



# 从记忆到认知： AI Agent 如何实现真正的个性化服务

Bojie Li

Co-Founder & Chief Scientist, Pine AI

Oct. 2025

Press Space for next page →

# 概览



记忆的重要性与挑战

个性化价值 · 三层能力



记忆的表示

Notes · JSON Cards



记忆的检索

RAG · 上下文感知



记忆的评估

Rubric · LLM Judge



前沿研究

ReasoningBank

从个性化需求出发 → 理解记忆挑战 → 设计存储方案 → 实现智能检索 → 科学评估迭代

## 第一部分：记忆的重要性与挑战

# 个性化是真问题，而且是未来的核心竞争力

## 推荐系统的演进

### 传统媒体

人人日报，大家看一样的内容

### 字节跳动革命

每个人看到的都不一样

"每个人生活在不同的世界，有不同的价值观"

### 结论

个性化产品更符合人性 → 用户更愿意用

## AI 的未来也是如此

不应该只有一个 **Universal Value** (普世价值)

- 应该适应每个用户的价值观和偏好
- 细节上的价值观差异很大
- 个性化是 AI 产品的核心竞争力

### 关键洞察

正如推荐系统通过个性化内容提升用户体验，AI Agent 也需要通过个性化记忆来理解和服务每个独特的用户。

# 技术难点：记住事实 vs. 学习偏好

## 事实信息 (Factual Information)

相对容易

- 生日、地址、卡号
- 工作信息、联系方式
- 记住就行，没有歧义

我们已经做得不错

示例：

- "我的会员号是 12345" 
- "我的生日是 1990年1月1日" 
- "我住在北京市海淀区" 

## 用户偏好 (User Preference)

非常难

需要解决多个挑战：

### 1. 上下文依赖性强

- 用户在写论文时要求学术格式
- 不代表写旅游攻略也要学术格式
- AI 容易过度泛化偏好

### 2. 一次性行为 vs. 长期偏好

- "昨天我点了川菜"  $\neq$  "用户喜欢吃川菜"
- 可能只是朋友喜欢，或者一时兴起

### 3. 需要极精细的评估

- 必须有数据和测试来平衡
- 不能靠感觉

# 个性化价值对齐

## 类比：推荐系统的成功经验

### 传统方法：普世人类价值

- LLM 被对齐到"普世"价值
- 但我们真的有普遍认同的人类价值吗?
- 在细节上，价值差异是巨大的

### 从推荐到对齐：AI 的进化

### AI 应该做的是

- 不只是一个普世价值
- 适应每个用户的价值和偏好
- 认识到价值差异是巨大的

正如字节跳动认为"每个人生活在不同的世界，有不同的价值观"，AI Agent 也需要：

1. 理解个体差异：每个用户有独特的价值观和偏好
2. 动态适应：根据用户行为和反馈持续调整
3. 上下文感知：同一用户在不同场景下的需求不同

# 用户记忆不只是记录对话

## 记忆的本质

### 就像理解朋友

- 我们不记得他们说的每一句话
- 我们建立一个关于他们是谁的心智模型
- 他们的偏好、习惯、价值观

### 核心类比

用户记忆系统的目标是构建一个关于用户的、尽可能简洁而强大的预测模型，能够解释用户已有的行为，预测用户未来的需求。

## 两种记忆类型对比

类型	难度	示例
事实	简单	生日、地址、卡号
偏好	复杂	上下文相关、不断演变

学习用户偏好比存储事实信息困难得多

- 上下文相关：论文写作风格 ≠ 旅行指南
- 一次性 **vs.** 长期：“昨天点了川菜” ≠ “喜欢辣的”
- 过度泛化风险：AI 容易错误外推

# 三个记忆能力层次



## 层次 1：基本回忆

存储和检索显式用户信息

"我的会员号是 12345" → 准确回忆

可靠性的基础



## 层次 2：多会话检索

连接不同对话中的信息

消除歧义："为我的车预约保养" → 两辆车中的哪一辆？

理解复合事件

"取消洛杉矶行程" → 找到航班 + 酒店



## 层次 3：主动服务

无需明确请求即可预见需求

预订国际航班？→ 检查护照是否快过期

智能的最高体现

## 我们的评估框架

基于这三个层次，我们设计了 60 个测试用例（每层 20 个），每个用例包含 1-3 个会话，每个会话约 50 轮对话，包含大量事实细节。通过 LLM-as-a-judge + Rubric 方法对 agent 的回答进行多维度评分。

# 层次 1：基本回忆的评估

## 场景：银行账户设置

准确存储和检索用户在长对话中提供的结构化信息。

### 测试用例

- 47 分钟的银行开户对话
- 包含姓名、地址、SSN、账号等
- 50+ 轮对话中的大量细节

### 最终提问

"我的支票账户号码是多少？我需要设置工资直存。"

### 期望回答

准确提供账号 4429853327，最好也提供路由号 123006800

## 对话摘录

- user: I live at 1847 Maple Street,  
Apartment 3B, Portland, Oregon.
- assistant: Thank you. Phone number?
- user: My cell is 503-555-8924.
- ...
- assistant: Your new checking account  
number is 4429853327.
- user: Let me write that down...  
4429853327, right?
- assistant: Correct. And your savings  
account is 4429853328.
- ...
- user: Can I use PIN 4827?
- assistant: Yes, 4827 is set as your PIN.
- ...
- assistant: Your online banking username  
is MRobertson503.

关键：从 50+ 轮对话中精确检索特定账号

# 层次 2：多会话检索 – 消歧场景

## 场景：多辆车的服务预约

用户在不同会话中提到拥有多辆车，当请求模糊时，Agent 需要主动消歧。

### 会话 1：保险添加新车

用户 William Chen 将 2023 Tesla Model 3 添加到保险，原有 2019 Honda Accord

### 会话 2：预约汽车保养

为 Honda Accord 预约了 11月24日 8AM 的 30K 保养服务

### 最终提问

"I need to schedule service for my car."

### 期望行为

识别歧义，列出两辆车的状态，询问具体哪辆

## 对话摘录

### # 会话 1 - 保险

- user: I just bought a 2023 Tesla Model 3.
- assistant: Is this replacing the Honda?
- user: It's an addition. I'm keeping the Honda for my wife to drive.
- ...
- assistant: Honda is SF-789234501-01, Tesla is SF-789234501-02.

### # 会话 2 - 保养预约

- user: I need an oil change and the 30,000 mile service.
- assistant: What vehicle?
- user: It's a 2019 Honda Accord.
- ...
- assistant: Friday Nov 24th at 8 AM.
- Confirmation: FS-447291.

关键：发现用户有两辆车，Honda 已有预约，Tesla 没有

# 层次 2：复合事件 – 旅行取消的级联效应

## 场景：一个大事情包含多个小事情

用户说"取消我的洛杉矶之旅"，系统需要理解\*\*\*"旅行"是一个复合事件\*\*\*，包含多个独立预订。

### 需要自动找出的关联预订

- 去往洛杉矶的机票
- 洛杉矶的酒店订单
- 可能的租车服务
- 活动门票、餐厅预订等

### 最终提问

"Cancel my LA trip next week."

### 期望行为

自动关联所有相关预订，提供统一的取消选项，说明各项的取消政策和退款情况

## 三个独立会话中分散的信息

### # 会话 1 - 航班预订 (Delta)

- 航班: DL 1234
- 日期: 12月20日 → 12月23日
- 目的地: Los Angeles
- 确认号: DELTA-ABC123

### # 会话 2 - 酒店预订 (Marriott)

- 酒店: Marriott Downtown LA
- 入住: 12月20日
- 退房: 12月23日
- 确认号: MAR-789456

### # 会话 3 - 租车预订 (Hertz)

- 公司: Hertz
- 取车: LAX, 12月20日 3PM
- 还车: LAX, 12月23日 12PM
- 确认号: HERTZ-456789

关键：三个会话分别与不同服务商交互，但都属于同一次"洛杉矶之旅"

# 层次 2：覆盖改写 – 订单的多次修改

## 场景：不断修改的定制家具订单

用户定制了一套餐桌椅，但在生产过程中多次修改需求。Agent 需要追踪所有变更，只保留当前有效的规格。

### 订单变更历史

- **8月20日**: 下单胡桃木餐桌 + 8把灰色椅子 + 1条长凳
- **9月5日**: 椅子颜色改鼠尾草绿，2把改为扶手椅
- **10月28日**: 绿色面料停产，需重新选色

### 最终提问

"What's the current status of my dining set order?"

### 期望行为

只返回\*\*最新状态\*\*：等待面料选择，交付日期待定，不要混淆历史规格

## 三个会话的关键变更

- # 会话 1 - 初始订单 (8月20日)
- 餐桌: 胡桃木 Hamilton, 72寸 (\$4,100)
  - 椅子: 8把标准椅, 灰色 (\$3,400)
  - 长凳: 1条配套长凳 (\$650)
  - 交付: 11月5日

- # 会话 2 - 设计变更 (9月5日)
- 椅子颜色: 灰色 → 鼠尾草绿 ✓
  - 椅子类型: 2把升级为扶手椅 (+\$200)
  - 交付: 11月5日 → 11月12日

- # 会话 3 - 面料问题 (10月28日)
- 问题: 鼠尾草绿面料停产！
  - 状态: 等待客户从新样品中选择
  - 交付: 待定 (取决于选择时间)

关键: Agent 必须识别旧信息已被覆盖，只有最新会话的状态才是有效的

# 层次 3：主动服务 – 护照过期预警

## 场景：国际旅行协调

用户在多个独立会话中提到了不同的信息，但从未将它们联系起来。Agent 需要主动关联这些分散的信息，推理出潜在风险。

### ⚠ AI 需要推理出的风险

- 护照到期：2025年2月18日（会话1提及）
- 返程日期：2025年1月22日（会话2提及）
- 日本要求入境时护照有效期  $\geq$  6个月！

### 最终提问

"I'm finalizing my trip to Tokyo in January. Is there anything I need to take care of before I go?"

### 期望行为

\*\*主动关联\*\*护照有效期与旅行日期，提醒用户护照可能不符合日本入境要求

## 三个独立会话的摘录

### # 会话 1 - 6月 护照更新地址 (USPS)

- user: I need to update my address on file.
  - assistant: I'll update that. I see your passport expires Feb 18, 2025.
  - user: Thanks, I'll deal with renewal later.
- # (护照有效期只是顺便被提到，未讨论旅行)

### # 会话 2 - 11月 机票预订 (Delta)

- user: I want to book Tokyo, Jan 15-22.
  - assistant: Great! I found flights for you.
  - user: Book the 2pm departure please.
  - assistant: Done. Confirmation: DELTA-JMK892
- # (只预订机票，未问护照问题)

### # 会话 3 - 10月 信用卡 (Chase)

- user: Will my Sapphire Reserve work abroad?
- assistant: Yes, no foreign transaction fees. Trip insurance covers purchases.

关键：三个会话中从未讨论过护照与旅行的关系，AI 需要自己推理出来！

# 层次 3：主动服务 – 设备损坏的保障整合

## 场景：手机屏幕摔碎

用户说“我的手机屏幕摔碎了”，Agent 需要主动整合分散在不同会话中的各种保障信息，找出最优解决方案。

### ⚠ AI 需要推理出的保障来源

- 原厂保修 (Apple 1 年, 到 2025 年 2 月)
- 信用卡保护 (Chase Sapphire, \$50 免赔额)
- 运营商保险 (用户拒绝了, 不适用)

### 最终提问

"My phone screen just cracked. What are my options?"

### 期望行为

主动列出所有保障选项，比较费用和流程，推荐最优方案 (Chase 信用卡保护)

## 多个独立会话中分散的信息

# 会话 1 - 2 月 购机 (Best Buy)

- **user:** I'll use my Chase Sapphire Reserve.
- **assistant:** That card extends warranties and has purchase protection.
- **Phone:** iPhone 14 Pro, \$1,099

# (原厂保修: 到 2025年2月)

# 会话 2 - 2 月 手机激活 (Verizon)

- **user:** No thanks, I don't need insurance.
- **assistant:** You can add it later if needed.

# (用户拒绝了 Verizon Mobile Protect)

# 会话 3 - 8 月 账单查询 (Chase)

- **user:** Do I have phone protection?
- **assistant:** Yes, up to \$800 per claim, \$50 deductible. Must pay bill with this card monthly.

# (确认信用卡保护仍然有效)

关键：AI 需要整合三个来源，推理出 Chase 保护是最优选择 (\$50 免赔 vs Apple \$379)

# 层次 3：主动服务 – 报税季材料准备

## 场景：临近报税季的主动提醒

当用户在1月初提到"准备报税"时，Agent 应主动汇总全年散落在不同会话中的税务相关信息。

### AI 需要主动关联的历史信息

- 2月：房贷申请 (利息 \$31,000, Points \$7,500)
- 6月：股票卖出 (Apple, 资本利得 \$33,000)
- 8月：慈善捐赠 (Microsoft 股票 \$25,200)
- 10月：副业咨询收入 (\$18,000, 有家庭办公室)

### 最终提问

"I'm preparing my taxes. What should I know?"

### 期望行为

主动列出所有税务相关项目，提醒需要的表格，标记容易遗漏的抵扣项

### 全年会话中分散的税务信息

#### # 会话 1 - 2月 房贷 (First National Bank)

- 贷款: \$500,000, 6.75% 利率
- 利息: ~\$31,000/年 (可抵扣)
- Points: \$7,500 (可抵扣)

#### # 会话 2 - 6月 股票 (Charles Schwab)

- 卖出: 300股 Apple, \$55,590
- 成本: \$22,500 → 资本利得 \$33,000

#### # 会话 3 - 8月 捐赠 (United Way)

- 捐赠: 72股 Microsoft, 价值 \$25,200
- 避免资本利得税 \$4,374

#### # 会话 4 - 10月 副业 (SBDC)

- 收入: \$18,000, 家庭办公室 6%
- 自雇税: ~\$1,683

关键：四个会话跨越整年，AI 需要主动汇总并提醒用户准备所有相关表格

# 层次 3：主动服务的核心能力

## 与层次 1、2 的本质区别

### 触发方式与信息来源对比

层次	触发方式	信息来源
L1	用户直接询问	单一会话
L2	用户模糊请求	多个会话
<b>L3</b>	无需询问	跨时间跨领域

## 三个典型场景回顾

**\*\*护照预警\*\***: 机票 + 护照有效期 → 入境风险

**\*\*设备保障\*\***: 购买 + 信用卡 + 保险 → 最优方案

**\*\*报税准备\*\***: 全年交易记录 → 完整税务清单

## 关键技术挑战

### 时间跨度

需要关联数月甚至数年前的对话，识别仍然有效的信息

### 领域跨越

将不同服务商、不同场景的信息进行关联推理

### 主动推理

用户没有明确请求，但 Agent 应该主动发现并提醒

### 优先级判断

识别真正紧急和重要的问题，避免信息过载

## 第二部分：记忆的表示

# 记忆的表示（一）：自然语言方式

## Simple Notes 模式

### 极简主义设计

每条记忆是原子化的事实陈述：

- "用户邮箱: john@example.com"
- "偏好编程语言: Python"

#### 优势

- 极低认知负荷 -  $O(1)$  操作复杂度

#### 劣势

- 信息关联性丢失 - 语义割裂

## Enhanced Notes 模式

### 完整上下文保留

段落形式保存完整背景：

"用户在 TechCorp 担任高级软件工程师，专注于机器学习领域已有三年，目前正在领导一个推荐系统项目。"

#### 优势

- 语义完整性 - 叙事结构保留

#### 劣势

- 存储冗余 - 更新复杂

### 共同特点

以自然语言为主要载体，适合人类阅读理解，但缺乏机器可操作的结构化信息。

# 记忆的表示（二）：结构化方式

## JSON Cards 模式

### 结构化组织

三层嵌套：类别 → 子类别 → 键值对

```
{  
  "personal": {  
    "contact": {"email": "john@example.com"}  
  },  
  "work": {  
    "position": {"title": "Senior Engineer"}  
  }  
}
```

### 优势

- 部分更新 - 可扩展

### 劣势

- 刚性结构 - 多维信息难分类

## Advanced JSON Cards 模式

### 情境知识管理

在基础 JSON 上增加元数据字段：

- `backstory` : 信息来源的叙事背景
- `person` : 信息主体的身份标识
- `relationship` : 主体与用户的关系
- `timestamp` : 记录时间戳

### 示例

"为 8 岁女儿 Sarah 的湿疹治疗联系的皮肤科医生 Dr. Chen"  
→ `person: Sarah, relationship: daughter`

### 共同特点

以结构化数据为主要载体，便于程序化操作和精确检索，适合需要消歧的关键信息存储。

# 知识图谱的局限性

## 知识图谱的承诺

三元组表示：实体-关系-实体

看似强大

- 更灵活的信息网络 - 适合表示复杂关系 - 可以进行图查询

## 实际问题

语义降级不可避免

原始表达： "如果下周还下雨，我就取消去海边的计划，改成去博物馆"

知识图谱表示：

- (我, 有计划, 海滩旅行)
- (我, 有备选计划, 博物馆旅行)

丢失的信息：

- 条件关系： "如果-那么-否则"
- 时间依赖： "下周还下雨"
- 决策逻辑的核心结构

## 推理能力的限制

擅长：结构化查询

- 模式匹配 - 路径查找 - 找到所有与"我"相关的"计划"

不擅长：逻辑推理

- 反事实推理： "如果不下雨会怎样？" - 假设检验 - 类比推理

## 最佳实践

自然语言 + 结构化元数据

以完整、简洁的自然语言形式保存复杂信息，辅以 JSON Cards 等结构化元数据进行索引和检索。

在信息完整性与查询效率之间取得最佳平衡。

# 案例分析：ChatGPT 的记忆系统

## 四层上下文架构

通过逆向工程发现，ChatGPT 每次收到消息时会注入四层上下文：

### 1. Session Metadata (会话元数据)

设备类型、浏览器、时区、订阅级别等，会话结束后不保留

### 2. User Memory (用户记忆)

用户显式存储的长期事实（如“记住我是...”），每次请求都注入

### 3. Recent Conversations Summary (对话摘要)

近期对话的轻量级摘要（约15条），只包含用户消息，不含助手回复

### 4. Current Session (当前会话)

滑动窗口内的完整对话历史，超出 token 限制时旧消息被截断

## 关键设计选择

### 无向量数据库

不使用传统 RAG 的向量检索，而是预计算轻量级摘要直接注入，牺牲详细历史换取速度和效率

### 被动记忆机制

只有用户显式请求“记住这个”或模型检测到符合 OpenAI 标准的事实才会存储

### Simple Notes 模式

每条记忆是独立的事实陈述，缺乏信息之间的关联结构

### API 不可用

记忆功能未向开发者开放，限制了第三方应用集成

参考：Manthan Gupta, "I Reverse Engineered ChatGPT's Memory System"

# 案例分析：Claude 的记忆系统

## 与 ChatGPT 的核心差异

Claude 采用了完全不同的记忆架构：按需检索而非预算算注入。

### User Memories (用户记忆)

与 ChatGPT 类似的长期事实存储，但支持隐式更新——系统会在后台定期根据对话内容自动更新记忆

### Rolling Window (滚动窗口)

约 190k token 的完整消息历史，超出后旧消息被丢弃

### conversation\_search 工具

按需搜索历史对话，按主题或关键词检索，仅在模型认为需要时调用

### recent\_chat 工具

基于时间检索近期对话，同样是按需调用

## 设计哲学对比

### ChatGPT：预算算 + 注入

每次请求自动注入对话摘要，保证基本的跨会话连续性，但摘要是轻量级的，缺乏细节

### Claude：选择性检索

不自动注入历史摘要，而是让模型自主判断何时需要历史上下文，然后通过工具调用检索

## 权衡取舍

维度	ChatGPT	Claude
连续性	自动保证	依赖模型判断
细节深度	较浅	可按需深入
效率	固定开销	按需消耗

参考：Manthan Gupta, "I Reverse Engineered Claude's Memory System"

# ChatGPT 与 Claude 记忆系统的局限性

## 共同的不足

### 扁平化存储

两者都缺乏信息之间的关联和层次结构，无法表达复杂的语义关系

### 无消歧机制

当存在多个相关但不同的实体时（如两辆车），缺乏有效的区分手段

### 缺乏主动服务

两者都无法实现第三层次的主动预见性服务

## 各自的特定问题

### ChatGPT

对话摘要过于简略，丢失重要细节

### Claude

依赖模型判断何时检索，可能遗漏相关上下文

## 三层次评估框架对照

层次	ChatGPT	Claude
L1: 基本回忆	✓ 满足	✓ 满足
L2: 多会话检索	⚠ 摘要过浅	⚠ 检索不稳定
L3: 主动服务	✗ 未实现	✗ 未实现

## 改进方向

- 采用 Advanced JSON Cards 增强元数据
- 引入上下文感知的自动提取
- 建立记忆之间的关联图谱
- 实现基于记忆的主动推理

# 实验：四种记忆模式的对比

实验设计 ( `projects/week2/user-memory` )

基于三层次评估框架，系统性比较四种模式：

模式	简单性	表达力	更新性	适用场景
Simple Notes	★★★★★	★★	★★★★★	快速记录临时信息
Enhanced Notes	★★★★	★★★★★	★★	需要完整语义的场景
JSON Cards	★★★★	★★★	★★★★★	结构化信息管理
Advanced JSON	★★	★★★★★	★★★	需要消歧的关键信息

## 关键发现

没有“最好”的模式

最优选择取决于具体场景、成本预算和任务需求

混合使用是趋势

Simple Notes 快速记录 + Advanced JSON 处理关键信息

## 第三部分：记忆的检索

# 传统 RAG 的局限性

问题：扁平化处理导致信息丢失

案例一：黑猫白猫的计数问题

知识库中有 100 个独立案例：

- 90 只黑猫
- 10 只白猫

用户问："黑猫和白猫的数量比例？"

RAG 系统的困境：

- 检索受限于 top-k (如 k=20)
- 无法保证召回全部案例
- 只能基于不完整样本推理
- 结果：错误的比例结论

## 核心问题

简单的 RAG 方式，即把原始案例直接丢进知识库，是远远不够的。必须在索引阶段投入计算资源，对原始知识进行主动的提炼、抽象和结构化。

案例二：Xfinity 优惠规则

知识库中有三个孤立案例：

- 退伍军人 John 成功申请优惠
- 医生 Sarah 获得折扣
- 教师 Mike 不符合条件

用户问："我是护士，能享受优惠吗？"

RAG 系统的问题：

- "护士"与"医生"语义相近
- 优先检索到 Sarah 的案例
- 错误推断护士也可以
- 根本原因：未召回完整规则边界

# 解决方案：知识提炼与结构化

## 案例一的正确做法

预先提炼统计摘要

将 100 个个体案例压缩为：

### 摘要文档

"共有 100 只猫： - 90 只黑猫（90%） - 10 只白猫（10%）"

结果：一次检索就能获得准确、完整的统计信息

### 核心原则

将"100 个个体案例"压缩为统计摘要，将"三个孤立案例"提炼为明确规则。只有这样，才能构建出真正高效、可靠的 Agent 知识体系。

## 案例二的正确做法

提炼明确规则

从三个孤立案例中提取：

### 规则文档

"Xfinity 优惠仅适用于： - 退伍军人 - 医生

其他职业不符合条件。"

结果：无论用户问及任何职业，一次检索就能获得完整、准确的规则定义

# 结构化索引：RAPTOR vs GraphRAG

## RAPTOR：树状层次结构

自下而上的递归抽象

1. 叶子节点：文档切分为小文本块
2. 聚类：语义相近的块分组
3. 摘要：为每组生成父节点
4. 递归：逐层抽象到根节点

检索过程：

- 从高层摘要定位宏观概念
- 沿树向下钻取到具体细节
- 由宏观到微观的检索路径

擅长

捕捉知识的层次结构和抽象关系

两者关系

并非互相取代，而是互补。理想方案是结合使用，构建既有深度又有广度的立体化知识索引。

## GraphRAG：网络关联图

实体-关系建模

1. 提取实体：人物、地点、概念、术语
2. 提取关系：实体之间的各种关系
3. 社区发现：语义紧密关联的实体集群
4. 集群摘要：为社区生成摘要

检索过程：

- 定位核心实体
- 遍历关系边找到相关实体
- 通过社区分析提供上下文

擅长

揭示知识的横向关联和网络结构

# 上下文感知检索：解决上下文丢失

## 问题：孤立文本块的歧义

示例文本块：

"该公司第二季度的收入增长了3%"

缺失的上下文：

- "该公司"是哪家公司？
- 报告发布于何时？
- 与哪个产品线相关？

结果

语义信息严重损失，检索准确率下降

## 解决方案：上下文前缀

Anthropic 的上下文感知检索

第一步：为文本块生成上下文前缀

LLM 生成：

"本段内容节选自 ACME 公司 2025 年第二季度财务报告的'关键业绩指标'章节"

第二步：拼接后索引

"本段内容节选自 ACME 公司 2025 年第二季度财务报告的'关键业绩指标'章节 该公司第二季度的收入增长了3%"

效果

结合 BM25，检索失败率降低 49%；结合重排序器，失败率降幅达 67%

# 用户记忆的双层结构

## JSON Cards (常驻上下文)

结构化核心事实，随身备忘录

护照 2025-02 过期 · 东京之行

## 上下文感知 RAG (按需检索)

非结构化对话细节，强大搜索引擎

[上下文: 11月预订1月航班...]

### 两者必须协同工作

1. JSON Cards  
提供事实框架

2. LLM 推理  
发现潜在关联

3. RAG 验证  
获取对话证据

4. 主动服务  
护照快过期！

JSON Cards 告诉 Agent "有什么"，RAG 告诉 Agent "细节是什么"，二者缺一不可

# Agent 记忆架构

## 上下文常驻

### 基础知识 → System Prompt

用户 JSON Cards 直接放入 Agent 上下文，无需调用工具即可访问

## 三个检索工具

### search\_user\_memory

Agentic Search on User Memory

后端: Embedding Search → Rerank → 返回相关记忆

### search\_conversations

Agentic Search on Conversation Summaries

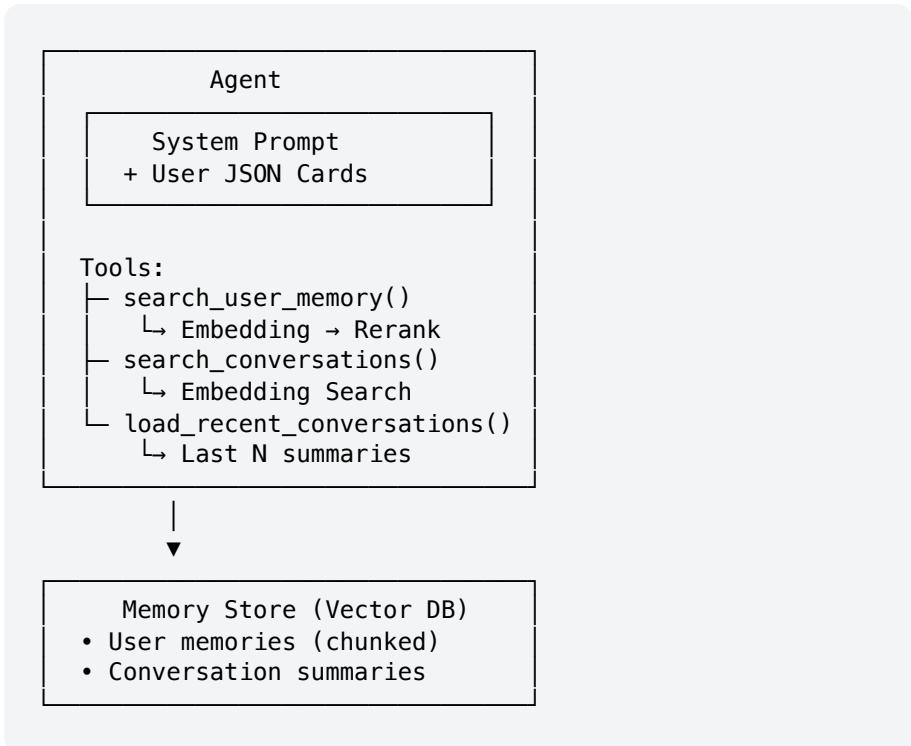
后端: Embedding Search → 返回相关历史对话摘要

### load\_recent\_conversations

Load Last N Conversation Summaries

直接加载最近 N 轮对话的摘要，无需语义搜索

## 架构示意图



## 设计原则

高频信息常驻上下文，长尾细节按需检索

# 主动服务：存储+检索的自然结果

## 核心洞察

### 主动服务不是独立的能力层

当存储和检索做好了，主动服务就会自然发生。它是结构化存储与智能检索协同工作的涌现结果。

## 为什么会自然发生？

### 结构化存储（JSON Cards）提供：

- 关键事实常驻上下文（如护照有效期）
- 元数据支持关联推理（如时间戳、实体类型）

### 智能检索（上下文感知 RAG）提供：

- 按需访问历史对话细节
- 自动连接相关信息片段

两者结合，Agent 自然能发现“东京机票在1月→护照2月过期”这样的关联

## 主动服务的实例

### 国际旅行预警

JSON Cards 存储护照信息 + RAG 检索航班预订 → 自动发现时间冲突

### 设备损坏处理

JSON Cards 存储设备和保险信息 → 自动列出所有适用的保护选项

### 报税季准备

JSON Cards 存储收入类型 + RAG 检索交易记录 → 自动汇总相关文件

### 实现路径

不需要单独为“主动服务”设计机制。专注于做好存储和检索，LLM 的推理能力会完成剩下的工作。

## 第四部分：记忆的评估

# 为什么需要评估?

## 评估是 Agent 工程的指南针

Agent 系统的构建涉及大量设计决策，而这些决策往往没有显而易见的“正确答案”。

### 关键决策点

- 工作流设计: Workflow vs Autonomous 模式
- 提示词设计: 结构化 vs 规则列表
- 记忆模式: Simple Notes vs JSON Cards
- 检索策略: 预计算注入 vs 按需检索

### 核心洞察

某些看似合理的设计实际上会损害性能，而另一些看似琐碎的细节却能带来显著提升。只有通过严格的对比评估，这些反直觉的真相才能被揭示。

## 评估的三重价值

### 1. 指导设计决策

在没有评估的情况下，我们只能依靠直觉，而直觉往往靠不住

### 2. 提供改进信号

不仅告诉“好还是不好”，更重要的是揭示“为什么好/不好”

### 3. 支撑模型升级决策

新模型发布时，只有在自己的评估集上测试，才能做出数据驱动的升级决策

### 消融实验方法论

保持系统其他部分不变，只修改一个特定组件，观察对整体性能的影响

# 评估环境的基本组成

## 五大核心要素

### 1. 数据集 (Dataset)

定义任务集合，每个任务包含初始状态、目标描述、参考解决方案

### 2. 环境状态 (Environment State)

维护任务执行过程中的所有可变信息（数据库、文件系统、对话历史）

### 3. 工具接口 (Tools)

Agent 与环境交互的通道，需保证功能完整但避免过度简化

### 4. 评价指标 (Rubric)

定义如何量化 Agent 表现，是评估中最具挑战性的部分

### 5. 执行协议 (Interaction Protocol)

规定交互模式和终止条件

## 人机交互型评估的关键原则

### 渐进式信息透露

绝不能一开始就把用户掌握的所有信息全部暴露给 Agent。信息应该按需、渐进地在对话过程中透露。

### 用户模拟 (User Simulation)

用另一个 LLM 扮演用户角色，根据预定义指令：

- 逐步透露必要信息
- 回应 Agent 的询问
- 任务完成后发出终止信号

### 双重验证机制

- 检查数据库最终状态是否正确
- 检查对话中是否输出了必要的关键信息

# Rubric: LLM 评判的依据

## 什么是 Rubric?

Rubric (结构化评分标准) 是让 LLM-as-Judge 评判过程客观、一致、可解释的核心工具。类似于高考、GRE 写作、托福口语的评分细则。

## 四个设计准则

**\*\*基于专家指导\*\*:** 反映领域专业知识, 捕捉正确回应所需的核心事实、推理步骤

**\*\*全面覆盖\*\*:** 涵盖多个维度 (准确性、连贯性、完整性), 同时定义正面和负面标准

**\*\*重要性权重\*\*:** 事实正确性必须优先于文体清晰度 (Essential / Important / Optional / Pitfall)

**\*\*自包含评估\*\*:** 每个评价项独立可操作, 不依赖外部上下文

## 用户记忆评估的 Rubric 示例

```
dimensions:  
  factual_precision:  
    weight: essential  
    levels:  
      - 4: 所有事实完全正确  
      - 3: 关键事实正确, 细节略有偏差  
      - 2: 部分事实正确  
      - 1: 事实存在重大错误
```

```
factual_recall:  
  weight: important  
  levels:  
    - 4: 提供了所有相关信息  
    - 3: 提供了主要信息  
    - 2: 遗漏部分关键信息
```

```
hallucination:  
  weight: veto # 一票否决  
  description: 任何编造的信息直接判定失败
```

## 防范奖励作弊

在 Rubric 中明确定义反向指标: 幻觉、讨好用户、关键词堆砌、回避问题

# 评估方法论：来自 Anthropic 的最佳实践

## 三种评估类型

### 单元测试 (Unit Tests)

确定性检查，用于验证格式、边界条件等可明确判断对错的场景

### LLM-as-Judge

用 LLM 评判输出质量，配合清晰的评分标准 (rubric) 可以达到与人类判断高度一致的效果

### 人工评估

在现实条件下测试，发现自动化评估无法捕捉的"粗糙边缘"

## 好的评估特点

- 具体清晰：有单一正确答案
- 真实世界：反映用户实际会遇到的问题
- 可诊断：足够简单以理解失败原因
- 代表性：反映终端用户体验

## Agent 评估的三个维度

### 最终答案正确性

Agent 是否给出了正确的最终答案？使用 LLM Judge 对比参考答案进行评分

### 工具使用准确性

Agent 是否选择了正确的工具？参数是否正确？能否从错误中恢复？

### 最终状态正确性 (τ-bench)

Agent 是否达到了正确的最终状态？适用于有副作用的任务（如取消订单）

## 评估技巧

- 如果每次系统改动对结果的影响越明显，需要的测试样本就越少
- 使用真实任务：有明确正确答案的用户实际场景
- 没有什么能完美替代人工评估：反复测试和直觉检查不可或缺

参考：Anthropic, "Context Engineering Best Practices" (AWS re:Invent 2025)

## 第五部分：前沿研究

ReasoningBank: 从经验中学习推理策略

# 现有 Agent 记忆系统的局限性

问题：Agent 无法从历史中学习

当前状态

现有的 LLM Agent 在处理连续任务流时，无法有效学习积累的交互历史。每个任务都被孤立处理，导致系统不断重复过去的错误，丢失有价值的洞察。

根本问题

缺乏真正的自我演化能力：Agent 不能随着时间推移而变得更强。

现有方法的缺陷

两类主流方案

原始轨迹存储

直接保存交互过程，缺乏提炼

成功流程记录

只保留 workflows / procedures，忽视失败

共同缺陷

- 无法提取高层次、可迁移的推理模式
- 过度强调成功经验，忽视失败的宝贵教训
- 被动记录，无法生成可执行的指导

# ReasoningBank：推理策略的记忆库

## 核心创新

### 从成功和失败中学习

ReasoningBank 从 Agent 自我判断的成功和失败经验中提炼可泛化的推理策略，不依赖真实标签（ground-truth labels）。

### 记忆内容的差异

方法	存储内容
原始轨迹	完整交互序列
成功流程	有效的行动模式
<b>ReasoningBank</b>	可迁移的推理策略

## 闭环学习机制

### 1. 检索相关记忆

面对新任务时，从 ReasoningBank 中检索语义相关的推理策略

### 2. 指导行动决策

用检索到的策略指导 Agent 的交互过程

### 3. 分析新经验

任务完成后，Agent 自我判断成功或失败

### 4. 提炼并整合

从新经验中提取推理策略，更新 ReasoningBank

# 为什么失败经验同样重要?

## 失败中的宝贵教训

### 传统观点的误区

大多数记忆系统只关注成功案例，认为失败不值得保存。但失败经验包含了关键的"预防性"知识。

#### 示例：网页导航任务

成功经验告诉你：“点击‘男装’类别找到商品”

失败经验告诉你：“不要在首页直接搜索，搜索框对复杂查询支持差”

失败经验提供了成功路径无法覆盖的边界条件。

## 对比信号的价值

### 成功 vs. 失败的对比学习

当同一类任务既有成功也有失败案例时，Agent 可以通过对比发现：

- 哪些策略在特定情境下有效
- 哪些看似合理的路径实际会失败
- 成功与失败的关键分界点

## ReasoningBank 的处理方式

从成功经验提取：有效策略（“这样做可行”）

从失败经验提取：预防性策略（“避免这样做”）

两者结合形成更完整的推理知识。

# MaTTS：记忆感知的测试时扩展

## 深度 vs. 广度

经验扩展的两种路径

### 广度扩展

增加更多任务数量（更多用户、更多场景）

### 深度扩展（MaTTS）

在每个任务上进行更多探索（更多尝试、更多变体）

## MaTTS 的核心思想

通过在单个任务上分配更多计算资源，生成丰富、多样的探索经验，为记忆合成提供更高质量的对比信号。

## 记忆与扩展的协同效应

### 正向反馈循环

高质量记忆 → 更有效的探索 → 更丰富的经验

↑  
←—— 更强的记忆合成 ←——

## 两种扩展模式

### 并行扩展

同时生成多个独立的解决路径

### 顺序扩展

根据前一次结果调整下一次尝试

MaTTS 将记忆驱动的经验扩展确立为 Agent 系统的新扩展维度。

# 实验结果与关键发现

## 基准测试结果

### 三大评估场景

#### WebArena (网页浏览)

- 复杂的网页交互任务
- 需要多步骤导航和操作

#### Mind2Web (网页理解)

- 真实世界网页的元素识别
- 动作预测和执行

#### SWE-Bench-Verified (软件工程)

- 代码库级别的问题修复
- 需要理解大型代码库

### 核心指标

- 效果: 最高 34.2% 相对改进
- 效率: 减少 16.0% 交互步骤

## 关键发现

### 记忆质量 > 数量

检索 1 个相关记忆的效果优于检索 4 个记忆。过多记忆可能引入冲突或噪声。

### 失败经验有独特价值

结合失败经验的系统表现优于只学习成功案例的系统。

### 涌现行为

随着记忆积累, Agent 开始展现出之前未见过的复杂推理策略。

### MaTTS 与记忆的协同

ReasoningBank + MaTTS 的组合效果最优, 验证了记忆与扩展的正向循环。

# ReasoningBank 对用户记忆系统的启示

## 从任务记忆到用户记忆

### 共同的核心挑战

ReasoningBank 解决的是 Agent 如何从任务交互中学习；用户记忆系统解决的是 Agent 如何理解和服务用户。两者面临相似的核心问题：

- 如何从原始数据中提炼高层次知识？
- 如何在检索时找到真正相关的信息？
- 如何让记忆系统持续演化？

### 关键洞察

不能简单存储原始数据，必须投入计算资源进行主动的提炼、抽象和结构化。

## 可迁移的设计原则

### 原则一：双向学习

不仅从用户的正向反馈中学习偏好，也从负向反馈中学习边界

### 原则二：闭环更新

记忆系统不是一次性构建，而是随着交互持续演化

### 原则三：质量优先

记忆的相关性和质量比数量更重要

### 原则四：自我判断

通过 LLM-as-a-judge 实现自动化的质量评估，减少人工标注依赖

# 总结：从记忆到认知的演进

## 技术演进路径

### 1. 记住事实

Simple Notes / JSON Cards

✓ 准确存储结构化信息

### 2. 理解上下文

Enhanced Notes / Advanced JSON

✓ 保留语义完整性和情境信息

### 3. 跨会话关联

结构化索引 + 上下文感知检索

✓ 消歧、发现复合事件

### 4. 主动预见

双层记忆架构 + 深度推理

✓ 无需明确请求即可提供帮助

## 核心洞察

### 个性化是真需求

从推荐系统的成功看，个性化产品更符合人性。AI Agent 也需要个性化记忆来适应每个用户的独特价值观和偏好。

### 偏好学习是难点

事实信息相对简单，但学习用户偏好面临上下文依赖、过度泛化等挑战，需要精细的评估和持续迭代。

### 知识提炼是关键

不能简单地把原始数据丢进知识库，必须投入算力进行主动提炼、抽象和结构化。

### 双层架构是最优解

结构化核心事实（常驻上下文）+ 上下文感知检索（按需访问），在完整性与效率间取得平衡。

# 未来展望

## 技术挑战

### 偏好学习的精细化

- 更好的上下文依赖建模
- 区分一次性行为与长期偏好
- 减少过度泛化风险

### 记忆压缩与组织

- 自动发现知识层次
- 动态调整记忆结构
- 平衡详细度与可访问性

### 跨模态记忆整合

- 文本、图像、音频的统一表示
- 多模态信息的关联检索

## 应用前景

### 个性化价值对齐

- 从普世价值到个体价值
- 动态适应用户的价值观演变
- 在细节层面实现真正的个性化

### 操作系統級助手

- 跨设备、跨应用的统一记忆
- 长期持续的用户画像构建
- 真正的主动式服务

### 隐私与透明性

- 用户对记忆的完全控制
- 可解释的记忆管理
- 敏感信息的分级保护

## 愿景

构建一个真正"懂你"的 AI 助手，不只是记住你说的话，而是理解你是谁，预见你的需求，成为你可信赖的终身伙伴。

# 从记忆到认知： AI Agent 如何实现真正的个性化服务

从简单记录到深度理解，从被动响应到主动服务

**Bojie Li**

Co-Founder & Chief Scientist, Pine AI

Powered by  Sliderv