

Agent Memory 核心论文解读

10篇必读论文 | 从入门到精通 AI Agent 记忆系统

📖 阅读指南

两大流派

🎓 模型驱动派（算法岗推荐）：

- 改造模型内部结构
- 从底层嵌入记忆能力
- 适合做算法创新、发论文

🔧 应用驱动派（开发岗推荐）：

- 在应用层构建记忆系统
- 不改模型，加"外挂"
- 适合做工程落地、快速验证

阅读顺序建议

开发岗（快速上手）：

1. MemGPT（应用驱动的代表）
↓
2. Mem0（生产级实现）
↓
3. Zep（时序知识图谱）
↓
4. Agent Memory Survey（全面了解）

算法岗（深入研究）：

1. Agent Memory Survey（建立框架）
↓
2. Memorizing Transformers（模型驱动的开山之作）
↓
3. MemoryLLM（可更新记忆）
↓

4. MemGPT（对比应用驱动）



5. 其他前沿论文

🎓 Part 1: 应用驱动派（5篇）

1. MemGPT ★★★★★ 必读第一篇

论文标题：MemGPT: Towards LLMs as Operating Systems

发表时间：2023年10月

机构：UC Berkeley

论文链接：<https://arxiv.org/abs/2310.08560>

GitHub：<https://github.com/cpacker/MemGPT> (19k+ Stars)

lettaPublic

Watch140Fork2kStar19.3k

main60 Branches171 Tags

Go to fileAdd fileCode

SootyOwlfix: Implement architecture-specific OTEL installation logic (#3061) ✓d8b3bb8 · yesterday6,578 Commits

.github	feat: parallel tool calling in model settings [LET-6239] (#6...	5 days ago
alembic	fix: fix alembic on main (#6116)	2 weeks ago
assets	chore: Update README.md (#2215)	11 months ago
certs	feat: support local https mode (#2217)	11 months ago
db	chore: migrate package name to letta (#1775)	last year
examples/notebooks/data	chore: remove old examples (#6255)	5 days ago
fern	drop docs and bump version	3 days ago
letta	drop docs and bump version	3 days ago
otel	feat: Ship traces to datadog and add trace correlation (#6...	5 days ago
sandbox	feat: tool function arguments passed in at runtime	3 months ago
scripts	cleanup	7 months ago
tests	test: add sleep before agent deletion in test (#6412)	3 days ago
.dockerignore	fix: patch Dockerfile for purpose of docker run (#2177)	11 months ago
.env.example	feat: various fixes (#2320)	11 months ago
.gitattributes	chore: .gitattributes (#1511)	last year
.gitignore	feat: Write tests for search messages [LET-4212] (#4447)	2 months ago
.pre-commit-config.yaml	chore: migrate to ruff (#4305)	3 months ago
CITATION.cff	fix: Update CITATION.cff (#2009)	last year
CONTRIBUTING.md	feat: uv migration (#3493)	3 months ago

About

Letta is the platform for building stateful agents: open AI with advanced memory that can learn and self-improve over time.

[docs.letta.com/](#)

ai ai-agents llm llm-agent

ReadmeApache-2.0 licenseContributingCite this repositoryActivityCustom properties19.3k stars140 watching2k forksReport repository

Releases169

v0.15.1Latest3 days ago

+ 168 releases

Contributors153

+ 139 contributors

核心思想：

把 LLM 当作操作系统，借鉴操作系统的虚拟内存管理机制：

- 主内存（Main Context）：固定大小的上下文窗口
- 外部存储（External Context）：无限容量的外部数据库

- 页面置换算法：智能决定哪些信息保留在主内存

技术创新：

1. 分层存储：

- Layer 1：工作记忆（当前对话）
- Layer 2：外部存储（历史信息）
- Layer 3：归档存储（长期知识）

2. 自主管理：

- LLM 决定何时 swap in/out
- 基于重要性动态调整
- 类似 OS 的内存管理

适合场景：

- 长期对话（跨会话记忆）
- 复杂任务追踪
- 研究助手

2. Mem0 ★★★★★ 生产级首选

论文标题：Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory

发表时间：2024年4月

论文链接：<https://arxiv.org/html/2504.19413v1>

GitHub：<https://github.com/mem0ai/mem0> (43k+ Stars)



mem0

The Memory Layer for Personalized AI



GITHUB TRENDING

#1 Repository Of The Day

[Learn more](#) · [Join Discord](#) · [Demo](#) · [OpenMemory](#)



downloads 1.9M/month

commit activity 18/month

pypi package v1.0.1

npm v2.1.38

Y Combinator S24



[Building Production-Ready AI Agents with Scalable Long-Term Memory →](#)



+26% Accuracy vs. OpenAI Memory



91% Faster



90% Fewer Tokens



mem0ai v1.0.0 is now available! This major release includes API modernization, improved vector store support, and enhanced GCP integration. [See migration guide →](#)



Research Highlights

- **+26% Accuracy** over OpenAI Memory on the LOCOMO benchmark
- **91% Faster Responses** than full-context, ensuring low-latency at scale
- **90% Lower Token Usage** than full-context, cutting costs without compromise
- [Read the full paper](#)

核心理念：

增强的记忆框架，自动提取和管理关键信息：

- 实体提取 + 关系建模
- 图谱存储 + 向量检索
- 自动更新 + 去重

技术架构：

用户对话



实体识别 (NER)



关系提取 (RE)






知识图谱 (Neo4j) + 向量库 (组合)



检索 (混合：图+向量)

独特优势：

-  开箱即用（10行代码集成）
-  支持多种后端（Redis、Qdrant、PostgreSQL）
-  自动去重和更新

⚠ 注意：

社区反馈有稳定性问题，生产环境使用前需充分测试

适合场景：

- 个性化推荐
 - 对话 Agent
 - 用户画像
-

3. Zep 时序知识图谱

论文标题：Temporal Knowledge Graphs for Agent Memory

发表时间：2025年1月

论文链接：<https://arxiv.org/pdf/2501.13956>

GitHub：<https://github.com/getzep/zep>



Zep: The Memory Foundation For Your AI Stack

Zep: AI 堆栈的内存基础

Examples, Integrations, & More

示例、集成及更多

[Discord](#) [Follow @zep_ai](#)

What is Zep? Zep 是什么?

Zep is a memory platform for AI agents that learns from user interactions and business data. It builds a temporal knowledge graph to provide AI assistants with personalized, accurate, and up-to-date information, enhancing user experiences through continuous learning.

Zep 是一个面向人工智能代理的记忆平台，它能够从用户交互和业务数据中学习。它构建一个时间知识图谱，为人工智能助手提供个性化、准确且最新的信息，并通过持续学习提升用户体验。

How Zep works Zep 的工作原理

1. Add chat messages or data artifacts to Zep during each user interaction or agent event
在每次用户交互或代理事件期间，将聊天消息或数据工件添加到 Zep 中。
2. Zep intelligently integrates new information into the user's Knowledge Graph, updating existing context as needed
Zep 能够智能地将新信息整合到用户的知识图谱中，并根据需要更新现有上下文。

核心思想：

用时序知识图谱管理记忆，捕捉事件的时间关系：

- 节点：实体、事件
- 边：关系 + 时间戳
- 查询：时间范围 + 语义相似度

技术创新：

1. 时间衰减：越久的记忆重要性越低
2. 事件链：追踪事件的因果关系

3. 知识演进：记录信息的更新历史

适合场景：

- 需要追溯历史的场景
 - 事件驱动的应用
 - 长期用户关系管理
-

4. HippoRAG ★★★★★ 神经生物学启发

论文标题：HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models

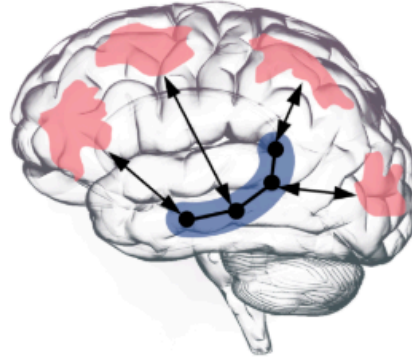
发表时间：2024年5月

机构：OSU、UCLA

论文链接：<https://arxiv.org/abs/2405.14831>

GitHub：<https://github.com/OSU-NLP-Group/HippoRAG>

HippoRAG 2: From RAG to Memory



[Open in Colab](#)

[arXiv 2502.14802 HippoRAG 2](#)

[Dataset HippoRAG 2](#)

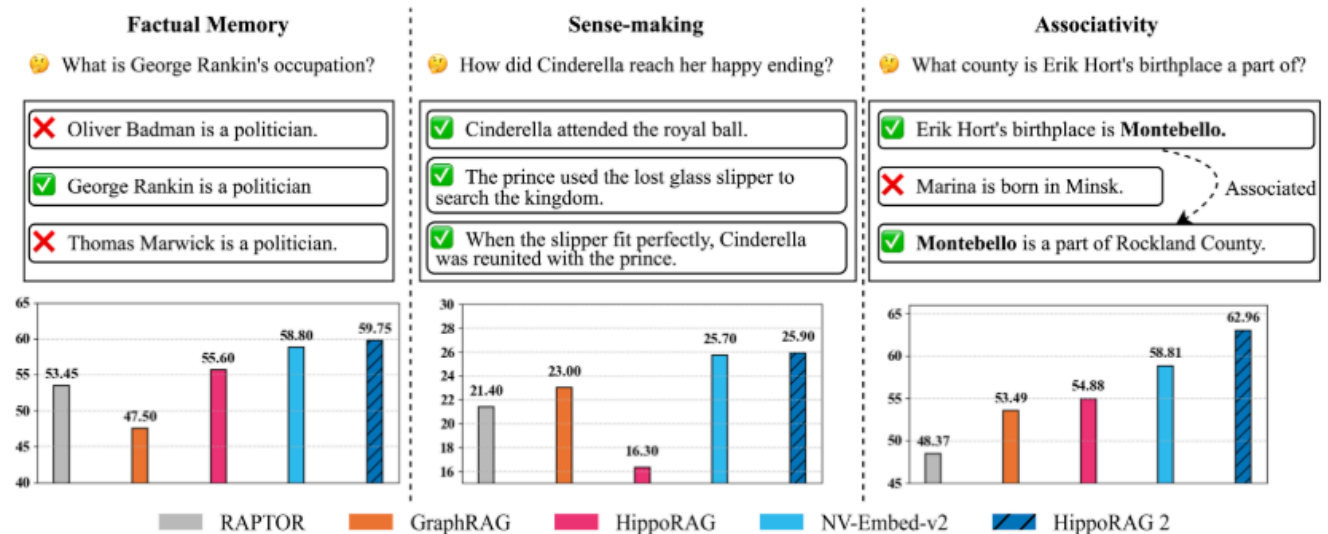
[arXiv 2405.14831 HippoRAG 1](#)

[GitHub HippoRAG 1](#)

HippoRAG 2 is a powerful memory framework for LLMs that enhances their ability to recognize and utilize connections in new knowledge—mirroring a key function of human long-term memory.

Our experiments show that HippoRAG 2 improves associativity (multi-hop retrieval) and sense-making (the process of integrating large and complex contexts) in even the most advanced RAG systems, without sacrificing their performance on simpler tasks.

Like its predecessor, HippoRAG 2 remains cost and latency efficient in online processes, while using significantly fewer resources for offline indexing compared to other graph-based solutions such as GraphRAG, RAPTOR, and LightRAG.



核心思想：

模拟人脑海马体的记忆形成机制：

- 海马索引：快速定位相关记忆
- 新皮层存储：长期知识存储
- 模式分离：区分相似但不同的记忆

技术实现：

- Personalized PageRank (PPR) 模拟海马索引
- 知识图谱存储
- 多跳检索

实验效果：

- 在多跳问答任务上超越传统 RAG 20%+
- 检索准确率显著提升

适合场景：

- 多跳推理
 - 复杂关系理解
-

5. MemOS ★★★★★ 类操作系统架构

论文标题：MemOS: An Operating System for AI Agent Memory

发表时间：2025年5月

论文链接：<https://arxiv.org/abs/2505.22101>

MemOS: Memory Operating System for AI Agents

MemOS is an open-source Agent Memory framework that empowers AI agents with **long-term memory**, **personality consistency**, and **contextual recall**. It enables agents to **remember past interactions**, **learn over time**, and **build evolving identities** across sessions.

Designed for AI companions, role-playing NPCs, and multi-agent systems, MemOS provides a unified API for **memory representation**, **retrieval**, and **update** — making it the foundation for next-generation **memory-augmented AI agents**.



MemOS 1.0: 星河 (Stellar)

[status](#) [Preview](#)[Maintained by](#) [MemTensor](#)[pypi package](#) [v1.1.3](#)[python](#) [3.10](#) | [3.11](#) | [3.12](#) | [3.13](#)[Platform](#) [Linux](#) | [macOS](#) | [Windows](#)[Documentation](#) [view](#)[arXiv](#) [2507.03724](#)[GitHub](#)[Discussions](#)[Discord](#) [join chat](#)[WeChat](#) [Group](#)[/ License](#) [Apache 2.0](#)

MemOS **Free** API Is Now Open

Let Your AI **Remember Users** ■ **Keep Its Persona** ■ **Build Real Relationships**



**Click to Get
Your API key**



Get Free API: [Try API](#)

MemOS: An Operating System for Memory-Augmented Generation (MAG) in Large Language Models (Short Version)

Zhiyu Li*, Shichao Song^{1,3,*}, Hanyu Wang^{1,3,*}, Simin Niu^{3,*}, Ding Chen^{4,*}, Jiawei Yang^{1,3}, Chenyang Xi¹, Huayi Lai³, Jihao Zhao³, Yezhaohui Wang¹, Junpeng Ren¹, Zehao Lin¹, Jiahao Huo¹, Tianyi Chen², Kai Chen¹, Kehang Li², Zhiqiang Yin³, Qingchen Yu¹, Bo Tang^{1,†}, Hongkang Yang^{1,†}, Zhi-Qin John Xu^{2,†}, Feiyu Xiong^{1,†}

¹MemTensor (Shanghai) Technology Co., Ltd., ²Shanghai Jiao Tong University, ³Renmin University of China, ⁴Research Institute of China Telecom

Abstract

Large Language Models (LLMs) have emerged as foundational infrastructure in the pursuit of Artificial General Intelligence (AGI). Despite their remarkable capabilities in language perception and generation, current LLMs fundamentally lack a unified and structured architecture for handling memory. They primarily rely on parametric memory (knowledge encoded in model weights) and ephemeral activation memory (context-limited runtime states). While emerging methods like Retrieval-Augmented Generation (RAG) incorporate plaintext memory, they lack lifecycle management and multi-modal integration, limiting their capacity for long-term knowledge evolution. To address this, we introduce —a memory operating system designed for LLMs that, for the first time, elevates memory to a first-class operational resource. It builds unified mechanisms for representation, organization, and governance across three core memory types: parametric, activation, and plaintext. At its core is the **MemCube**, a standardized memory abstraction that enables tracking, fusion, and migration of heterogeneous memory, while offering structured, traceable access across tasks and contexts. MEMOS establishes a memory-centric execution framework with strong controllability, adaptability, and evolvability. It fills a critical gap in current LLM infrastructure and lays the groundwork for continual adaptation, personalized intelligence, and cross-platform coordination in next-generation intelligent systems.

Date: 2025.05.27

Correspondence: {tangb,yanghk,xuzq,xiongyf}@mentensor.cn

Author Legend: *Co-equal primary author, †Correspondence

核心思想：

提出类操作系统的记忆中枢，解决四大问题：

1. 长期对话状态建模差
2. 知识演进缺失
3. 用户偏好无法持久建模
4. 平台记忆孤岛

记忆分类：

- 参数记忆：模型权重、LoRA 模块
- 激活记忆：KV 缓存、注意力图
- 明文记忆：外部文本、图谱、Prompt

技术框架：

- 统一的记忆 API
- 跨平台记忆共享
- 记忆生命周期管理

Part 2: 模型驱动派（5篇）

6. Memorizing Transformers ★★★★★ 开山之作

论文标题：Memorizing Transformers

发表时间：2022年3月

机构：Google Research

论文链接：<https://arxiv.org/abs/2203.08913>

GitHub：<https://github.com/lucidrains/memorizing-transformers-pytorch>

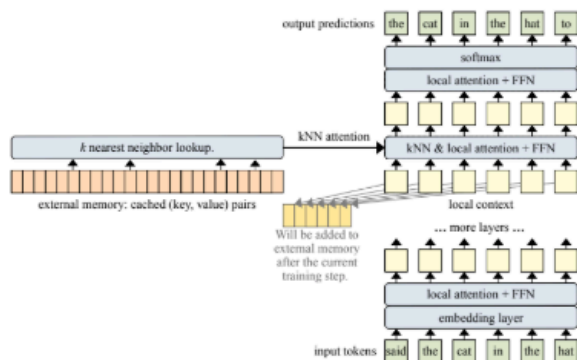


Figure 2: We extend Transformers with access to (key, value) pairs of previously seen subsequences.

Memorizing Transformers - Pytorch

Implementation of [Memorizing Transformers](#) (ICLR 2022), attention net augmented with indexing and retrieval of memories using approximate nearest neighbors, in Pytorch

This repository deviates from the paper slightly, using a hybrid attention across attention logits local and distant (rather than the sigmoid gate setup). It also uses cosine similarity attention (with learned temperature) for the KNN attention layer.

核心理想：

首次在 Transformer 中引入外部记忆，通过 KNN 检索历史信息：

- 每个 token 都存储到外部记忆库

- 推理时通过 KNN 检索相关历史
- 融合外部记忆和内部注意力

算法流程：

1. 编码：将历史 token 存入记忆库
2. 检索：KNN 搜索最相关的 k 个记忆
3. 融合：Attention(Q, K_memory, V_memory)
4. 输出：结合内部和外部注意力

实验效果：

- 在长文本任务上困惑度降低 **4%**
- 外推能力显著提升

算法岗价值：

- 理解外部记忆的设计思想
- KNN 检索的优化策略
- Attention 融合机制

7. MemoryLLM ★★★★★ 可更新记忆

论文标题：MemoryLLM: Towards Self-Updatable Large Language Models

发表时间：2024年2月

机构：清华大学

论文链接：<https://arxiv.org/abs/2402.04624>

GitHub：<https://github.com/wangyu-ustc/MemoryLLM>

MemoryLLM & M+

This is the official implementation of paper **MemoryLLM: Towards Self-Updatable Large Language Models and M+: Extending MemoryLLM with Scalable Long-Term Memory**.



Official Links

[memoryllm](#) [paper](#) [m+](#) [paper](#)

[memoryllm 7b](#) [checkpoints](#) [memoryllm 8b](#) [checkpoints](#) [memoryllm 8b chat](#) [checkpoints](#) [mplus 8b](#) [checkpoints](#)

Release Notes

- [2025/07/27] 🔥 Updated the training code of `mplus-8b` and open-sourced at [mplus-8b-branch](#).
- [2025/02/07] 🔥 The model `mplus-8b` has been uploaded to [mplus-8b](#).
- [2025/02/01] 🔥 New paper [M+: Extending MemoryLLM with Scalable Long-Term Memory](#) is on Arxiv!
- [2024/08/30] 🔥 We release [memoryllm-8b-chat](#), the chat model built on top of [memoryllm-8b](#).
- [2024/08/23] 🔥 We release [memoryllm-8b](#) with 1.67B memory equipped on Llama3!
- [2024/06/21] 🔥 Training code is provided in the folder `train`.
- [2024/06/02] 🔥 **MemoryLLM** checkpoint is [released](#)!
- [2024/05/02] 🔥 **MemoryLLM** is accepted to ICML 2024!

核心思想：

在每层 Transformer 中插入可更新的 Memory Tokens：

- **固定参数**：预训练的模型参数（不变）
- **Memory Tokens**：可读写的记忆单元（可更新）
- **终身学习**：持续学习新知识，对抗遗忘

技术细节：

1. Memory Tokens 设计

2. 读写机制 (Read/Write Gates)
3. 更新策略 (何时更新、如何更新)
4. 遗忘控制

适合研究方向:

- 终身学习
- 模型编辑
- 知识更新

8. Memory³ ★★★★★ 分层记忆

论文标题: Memory³: Language Modeling with Explicit Memory

发表时间: 2024年7月

论文链接: <https://arxiv.org/abs/2407.01178>

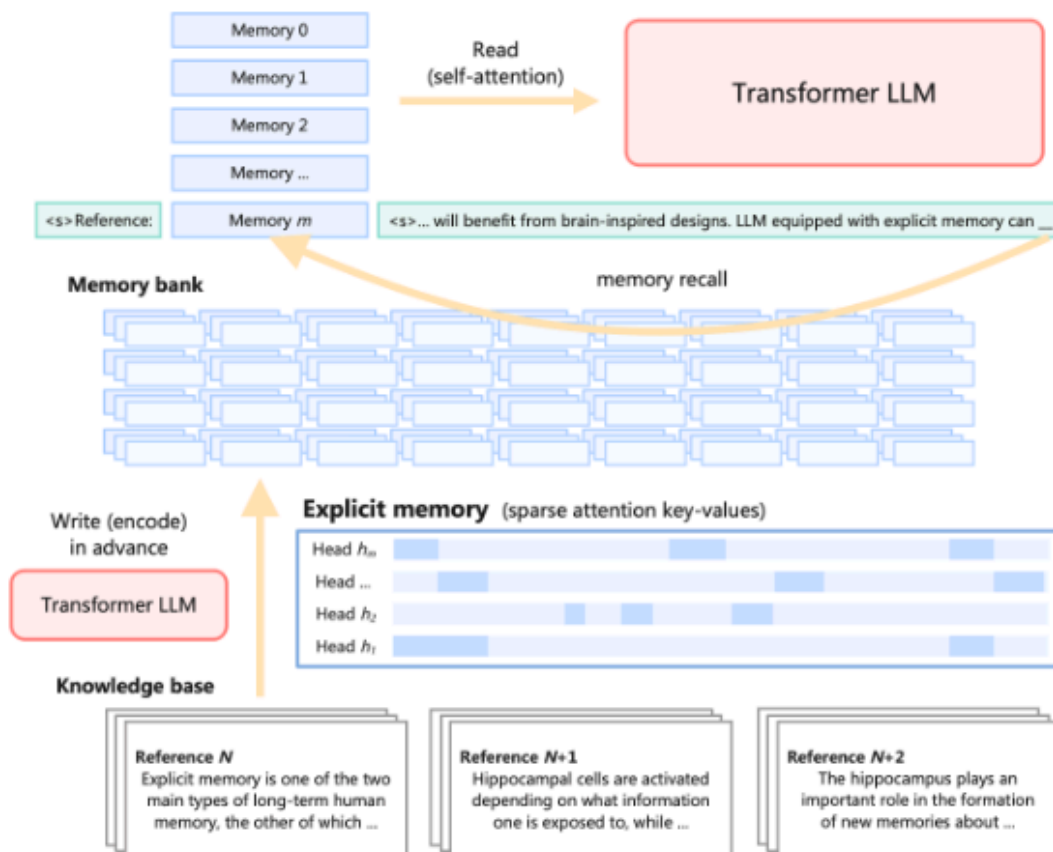


Figure 1: The Memory³ model converts texts to explicit memories, and then recalls these memories during inference. The explicit memories can be seen as retrievable model parameters, externalized knowledge, or sparsely-activated neural circuits.

核心思想:

模拟人脑, 将记忆分为三层:

- 感知记忆：原始输入
- 短期记忆：工作上下文
- 长期记忆：知识存储

算法设计：
每一层有不同的编码方式和检索策略

创新点：
首次提出分层记忆的完整框架

9. WISE ★★★★★ 双参数体系

论文标题：WISE: Rethinking the Knowledge Memory for Lifelong Model Editing
发表时间：2024年5月
机构：浙江大学
论文链接：<https://arxiv.org/abs/2405.14768>
GitHub：<https://github.com/zjunlp/EasyEdit>

EasyEditPublic

Watch25Fork326Star2.6k

main6 Branches0 Tags

Go to fileAdd fileCode

littlefive5

adapt the compute for new transformer version

e2c8553 · yesterday

1,578 Commits

.github/ISSUE_TEMPLATE	update readme.md	last month
demo	Merge branch 'zjunlp:main' into recovery-branch	4 months ago
easyeditor	adapt the compute for new transformer version	yesterday
examples	Merge branch 'main' into Xubqpanda	2 months ago
figs	update	6 months ago
hparams	Merge branch 'main' into main	last month
steer	Fix Transformers update bug #630	last week
tutorial-notebooks	vllm support	3 weeks ago
.DS_Store	update readme.md	last month
.gitignore	update reps	4 months ago
Dockerfile	Optimize Dockerfile for Efficient Layer Caching & Conda E...	10 months ago
LICENSE	Initial Commit	2 years ago
README.md	Update README.md	last month
README_2.md	vllm support	3 weeks ago

About

[ACL 2024] An Easy-to-use Knowledge Editing Framework for LLMs.

[ACL 2024] 一个易于使用的 LLM 知识编辑框架。

zjunlp.github.io/project/KnowEdit

natural-language-processingtool

efficientartificial-intelligence llama

gptbaichuan mmedit unlearning

trustworthy-ailarge-language-models

chatgptmodel-editing

knowledge-editingknowlmeasyedit

safeeditknoweditcknowledit

easyedit2

Readme

MIT license

Activity

Custom properties

2.6k stars

25 watching

326 forks

Report repository

核心思想：

- 主记忆：预训练知识（冻结）
- 侧记忆：后续编辑的知识（可更新）

适合场景：

- 知识编辑
- 事实更新
- 避免灾难性遗忘

10. Titans ★★★★★ 学习遗忘

论文标题：Titans: Learning to Memorize at Test Time

发表时间：2025年1月

机构：Google DeepMind

论文链接：<https://arxiv.org/abs/2501.00663>

GitHub：<https://github.com/lucidrains/titans-pytorch>

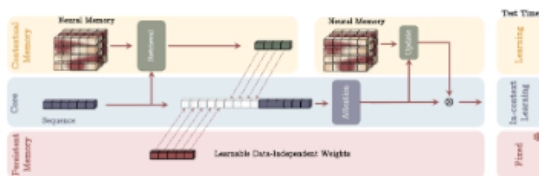


Figure 2: **Memory as a Context (MAC) Architecture.** This architecture includes three branches of (1) core, (2) contextual (long-term) memory, and (3) persistent memory. The core branch concatenates the corresponding long-term and persistent memories with the input sequence. Next, attention performs on the sequence and decides what part of the information should store in the long-term memory. At the test time, parameters corresponds to contextual memory are still learning, parameters corresponds to the core branch are responsible for in-context learning, and parameters of persistent memory are responsible to store the knowledge about tasks and so are fixed.

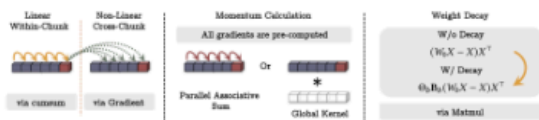


Figure 1: The illustration of how the training of neural memory can be done in parallel and using matmuls.

Titans - Pytorch

Unofficial implementation of [Titans](#) in Pytorch. Will also contain some explorations into architectures beyond their simple 1-4 layer MLP for the neural memory module, if it works well to any degree.

核心思想：

设计神经网络模块，学习何时存储、何时遗忘：

- 存储门控：决定哪些信息值得存储
- 遗忘门控：决定何时清理记忆
- 检索门控：决定调用哪些记忆

创新点：

让模型学会"断舍离"，避免记忆过载

两大流派完整对比

维度	模型驱动	应用驱动
代表论文	Memorizing Transformers MemoryLLM Memory³	MemGPT Mem0 Zep
核心思路	改造模型内部结构	应用层记忆管理
实现方式	Memory Tokens KNN 检索 门控机制	外部数据库 知识图谱 向量检索
优势	<ul style="list-style-type: none">性能上限高读取效率高深度集成	<ul style="list-style-type: none">落地快易扩展模型无关
劣势	<ul style="list-style-type: none">研发成本高需要重新训练通用性差	<ul style="list-style-type: none">依赖底层模型可能有延迟幻觉问题
适合岗位	🔬 算法工程师	🔧 开发工程师
研究方向	模型架构创新 训练算法优化	系统架构设计 工程实践优化

📖 综述论文（2篇必读）

Agent Memory Survey ⭐⭐⭐⭐⭐ 最全面

论文标题：大模型智能体记忆机制综述

论文链接：<https://arxiv.org/pdf/2404.13501.pdf>

GitHub：https://github.com/nuster1128/LLM_Agent_Memory_Survey

核心贡献：

- 1. 记忆分类框架：
 - 参数化记忆
 - 上下文记忆（结构化/非结构化）
- 2. 六大操作：
 - 巩固（Consolidation）
 - 更新（Update）
 - 索引（Indexing）

- 遗忘 (Forgetting)
- 检索 (Retrieval)
- 压缩 (Compression)

3. 研究主题:

- Memory 架构设计
- Memory 操作优化
- Memory 评估方法
- Memory 应用场景

使用建议:

- 建立 Memory 完整认知框架
- 了解研究全貌
- 选择研究方向

Multimodal Memory Survey ★★★★★ 多模态扩展

GitHub: <https://github.com/patrick-tssn/Awesome-Multimodal-Memory>

核心内容:

收录 400+ 篇多模态记忆论文:

- 视觉记忆 (Visual Memory)
- 机器人记忆 (Robotic Memory)
- 多模态上下文建模
- 音频、视频、图像、3D 记忆

适合方向:

- 多模态 Agent
- 具身智能
- VLM 应用

其他重要论文

MemAgent ★★★★★

- 论文: <https://arxiv.org/abs/2507.02259>

- 核心：强化学习记忆聚合
- 适合：Agent RL 研究

MemLong ★★★★★

- 论文：<https://arxiv.org/abs/2408.16967>
- 核心：检索记忆模块 + 可控注意力
- 适合：超长文本处理

🎯 如何选择论文阅读？

按目标选择

你的目标	推荐论文	阅读顺序
快速了解全貌	Agent Memory Survey	直接读综述
做工程落地	MemGPT → Mem0 → Zep	先易后难
做算法创新	Survey → Memorizing Transformers → MemoryLLM	建立框架后深入
发论文	读最新 5 篇（2024-2025）	找到 gap

💡 论文阅读建议

开发岗阅读策略

重点关注：

- ✅ 系统架构（怎么设计的）
- ✅ 实现细节（怎么实现的）
- ✅ 性能指标（效果如何）
- ❌ 可以跳过数学推导

阅读顺序：

Abstract → Introduction → System Design → Experiments

时间分配：

- 一篇论文：30-60 分钟
 - 重点：架构图 + 代码实现
-

算法岗阅读策略

重点关注：

- ☒ 问题定义（解决什么问题）
- ☒ 算法设计（创新点是什么）
- ☒ 数学原理（为什么有效）
- ☒ 实验设计（如何验证）
- ☒ 局限性（未来工作）

阅读顺序：

Abstract → Introduction → Method（重点！）→ Experiments → Conclusion

时间分配：

- 一篇论文：2-4 小时（精读）
 - 做笔记、画图、推公式
-



相关文档

- [Agent Memory 工具对比](#) - 实用工具推荐
 - [Agent Memory 技术教程](#) - 从原理到实战
 - [返回 Agent 资源总览](#)
-

👉 返回主文档：[AgentGuide README](#)