出四个使多模态记忆成为自研究（autoresearch）特别适合领域的特性，为未来将此类方法应用于其他人工智能系统领域提供了指导。

2 相关工作

自主科学发现。人工智能驱动研究的愿景已迅速发展。AI Scientist (Lu et al., 2024) 在三个机器学习领域实现了每篇论文 ~\$15 的端到端论文生成，其后续版本 AI Scientist v2 (Yamada et al., 2025) 通过智能体树搜索消除了人工编写的模板。FunSearch (Romera-Paredes et al., 2024) 将大语言模型的创造力与程序化评估相结合，以发现新颖的数学构造。AutoResearchClaw (Liu et al., 2026b) 引入了包含多智能体辩论和自愈执行的 23 阶段自主研究流水线。AI-Researcher (Tang et al., 2025) 提出了具有结构化迭代优化的协作式多智能体框架，而双层自研 (Qu & Lu, 2026) 则对搜索策略本身进行元最优化。Tie et al. (2026) 的全面综述描绘了从基础模块（2022–2023 年）到闭环系统（2024 年），再到可扩展的人机协同（2025 年及以上）的演化历程。我们将自研范式应用于多组件人工智能系统的最优化，其中挑战已从发现孤立的成果转变为诊断并改进紧密耦合模块间的交互。

多模态记忆系统。增强记忆的大型语言模型智能体已从仅文本系统发展而来，包括基于操作系统启发的记忆层次结构的 MemGPT (Packer et al., 2023)、采用近期内-重要性-相关性评分机制的 Generative Agents (Park et al., 2023)、具备高效终身记忆的 SimpleMem (Liu et al., 2026a)，以及由大语言模型引导重组的 A-Mem (Xu et al., 2025a)，如今已演进至多模态架构。MemVerse (Liu et al., 2025) 将情景-语义记忆与多模态知识图谱结合，但每条摄入项需调用三次大语言模型。Mem0 (Chhikara et al., 2025) 提供动态事实提取，并支持可选的图谱记忆。VisRAG (Yu et al., 2025) 直接索引视觉页面，避免了文本提取带来的信息损失。Claude-Mem (Anthropic, 2024) 提供基于商业嵌入的对话记忆。这些系统均需要对检索策略、摄入流水线和提示配置进行大量手动调优，而这正是自主研究流水线能够加速最优化的领域。

自动化机器学习。神经架构搜索 (Hutter et al., 2019b; Zoph & Le, 2017; Liu et al., 2019) 自动化模型设计，但主要在定义明确的架构搜索空间中运行，采用可微分或基于强化学习的目标。超参数优化方法 (Bergstra et al., 2011; Falkner et al., 2018) 能够高效地遍历连续和分类空间，而像 Auto-sklearn 2.0 (Feurer et al., 2022) 这样的系统则通过元学习自动化完整的机器学习流水线，包括预处理和模型选择。最近，基于大语言模型的智能体已被应用于机器学习任务：MLAgentBench (Huang et al., 2024) 在涉及代码修改的机器学习研究任务上对大语言模型智能体进行基准测试，展示了语言引导优化的潜力。我们的情景存在根本差异：「搜索空间」不仅包含超参数和架构选择，还包括提示工程、数据流水线中的错误检测与修复、评估格式对齐以及跨组件交互诊断，这些均需要超越传统自动化机器学习的自然语言理解与代码修改能力。

3 自研引导的发现 OmniMem

在本节中，我们描述了自主最优化过程及其所生成的系统。我们首先概述流水线 (§3.1)，然后介绍发现的 OmniMem 架构 (§3.2)，最后阐述针对基准的最优化策略 (§3.3)。

3.1 流水线概览

如第 1 节所述，多模态记忆系统的设计空间过于庞大且相互关联，手动探索难以有效覆盖。为解决此问题，我们部署了 AutoResearchClaw (Liu et al., 2026b)，一个包含 23 个阶段的自

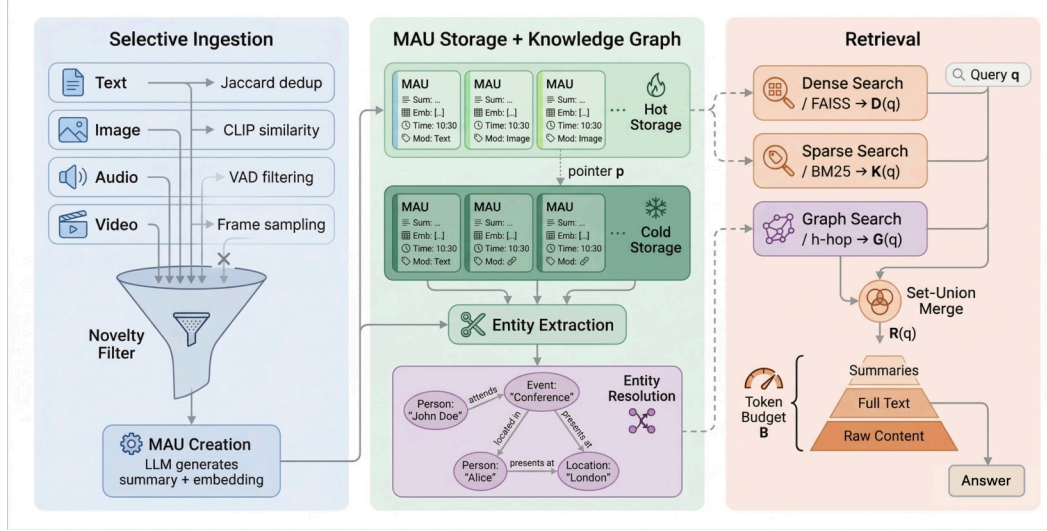


图 2: OmniMem 架构概览。左侧：选择性摄入通过模态特定的新颖性检测器过滤多模态输入（文本、图像、音频、视频），并创建带有 LLM 生成摘要和嵌入的 MAU。中心：MAU 存储在热存储（摘要、嵌入、元数据）和冷存储（原始内容）中，实体提取构建带有类型实体和关系的知识图谱。右侧：检索通过集合并集合并结合稠密（FAISS）、稀疏（BM25）和图（ h -跳）搜索，然后在 token 预算 B 下，通过金字塔机制逐步扩展结果（摘要 \rightarrow 全文 \rightarrow 原始内容）。

主研究流水线，以系统性地优化 OmniMem。该流水线接收三个输入：(1) SimpleMem (Liu et al., 2026a) 代码库，一个单峰值文本仅有的终身记忆框架，作为起点；(2) 两个带有定量指标 (F1) 的基准评估工具包；(3) 对大语言模型 (LLM) 提供商的 API 访问权限。随后进入迭代环：在每一步中，流水线分析先前结果，生成改进假设，在代码中实现变更，于基准上进行评估，并决定是否 **继续**（指标提升 $\geq 0.5\%$ ），**迭代**（结果模糊；优化当前假设），或 **转向**（连续两次退化；回滚并尝试新方向）。在总共 ~ 50 次实验中，大多数结果导致“继续”决策，其余则在“迭代”与“转向”之间分配。流水线的完整阶段（范围界定、文献发现、多智能体辩论、实验设计、沙盒执行、分析、文档编写和最终确认）详见 Liu et al. (2026b)；此处我们聚焦于其发现成果及促成这些发现的机制。

3.2 发现的架构

流水线以 SimpleMem (Liu et al., 2026a) 为起点，这是一个单峰值的仅文本终身记忆框架。我们向 AutoResearchClaw 提供 SimpleMem 的代码库，并指示其将系统从仅支持文本的记忆扩展为完整的多模态支持，自主设计用于摄入、存储和检索异构信号（文本、图像、音频、视频）所需的架构组件。通过迭代实验，流水线最终收敛到一个围绕三个原则组织的架构：选择性摄入、渐进式检索和结构化知识（图 2）。

3.2.1 选择性摄取

第一个原则是**选择性摄入**：系统首先过滤冗余输入，然后将保留的信号封装成统一的多模态表示。

基于新颖性的过滤。在任何数据进入记忆存储之前，轻量级的感知编码器会评估传入信息的新颖性，并丢弃冗余内容。对于视觉信息，通过比较连续帧之间的 CLIP 嵌入来检测场景变化；对于音频，使用 VAD 语音概率门控来保留语音部分，拒绝静音；对于文本，通过与近

期摘要的 Jaccard 重叠度过滤近似重复内容。这种过滤显著降低了存储需求，同时不丢失语义内容。

多模态原子单元。通过新颖性过滤器的信号被封装为 **多模态原子单元** (MAUs), $\mathcal{M} = \langle s, \mathbf{e}, p, \tau, m, \ell \rangle$, 将紧凑的可搜索元数据与庞大的原始内容解耦。此处 s 为文本摘要, $\mathbf{e} \in \mathbb{R}^d$ 为其嵌入, p 指向冷存储中的原始内容, τ 为时间戳, m 为模态, ℓ 存储与其他 MAUs 的结构化链接。这形成了一种两层设计: 热存储保存摘要、嵌入及时间和图谱元数据以实现快速检索, 而 冷存储保存大型资产 (图像、音频、视频), 并通过 p 按需访问。

3.2.2 渐进式检索与混合搜索

一旦记忆被摄入并存储为 MAU, 接下来的挑战就是在查询时如何高效地检索它们。第二个原则是**渐进式检索**: 不是一次性将所有检索到的内容加载到 LLM 上下文中, 而是 OmniMem 在明确的 token 预算下分阶段扩展信息。

混合稠密-稀疏检索。给定用户查询 q , 通过 FAISS (Johnson et al., 2021) (一个用于高维向量高效相似度搜索的库) 进行稠密检索, 基于 L2 规范化的 MAU 嵌入向量的内积搜索, 获得语义相似的候选结果 $\mathcal{D}(q)$ 。同时, 对 MAU 摘要进行 BM25 (Robertson & Zaragoza, 2009) 得分, 得到关键词匹配的候选结果 $\mathcal{K}(q)$ 。自主流水线的一个关键发现是集合合并: 经验上, 基于得分的重排序 (标准方法) 会破坏语义排序并降低性能。相反, 稠密结果保留其原始秩, 而仅 BM25 的结果被迫追加到末尾:

$$\mathcal{R}(q) = \mathcal{D}(q) \cup (\mathcal{K}(q) \setminus \mathcal{D}(q)). \quad (1)$$

金字塔检索。上述混合搜索生成一个候选集 $\mathcal{R}(q)$, 每个候选在稠密检索过程中仅通过余弦相似度 $\text{sim}(q, \mathcal{M}_i) = \mathbf{e}_q^\top \mathbf{e}_i$ 得分一次。金字塔机制随后在三个层级上逐步扩展这些候选的内容, 并复用该得分来控制每次转移: **第 1 层**仅返回摘要 (每个 ~ 10 个 token) 给相似度最高的 k 个候选; **第 2 层**加载全文或详细描述, 条件是候选的相似度超过阈值 θ ; **第 3 层**在显式的 token 预算 B 下, 从冷存储中加载原始内容 (图像、音频), 按每 token 相似度递减顺序贪婪扩展项目。所有转移均由确定性规则控制, 而非 LLM 判断, 从而避免额外延迟, 同时根据每个查询的复杂度自适应调整上下文深度。

3.2.3 知识图谱增强型检索

虽然混合搜索和金字塔检索能够处理可以从单个 MAU 中回答的查询, 但许多现实世界的查询需要对多个关联的事实进行推理 (例如, “我在三月份参加完会议后送给那个人什么礼物?”)。因此, 第三个原则是**结构化知识**: OmniMem 维护一个知识图谱 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, 用于捕捉所有 MAU 中的实体和关系。

在创建 MAU 的过程中, 大语言模型 (LLM) 从每个摘要中提取实体和有向关系, 生成实体-关系三元组。每个实体均带有来自 7 个类别 (人物、地点、事件、概念、时间、组织、物体) 的类型标签, 并与对应的源 MAU 相链接。当新的 MAU 被引入时, 同一个真实世界实体可能以不同的表面形式出现 (例如, “Dr. Smith” 与 “John Smith”)。为防止结点碎片化, 实体消解会将那些混合相似度超过阈值的实体进行合并, 其中混合相似度结合了名称嵌入的余弦相似度与 Jaro-Winkler 字符串相似度。

在查询时, 系统识别查询中提到的种子实体 $\mathcal{V}_q \subset \mathcal{V}$, 并在 h 跳内执行有界邻域扩展。每个到达的实体根据距离衰减的相关性进行打分 $r_G(v) = \beta^{d(v, \mathcal{V}_q)} \cdot \text{conf}(v)$, 其中 $d(v, \mathcal{V}_q)$ 是到任意种子实体的最短路径距离, $\beta \in (0, 1)$ 是衰减因子。与高得分图实体相关的 MAU 将与来自 $\mathcal{R}(q)$ 的混合搜索结果合并, 为答案生成提供直接内容匹配和关联连接的证据。

3.3 特定基准的最优化

在描述了流水线从 SimpleMem 中发现的多模态架构后，我们现在转向其如何针对每个目标基准优化该架构。该流水线采用两阶段策略：在小规模训练子集上进行快速迭代，随后在保留的测试集上进行评估。

用于快速迭代的开发子集。对于每个基准，流水线会选择一个小规模代表性子集，以便在最优化环中进行快速实验。在 LoCoMo 上，使用少量对话的子集进行迭代开发，使得每次实验可在 2 小时内完成。在 Mem-Gallery 上，使用少量数据集的子集，每次实验可在几分钟内完成。这种设计使流水线能够在数天内探索数十个假设。在最优化轨迹收敛后，最终配置将在完整的基准上进行评估，以确保泛化能力，并保持与先前记忆系统所使用的评估协议的一致性。

迭代诊断与修复。在每个最优化周期中，流水线自主地在两个层面诊断并修复故障。在执行层面，当实验失败或产生意外输出时，自愈模块会分类错误（API 错误、依赖错误、运行时异常、输出格式不匹配），并生成针对性的修复方案。例如，当嵌入服务因过期的 API 密钥返回 403 错误时，该模块检测到认证失败模式，并自动切换至本地 sentence-transformer 后端，无需人工干预。在语义层面，当实验成功但产生意外的低性能指标时，流水线会进行更深入的分析。

4 实验

我们从两个维度对 OmniMem 进行评估：(1) 自主最优化过程，具体而言，流水线是否在多种基准上发现了有意义的改进；(2) 最终系统质量，考察所发现的架构是否达到了前沿水平，以及各个组件是否作出了有意义的贡献。

4.1 实验设置

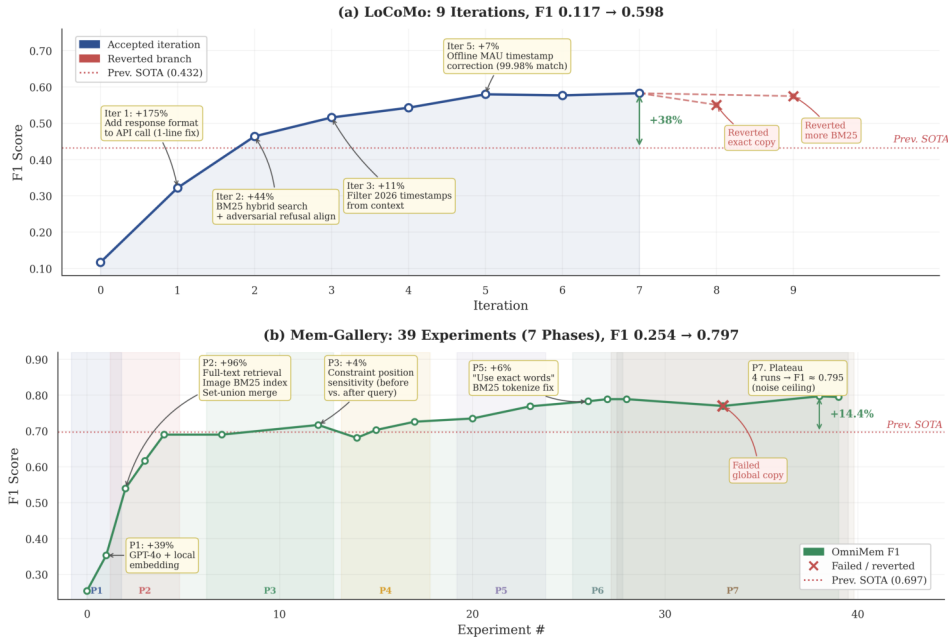


图 3: LoCoMo 上的最优化轨迹（上方，9 次迭代）和 Mem-Gallery（下方，7 个阶段共 39 次实验）。实线表示接受的迭代；失败/回滚的实验用 × 标记。虚线红色曲线表示之前的 SOTA。每个阶段的关键发现已标注。

基准。我们在两个涵盖不同类型记忆依赖推理的基准上进行评估: LoCoMo (Maharana et al., 2024)(1,986 QA pairs across multi-session dialogues, token-level F1) 和 Mem-Gallery (Bei et al., 2026)(1,711 QA pairs from 240 multimodal dialogues with 1,003 grounded images, F1)。详细的基准描述见附录 A。

基准。我们与六种代表不同设计哲学的内存系统进行对比: 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), 以及 SimpleMem (Liu et al., 2026a)(efficient life-long memory with atomization and adaptive pruning)。所有系统均在相同的划分和协议下进行评估。

实现细节。稠密检索使用 FAISS 与 all-MiniLM-L6-v2 嵌入 (384d); 稀疏检索使用 BM25; 视觉新颖性过滤使用冻结的 CLIP ViT-B/32。知识图谱抽取使用 GPT-4o 的 JSON 模式。默认配置: $\text{top-}k=20$, $\theta=0.4$, $B=6,000$ token。完整的基准配置详情见附录 C。

4.2 最优化轨迹

图 3 展示了最优化轨迹。该流水线在 ~ 72 小时的实时时间内完成了两个基准上的 ~ 50 项实验, 这一覆盖范围若由人类研究人员以每天 ~ 3 项实验的速度完成, 大约需要 4 周时间。我们重点展示了最具影响力的发现; 详细的每迭代表格见附录 F。

LoCoMo (9 次迭代, F1: $0.117 \rightarrow 0.598$)。该流水线在 48 小时内成功执行了 9 次迭代, 另有 2 次实验自动回滚 (完整轨迹见附录表 6)。影响最大的发现 (第 1 次迭代, +175%) 是识别出 API 调用缺少 `response_format` 参数, 这一行代码错误导致了 $9\times$ 的冗余, 严重破坏了 F1 准确率。在第 5 次迭代中, 流水线发现所有 4,277 个 MAU 时间戳均被错误地统一为摄入日期, 并自主生成了一个关键词匹配脚本, 无需重新摄入即可纠正其中 99.98% 的数据。此外, 流水线还发现对 FAISS 与 BM25 结果进行集合合并 (第 2 次迭代) 显著优于基于得分的融合方法, 该发现通过消融实验得到了验证 (第 4.4.1 节)。

Mem-Gallery (39 次实验, F1: $0.254 \rightarrow 0.797$)。最优化共经历 7 个阶段 (完整轨迹见附录表 7)。单次最大提升 (+53%) 来自发现: 返回完整的原始对话文本而非由大语言模型生成的摘要, 能显著提升 token 重叠 F1, 这一发现并不明显, 因为传统上摘要更受青睐以提高效率。该流水线还发现, 提示约束的位置 (问题之前与之后) 比约束内容的影响更大, 仅此一项改动便使某一类别性能提升了 +188%。在第 7 阶段后, 四次独立运行得到的 F1 值范围为 $[0.791, 0.797]$, 确认了性能上限, 触发流水线决定停止。

4.3 主要结果

为了将这些结果与现有的记忆系统进行对比, 我们在五个 LLM 骨干网络 (GPT-4o、GPT-4o-mini、GPT-4.1-nano、GPT-5.1 和 GPT-5-nano) 上对 OmniMem 与六种基准方法进行了受控比较。表 1 报告了在所有骨干网络上, LoCoMo 的各类别 F1 分数以及 Mem-Gallery 上的五项评估指标。

在 LoCoMo 上, OmniMem 在所有骨干模型中实现了最高的总体 F1 分数, 范围从 GPT-4.1-nano 的 0.492 到 GPT-5.1 的 0.613, 显著优于当前 LoCoMo 上的最先进方法 SimpleMem

表 1: 在五个 LLM 骨干模型上对 LoCoMo (左) 和 Mem-Gallery (右) 的比较。LoCoMo 列: MH = 多跳, SH = 单跳, Tmp = 时间, Open = 开放领域, Adv = 对抗, All = 整体 F1。Mem-Gallery 列: F1, EM = 确切匹配, B/B-1/B-2 = BLEU/BLEU-1/BLEU-2。最佳基准 (不包括 OmniMem) 为下划线; 最佳整体结果为加粗。

Backbone	Method	LoCoMo						Mem-Gallery				
		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	<u>0.915</u>	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	<u>0.246</u>	0.305	0.843	0.404	0.435	0.298	0.275	0.390	0.335
	SimpleMem	<u>0.318</u>	<u>0.195</u>	0.235	<u>0.308</u>	0.802	<u>0.432</u>	<u>0.535</u>	<u>0.348</u>	<u>0.310</u>	<u>0.468</u>	<u>0.390</u>
	OmniMem	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	<u>0.179</u>	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	<u>0.845</u>	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	<u>0.300</u>	<u>0.128</u>	0.178	<u>0.312</u>	0.891	<u>0.404</u>	<u>0.498</u>	<u>0.318</u>	<u>0.290</u>	<u>0.435</u>	<u>0.368</u>
	OmniMem	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	<u>0.234</u>	0.279	0.556	0.316	0.360	0.238	0.218	0.318	0.268
	SimpleMem	<u>0.298</u>	<u>0.145</u>	0.210	<u>0.285</u>	0.648	<u>0.342</u>	<u>0.518</u>	<u>0.338</u>	<u>0.300</u>	<u>0.452</u>	<u>0.380</u>
	OmniMem	0.477	0.216	0.244	0.583	<u>0.722</u>	0.492	0.780	0.430	0.353	0.610	0.488
GPT-5.1	MemVerse	0.287	0.173	<u>0.277</u>	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	<u>0.819</u>	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	<u>0.305</u>	<u>0.178</u>	0.272	<u>0.305</u>	0.807	<u>0.418</u>	<u>0.538</u>	<u>0.350</u>	<u>0.312</u>	<u>0.470</u>	<u>0.395</u>
	OmniMem	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	<u>0.203</u>	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	<u>0.745</u>	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	<u>0.278</u>	0.200	<u>0.245</u>	<u>0.282</u>	0.824	<u>0.388</u>	<u>0.522</u>	<u>0.340</u>	<u>0.302</u>	<u>0.458</u>	<u>0.385</u>
	OmniMem	0.357	0.371	0.253	0.561	0.719	0.522	0.787	0.437	0.357	0.617	0.494

(0.342–0.432) (Liu et al., 2026a)。OmniMem 在多跳、单跳和开放领域类别上均占主导地位, 尤其在开放领域问题上优势明显。

在 Mem-Gallery 上, OmniMem 的 F1 值范围为 0.749 至 0.810, 始终以显著优势超越所有记忆基准。SimpleMem 再次成为最强的基准 (使用 GPT-5.1 时 F1 最高达到 0.538), 但仍比 OmniMem 低超过 25 个百分点。这些模式证实, OmniMem 的提升源于其架构设计 (混合搜索、金字塔检索、知识图谱增强), 而非单一主导组件。

4.4 分析

4.4.1 消融研究

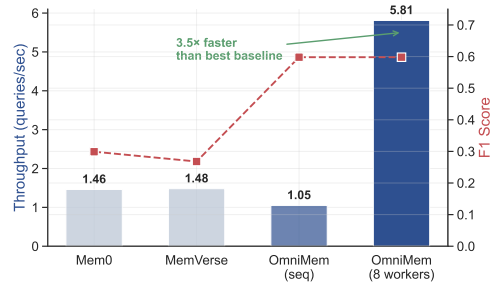
表 2: LoCoMo 上的组件消融实验 (在 4 个主干网络上的平均 $\Delta F1 \times 100$)。

表 2 展示了针对 LoCoMo 的消融实验，验证了流水线发现的关键设计选择。具体而言，我们逐个移除各个组件，并报告了在 4 个大语言模型骨干网络上的平均 $\Delta F1$ 值。金字塔扩展 (−17%) 和 BM25 混合搜索 (−14%) 是最关键的两项，且均在最优化过程中通过自主流水线得到了显著改进。大语言模型摘要贡献了 −12%，证实了紧凑的 MAU 摘要对检索质量至关重要。将 top-k 从 20 降低至 5 导致性能下降 −7%，而元数据上下文的影响较小 (−2%)。值得注意的是，影响最大的两个组件 (金字塔扩展和混合搜索) 恰好也是获得最多优化迭代次数的组件，表明流水线正确地分配了其搜索预算。

Component Removed	$\Delta F1$	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 效率

OmniMem 在使用 8 个并行工作进程时实现了 5.81 次查询/秒 (比最快的基准快 3.5×)，这得益于只读的 FAISS 和 BM25 索引支持并发查找 (图 4, 表 3)。所有基准方法均受限于顺序的 LLM 生成过程 (占每次查询时间的 85–97%)，而 OmniMem 通过线程安全的只读索引并行化了检索-生成流水线。



4.4.3 案例研究：多跳检索

我们以 LoCoMo 中的一个真实多跳查询为例，展示 OmniMem 的检索流水线。该查询要求跨分离的对话会话合成事实。查询内容为：“Caroline 和 Melanie 都画过什么主题？”正确答案是“sunsets”，但要正确回答，需要从不同会话中分别检索每个人的艺术创作历史，并识别出其中的重叠部分。

混合搜索。稠密检索返回提及卡罗琳画作的 MAU (例如，“卡罗琳画了一幅日落场景”) 和梅兰妮艺术项目的 MAU (例如，“梅兰和她的孩子们用棕榈树画了一幅日落”)，但这些结果出现在不同的会话中，且上下文不同。BM25 检索到更多包含关键词“画”的 MAU，这些在稠密检索中的排名较低。集合并集合并保留了稠密排序，并追加了仅在 BM25 中出现的结果。

知识图谱扩展。查询处理器提取种子实体 Caroline (人物) 和 Melanie (人物)。邻域扩展通过分离的关系路径将这两个实体分别链接到 绘画 (概念) 和 日落 (概念)，揭示了提及每个人绘画活动的多文档摘要，即使表面文本未同时提及两个人的名字。

金字塔检索与回答。从两条关系路径中加载一级摘要；其相似度得分超过 θ ，触发完整对话文本的二级扩展。大语言模型识别出“日落”为共同主题，并生成了正确答案 ($F1 = 1.0$)。相比之下，由于缺少跨会话实体链接，MemGPT 产生了幻觉“马” ($F1 = 0.0$)。

表 3: 延迟分解。

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

Ret./Gen. in milliseconds.

5 结论

我们提出了 OmniMem, 这是一个统一的多模态记忆框架, 其架构和配置通过 AutoResearchClaw 自主研究流水线发现。从一个简单的基准 (LoCoMo 上的 $F1 = 0.117$) 开始, 该流水线在 ~ 72 小时内自主执行了 ~ 50 次实验, 在 LoCoMo 和 Mem-Gallery 上均达到了最先进水平。最具影响力的发现包括错误修复 (+175%)、架构变更 (+44%) 以及提示工程 (特定类别上 +188%), 这些发现需要代码理解与跨组件推理, 超出了传统 AutoML 的能力范围。我们提出的六类发现分类法, 结合观察到多模态记忆由于其标量指标、模块化架构和快速迭代周期而非常适合自研究, 为将自主研究流水线应用于其他复杂 AI 系统领域提供了路线图。

伦理声明

OmniMem 在长时间范围内持续保留多模态个人数据 (文本、图像、音频、视频), 这引发了关于数据隐私与同意的重要考量。在现实世界中部署时, 必须获得用户的明确同意, 具备透明的数据保留策略, 并建立强有力的机制以审查、修改或删除存储的记忆。知识图谱组件能够跨记忆单元提取并关联实体, 若未配备充分的访问控制和加密措施, 可能被用于用户画像。关于自主研究流水线, 所有 AutoResearchClaw 实验均在已确立的学术基准上进行, 未使用真实用户数据; 将此类流水线部署至生产系统则需额外监督。我们主张采用增强记忆的智能体, 优先保障用户自主权、数据最小化以及被遗忘的权利。

可复现性声明

我们提供了完整的超参数 (表 4)、各基准的配置 (表 5) 以及系统架构细节 (附录 C)。完整的最优化轨迹, 包括每迭代的度量指标和决策, 均记录在附录 E 中。所有提示均原文复现于附录 G, 而 AutoResearchClaw 流水线的各个阶段在附录 D 中有详细描述。两个基准 LoCoMo (Maharana et al., 2024) 和 Mem-Gallery (Bei et al., 2026) 均为公开可用; 我们采用与先前工作相同的划分和协议进行评估, 并使用标准指标 (token 级别 F1、确切匹配、BLEU)。我们开源了完整的框架、基准测试工具包以及全部实验日志, 以支持独立复现。

参考文献

- 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 hyperparameter 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 基准详情

洛科莫。 长期对话记忆基准 (Maharana et al., 2024) 评估智能体在跨多会话对话中回忆和推理的能力。该基准包含 10 个对话，每个对话有 19-32 个会话（每条对话约 ~9 K token），共生成 1,986 个问答对，涵盖五个类别：**单跳** (SH) 需要检索单一事实；**多跳** (MH) 需要在多个会话间整合信息；**时间性** (T) 测试对事件发生时间的推理能力；**开放式** (O) 要求生成更长且具有上下文关联的回答；以及 **对抗性** (A) 测试正确拒绝无法回答问题的能力。评估采用基于词干提取的 token 级 F1 值。

记忆画廊。 Mem-Gallery (Bei et al., 2026) 评估社交互动中的多模态长期记忆，包含来自 240 次多轮对话的 1,711 组问答对，涉及 1,003 张有场景依据的图像和 3,962 轮对话。问题涵盖 9 个类别：动作识别 (AR)、复合分解 (CD)、视觉搜索 (VS)、时间线学习 (TTL)、时间推理 (TR)、事实检索 (FR)、视觉推理 (VR)、知识推理 (KR) 和多实体推理 (MR)。我们使用 Huggingface 提供的公开 20 数据集子集。

B 基准描述

MemVerse (Liu et al., 2025) 将情景记忆层与语义记忆层结合，构建多模态知识图谱。每个摄入项需要 3 次 LLM 调用（摘要生成、实体抽取、关系链接），导致摄入速率为 0.22 项/秒。检索基于知识图谱上的嵌入相似度实现。

Mem0 (Chhikara et al., 2025) 从对话输入中执行动态事实提取，将结构化事实存储，并可选地进行图内存增强。其数据摄入速率达每秒 1.28 项，搜索延迟为 18 毫秒，但本质上仅支持文本。

简单内存 (Liu et al., 2026a) 通过三种核心机制实现了高效的终身记忆管理：记忆原子化（将输入分解为细粒度的原子单元）、自适应合并（合并相关记忆以减少冗余）和上下文感知剪枝。它在 LoCoMo 上达到了顶尖性能，同时显著降低了 token 开销（~ 比 Mem0 减少 45%），总处理时间快了 4×。

Claude-Mem (Anthropic, 2024) 提供商业级基于嵌入的对话记忆，支持每轮对话的图像存储。

A-MEM (Xu et al., 2025a) 引入了 Agentic 记忆管理，其中大语言模型 (LLM) 自身决定何时以及如何重新组织存储的记忆，包括对记忆条目进行合并、拆分和抽象等操作，以保持紧凑且相关的状态。

MemGPT (Packer et al., 2023) 将其类比于操作系统的内存层次结构，实现了主上下文（类比于内存）和外部存储（类比于磁盘），并通过由大语言模型智能体管理的显式 push/pop 操作来实现。它在对话任务上表现出色，但本质上仅限于文本，并且在每个内存管理决策上受限于大语言模型的串行调用。

C OmniMem 技术细节

C.1 系统架构

完整的 OmniMem 实现包含跨 11 个子包的 13,300 行 Python 代码：

- **核心** (MAU 数据结构, 事件层次, 统一配置)
- **处理器** (模态特定的输入: 文本、图像、音频、视频)
- **存储** (三层持久化: MAU 存储为 JSON Lines, FAISS 向量存储, 文件系统/S3 上的冷存储)
- **检索** (金字塔检索器, 查询处理器, 扩展管理器)
- **知识** (实体提取器, 内存知识图谱, 图检索器)
- **编排器** (中心协调器, 873 行, 统一 API)

C.2 超参数和基准特定配置

表 4 列出了完整的超参数配置。表 5 总结了流水线发现的每个基准的配置。

表 4: OmniMem 的超参数设定。

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

表 5: 由自主流水线发现的各基准配置 ([†] 随问题类别变化)。

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

C.3 知识图谱模式

知识图谱使用 7 种实体类型 (Person, Location, Object, Event, Concept, Time, Organization) 和 7 种关系类型 (located_in, part_of, interacts_with, owns, attended_by, related_to)。在创建 MAU 时, 通过 GPT-4o 以 JSON 模式执行实体抽取。

D AutoResearchClaw 流水线细节

整个流水线包含 23 个阶段，分为 8 个阶段：

- A. **研究范围界定** (第 1-2 阶段)：制定 SMART 研究目标；自动检测硬件（GPU 型号、显存容量）。
- B. **文献发现** (第 3-6 阶段)：查询 OpenAlex、Semantic Scholar、arXiv；根据相关性/质量进行筛选（第 5 阶段设关卡）；提取结构化知识卡片。
- C. **知识融合** (阶段 7-8)：对发现进行聚类；通过多智能体辩论生成可验证的假设。
- D. **实验设计** (阶段 9-11)：设计协议（阶段 9 的门控）；生成具有 AST 验证的硬件感知 Python 代码；调度资源。
- E. **实验执行** (阶段 12-13)：在沙箱/ Docker/ SSH 中运行实验；具有结构化缺陷分类的自愈环（最多 10 次重试）。
- F. **分析与决策** (阶段 14-15)：统计分析（t 检验，自助法置信区间）；自主做出继续前进/调整方向/迭代优化的决策。
- G. **文档编写** (阶段 16-19)：生成大纲；撰写初稿；模拟同行评审；修订。
- H. **最终确定** (阶段 20-23)：质量检查点（阶段 20）；知识归档；LaTeX 导出；四层引用验证（arXiv 编号、DOI、标题匹配、相关性得分）。

E 最优化轨迹详情

表 6: LoCoMo 上的最优化轨迹。该流水线自主发现了 7 次连续改进，并正确回滚了 2 次失败的实验。

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		

F 完整迭代日志

F.1 LoCoMo：各类别在迭代过程中的性能表现

表 8 显示了每个类别在最优化轨迹中 F1 的演变情况。

显著模式：类别 5（对抗）在第 2 轮对齐评估标准的拒绝语句后，从 0.447 跃升至 1.000。类别 2（时间）表现出最显著的提升（总提升 +0.496），主要由第 5 轮的时间戳修正驱动。类

表 7: Mem-Gallery 上的最优化轨迹 (39 次实验, 7 个阶段)。每个阶段代表流水线最优化策略的定性转变。

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		

表 8: LoCoMo 按类别划分的 F1 值随迭代变化情况 (conv-26, 199 个问答对)。

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

别 3 (开放式) 是我们最强的类别, 相较于先前的最优方法 (SOTA), 超越了 SimpleMem 的 +0.200。

F.2 Mem-Gallery: 详细分阶段日志

第一阶段: 环境设置 (实验-000 至 001)。 从使用 gpt-4.1-nano 和 CLIP-B/32 512-dim 嵌入的 $F1 = 0.254$ 开始, 流水线在遇到 API 代理错误后, 首先将大语言模型升级为 gpt-4o, 并切换到本地的 all-MiniLM-L6-v2 嵌入 (384d)。结果: $F1 = 0.353$ (提升 39%)。

阶段 2: 架构突破 (实验-002 至 004)。 最大的单方面改进 (+53%) 来自于返回完整的原始对话文本, 而非大语言模型的摘要, 这与传统上更偏好摘要的观点相悖。流水线进一步引入了 BM25 混合搜索 (集合并集合并) 并为视觉问题类别创建了专用的图像-标题 BM25 索引。综合结果: $F1 = 0.690$ (+96%)。

第三阶段: 微调 (实验-004b 至 012)。 流水线发现, 格式约束 position (问题前与问题后) 在温度 =0 时比内容更重要。仅通过重新定位, KR 类别提升了 +188%。若干提示调优实验 (实验-008 至 011) 失败并已回滚。最终结果: $F1 = 0.717$ (+4%)。

第四阶段: 规模验证 (实验 014 至 018)。 在 14 个数据集上的验证暴露了数据完整性问题 (store_only 在 qa_only 之前未完成)。流水线学成 数据完整性 > 算法最优化。结果: $F1 = 0.726$ (+1%)。

第五阶段：确切引用（实验 020 至 023）。 两个关键发现：(1) “使用对话中的确切词语和短语”指令在 12/15 个数据集上提升了性能（F1 提升 +0.031）；(2) 一个简单的 BM25 tokenization 修复（去除标点符号：“sushi.” → “sushi”）带来了 +0.018 的 F1 提升，超过 10 轮的提示工程。结果：F1=0.771（提升 +6%）。

第六阶段：视觉推理能力增强（实验 026 至 027）。 通过为每张图像添加对话上下文（full_text[:300]）扩充图像目录，使 VR 类别提升 +0.087。为 TR/CD 类别增加时间顺序信息，进一步贡献了 +0.006 的提升。结果：F1=0.789（+2%）。

第七阶段：平台探索（Exp-028 至 039b）。 流水线探索了模型对比（gpt-4.1 ≈ gpt-4o）、按类别提示优化以及多种 BM25 配置。经过 4 次独立运行，F1 值在 [0.791, 0.797] 之间，流水线正确识别出性能上限（~0.795 由于随机噪声），并终止。最终峰值：F1=0.797。

G 提示目录

本节记录了在 OmniMem 中使用的代表性提示，按系统组件进行组织。所有提示均来自源代码；为提高可读性，仅做了轻微的格式调整。

G.1 核心系统提示

G.1.1 答案生成

编排器在生成答案时采用两部分提示结构：系统提示定义了助手的角色，用户提示则注入检索到的上下文并包含结构化输出要求。这种设计强制使用 JSON 模式输出，以确保答案提取的可靠性。

Answer Generation – System Prompt

您是一位专业的问答助手。您的任务是从提供的记忆上下文中提取简洁、准确的答案。在可能的情况下，应根据上下文进行合理推断。您必须输出有效的 JSON 格式。

Answer Generation – User Prompt Template

基于这些记忆：

{上下文}

问题: {问题}

要求：

首先，仔细思考推理过程。

确切的答案，简洁明了。

3. 根据所提供的信息作答。您可以基于给出的信息进行合理推断。

4. 尽你最大努力作答。如果上下文完全不包含与所问主题相关的任何信息，则仅回答 “unknown”。

5. 对于计数问题，只回答数字（例如，‘2’ 而不是 ‘两次’）。

对于是非题，请以 “是”、“否”、“很可能是” 或 “很可能否” 开头。

7. 列出多个项目时，用逗号分隔。

8. Return your response in JSON format.

输出格式：

{“reasoning”: ” 简要解释”, “answer”: ” 简洁的答案”}

LLM 调用时设置 `temperature=0.1` 且 `response_format=json_object`，以确保输出具有确定性且可解析。结构化的 JSON 输出是自主流水线的一项关键发现：在未强制格式的情况下，模型生成了冗长的自然语言回答，导致 F1 分数下降了 175%（参见迭代 1，表 6）。

G.1.2 实体与关系抽取

知识图谱模块从每个 MAU 摘要中提取带类型的实体和有向关系。提示强制采用结构化的 JSON 模式并附带置信度得分，从而使得下游的实体消解能够过滤低置信度的抽取结果。

Entity Extraction – System Prompt

您是一个专业的知识提取系统。从给定文本中提取实体和关系。

输出 JSON 格式：

```
{
  "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
    }
  ]
}
```

指南：

- 提取所有提及的有意义的实体
- 识别实体之间的关系
- 使用标准实体类型
- 根据清晰度分配置信度得分
- 对于视觉描述，要注意空间关系和物体属性

Entity Extraction – User Prompt

从这段文本中抽取实体和关系：

{文本}

选取了 7 种实体类型（人物、地点、物体、事件、概念、时间、组织），以涵盖两个基准中观察到的主要类别。多模态指南（“对于视觉描述，关注空间关系”）使得相同的抽取提示能够在任何输入模态生成的文本摘要上统一运行。

G.1.3 查询意图分析

在检索之前，查询处理器可选地分析查询意图，以提取结构化元数据。此分析将用于指导检索策略的选择（例如，对于时间相关查询启用时间排序，或对图像相关查询激活视觉搜索）。

Query Analysis – System Prompt

您是一名查询分析助手。仅输出有效 JSON。

Query Analysis – User Prompt Template

分析此内存检索查询并提取：

1. intent_type: 以下之一 [factual, temporal, comparative, exploratory, verification]
2. 实体: 命名实体列表 (人物、地点、事物)
3. time_references: 任何时间表达 (昨天, 上周等)
4. modality_hints: 可能的内容类型 [视觉, 音频, 文本, 视频]
5. reformulated_query: 优化的搜索查询

查询: {query}

仅以 JSON 格式回复, 无需解释。

G.2 模态处理器提示

OmniMem 在摄入阶段采用模态特定的提示, 将异构输入转换为统一的 MAU 文本摘要。一个关键的设计原则是提示极简主义: 简洁的指令在摄入时产生更短、更利于检索的摘要, 而详细的提示则保留用于回答生成时的按需扩展。

G.2.1 图像描述

两层图像描述生成服务于金字塔检索流水线的不同阶段:

Concise Image Caption (Ingestion – Level 1)

用一句话简洁描述此图像。重点关注关键物体、动作和上下文。

Detailed Image Caption (Expansion – Level 3)

请详细描述此图像, 包括:

1. 主要人物及其外貌
2. 发生的动作或事件
3. 情景与环境
4. 著名天体
5. 所有可见文本

简洁变体在摄入阶段使用 ($\text{max_tokens}=150$), 生成存储在热存储中的 ~ 10 -token 摘要。详细变体仅在三级金字塔扩展时调用 ($\text{max_tokens}=500$), 按需从冷存储加载更丰富的描述。这种两级设计在降低存储成本的同时, 保留了在处理复杂查询时访问细粒度视觉细节的能力。

G.2.2 音视频摘要

音频内容首先通过语音转文字进行转录, 然后进行总结:

Audio Transcript Summary

用一句话简洁概括这段音频内容：

{transcript[:2000]}

概要：

视频月活跃用户将视觉帧描述与音频字幕整合为统一摘要：

Video Content Summary

用 1-2 句话总结这段视频的内容：

{frame_ 描述}

{音频转录}

视频摘要：

两个提示均遵循极简主义原则：在金字塔第一层，仅需单句摘要即可满足基于嵌入的检索需求，而原始转录文本和帧描述则保留在冷存储中，以供深度扩展。

G.3 Mem-Gallery 基准提示

Mem-Gallery 评估采用通过自主最优化流水线发现的多层级提示架构。全局系统提示定义了任务上下文，特定类别的格式约束控制输出格式，而对话智能体提示则定义了每个问题。该流水线的关键发现是，约束的位置（问题之前或之后）在低温条件下对性能有显著影响，通过按类别优化，单个类别性能最高可提升 +188%（第 3.3 节）。

G.3.1 系统提示

Mem-Gallery System Prompt

您是一位在多模态长期对话记忆方面进行评估的人工智能助手。对于给定的问答任务，您的回答必须简洁，但需完整准确地回答问题。如果对话中关于同一事件出现多个信息，请始终以最新信息为准。

问答评估将包含多种多模态任务类型：

事实检索：检索对话中提到的用于回答的明确事实。

多实体推理：结合检索到的信息进行推理并得出答案。

时间推理：解决与时间相关的问题。

视觉中心推理：除了文本信息外，还应使用对话中的视觉图像来回答问题。

测试时学习：从历史对话中提供的图像中学习新的视觉知识，并在问答中使用。

以视觉为中心的查询：找到与给定查询信息匹配的图像，并返回其图像 ID。

冲突检测：检测对话历史与问题中提供的信息之间的矛盾。

知识消解：通过优先考虑最新信息来解决知识冲突或更新。

回答拒绝：当对话历史中不存在相关信息时，拒绝回答。

严格遵循所有指令。仅使用多模态对话中包含的信息进行回答。不要编造信息。始终保持与对话历史的一致性和真实性。

G.3.2 特定类别格式约束

九个类别中有三个类别在问题后附加了明确的格式约束。这些约束通过流水线迭代优化，以使模型输出与评估指标保持一致：

Answer Refusal (AR) Constraint

请根据对话中的信息提供您的答案。如果对话中没有关于该问题的信息，请回答：“未提及。”

Conflict Detection (CD) Constraint

请检查此信息是否与对话冲突，严格回复 “Yes.” 或 “No.”

Visual Search (VS) Constraint

返回图像的 `image_id`。如果有多个图像，请按升序排序，并用逗号分隔。格式示例：“D2:IMG_003, D2:IMG_010, D10:IMG_002”（仅作参考）。

其余六个类别（FR、MR、TR、VR、TTL、KR）使用默认系统提示，不附加格式约束。流水线发现，向这些类别添加约束要么没有效果，要么因过度指定而降低性能。

G.3.3 对话智能体提示

每个问题均由一个对话智能体提示语来构建，该提示语附加在检索到的记忆上下文之后：

Mem-Gallery Dialogue Agent Prompt

请根据对话内容及记忆信息，以简洁的方式回答关于 `{speaker_a}` 和 `{speaker_b}` 之间对话的问题。请仅提供答案内容，不要包含“答案：”之类的引导语。对于需要回答日期或时间的问题，请严格遵循格式要求，尽可能提供具体的日期或时间。生成的答案应尽量简洁，但需完整准确地回答问题。

当前的问题如下：

`{观测}` `{格式约束}`

G.3.4 推理模型适配

在使用倾向于生成冗长思维链输出的推理类模型（例如 o1、o3）时，流水线会在系统提示中附加一个简洁性增强指令。这一优化是在测试不同骨干模型的第 7 阶段发现的：

Reasoning Model Conciseness Boost

重要规则 — 答案简洁性：

你将根据短参考答案进行逐 token 的 F1 评估。冗长的回答会严重降低你的得分。请严格遵守以下规则，不得有任何例外：

1. 仅输出必要答案——无需推理、无需解释、无需重述问题。
2. 对于是非题 → 回答只能是“是。”或“否。”。不要添加解释。
3. 对于事实类问题 → 仅以简明的事实作答。
✓ “马耳他犬” ✗ “图像中的狗看起来像是一只马耳他犬。”
4. 对于列表类问题 → 以逗号分隔的项目，仅此而已。
5. 切勿使用项目符号、编号列表或多行格式。
6. 绝不要以“答案是”、“根据对话”等开头。

G.3.5 记忆上下文注入

检索到的记忆将按照以下模板进行格式化并注入提示中。对于多模态记忆，图像内容与文本上下文以 Base64 编码的内容片段交替方式插入：

Memory Context Template

检索到的内存内容如下：
{记忆上下文}

Multimodal Memory Template

{文本上下文}
图片：
image_id: {image_id}
image_content: [base64 编码的图像]