arXiv:2603.01160v1 [cs.AI] 1 Mar 2026

Semantic XPath：面向对话式人工智能的结构化智能体记忆访问

Yifan Simon Liu[*1]、**Ruifan Wu**[*1]、**Liam Gallagher**[*1]、
Jiazhou Liang[*1]**Armin Toroghi**[1], **Scott Sanner**[1,2]加拿大多伦多大学[2]加拿大多伦多的 Vector 人工智能研究所

摘要

对话式人工智能（ConvAI）智能体越来越多地维护结构化记忆，以支持长期、面向任务的交互。上下文内存方法会将不断增长的历史记录附加到模型输入中，在上下文窗口限制下难以扩展。基于检索增强生成的方法会检索与请求相关的信息，但大多数方法都假设记忆是扁平集合，并忽略其结构。我们提出 SEMANTIC XPATH，这是一种树结构记忆模块，用于访问和更新结构化对话记忆。与扁平 RAG 基线相比，SEMANTIC XPATH 的性能提升了 176.7%，而所用令牌仅为上下文内存所需令牌的 9.1%。我们还推出 SEMANTICXPATH CHAT，这是一个端到端对话式人工智能演示系统，可将结构化记忆和查询执行细节可视化。总的来说，本文展示了一种候选方案，可用于构建基于结构化记忆的新一代长期、面向任务的对话式人工智能系统。

1 引言

对话式人工智能（ConvAI）智能体旨在支持长期、面向任务的交互，用户在其中管理多项任务（例如旅行行程规划、待办事项清单管理、套餐食谱推荐），并在多个版本中反复查看和修订不断演变的任务产物。为支持此类交互，对话式人工智能智能体日益开始维护结构化记忆，以保存对话历史、修订历史以及每项任务产物的层次组织（Rezazadeh 等，2024；Sun 和 Zeng，2025）。

早期的对话式人工智能代理采用上下文内方法，将不断增长的对话历史追加到模型输入中（图 1，左图；Zhang 等，2020）。然而，这种方法在上下文窗口限制下难以扩展，并会带来更高的令牌成本和延迟。长上下文还会导致推理能力下降，出现较差的

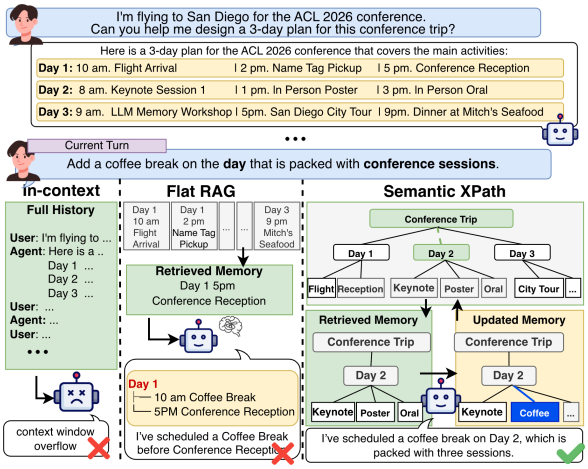


图 1：一个为在圣迭戈参加 ACL 会议、为期 3 天的行程规划的对话式人工智能智能体。**左：** 上下文内存储会附加完整对话。**中：** 平面检索增强生成从扁平化的记忆条目集合中检索内容。**右：** SEMANTIC XPATH 从结构化记忆中检索并更新相关子结构。

推理结果和幻觉输出，从而降低可靠性（Liu 等人，2024 年；Zhang 等人，2024 年）。我们的实证结果表明，上下文内存方法在仅经过五轮交互后，就无法满足一半的用户请求。

为解决这些局限，研究者提出了基于检索增强生成的系统，用于检索与当前用户请求相关的信息（Lewis 等人，2020年；Packer 等人，2023 年）。这些方法中有许多在检索时将记忆视为扁平的条目集合，忽略了记忆中通常存在的自然层次结构（Rezazadeh 等人，2024 年；Sarthi 等人，2024 年）。例如，在一份为期 3 天的 ACL 会议行程中，当用户要求“在会议场次安排最满的那天加一个咖啡休息时间”时，由于缺乏结构化的记忆组织，平面检索增强生成方法可能会检索到错误日期的会议相关活动（图 1，中）。

这些局限性促使我们提出一种能够检索相关结构化数据的结构感知型 RAG 方法

*贡献相同

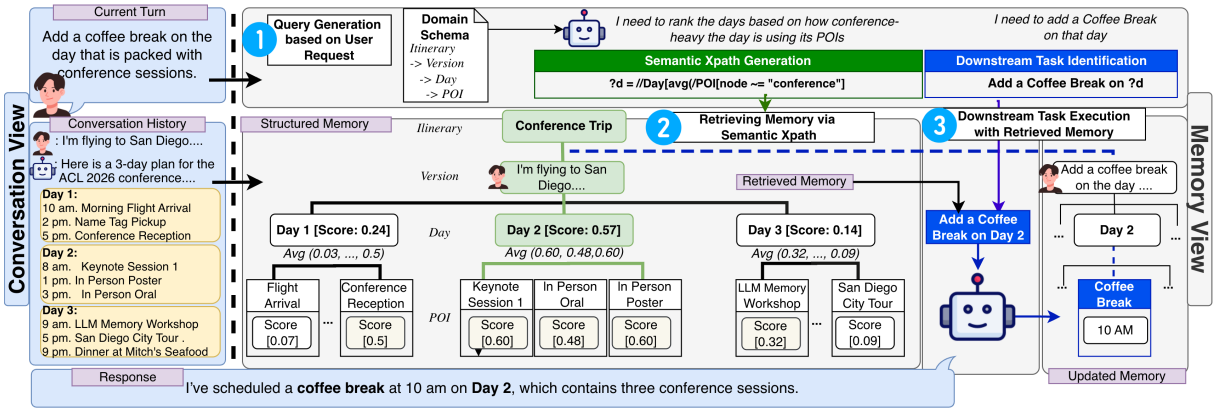


图 2：在 ACL 2026 行程规划中端到端运行的 SEMANTIC XPATH 流程，同时展示了对话视图（蓝色）和记忆视图（灰色）。**步骤 1：** 用户请求“在会议场次安排满满的那一天加一个咖啡休息时间。”，系统根据记忆模式将其转换为 SEMANTIC XPATH 查询。**步骤 2：** 系统通过结构匹配和语义相关性评分执行查询，选中第 *2* 天。**步骤 3：** 检索到的子结构被传递给下游生成模块，以插入一个咖啡休息点并回复用户。

从记忆中检索。针对层次化树形结构化数据的结构感知检索，一种广为人知的解决方案是 XPath 查询语言（Clark 和 DeRose，1999），但它缺乏语义感知，并且依赖精确匹配。因此，在本文中：

1. 我们提出了 SEMANTIC XPATH （图 1，右），这是一种树结构记忆模块，采用 XPath 风格的查询语言，仅检索相关记忆子结构，从而高效地访问和更新记忆。

2. 我们通过实验证明，与扁平 RAG 基线相比，SEMANTIC XPATH 的性能提升了 176.7%，而使用的令牌仅为上下文内存所需令牌的 9.1%。

3. 我们介绍 SEMANTICXPATH CHAT[1]，这是一个端到端的对话式人工智能系统，支持长期、面向任务的交互，并通过可视化结构化记忆和查询执行细节，展示 SEMANTIC XPATH。

2 系统描述

概述。 为支持用户与智能体之间的长期任务导向交互，我们推出了 SEMANTICXPATH CHAT，这是一个端到端的对话式人工智能系统，能够维护特定于任务的树状记忆。该系统采用由 SEMANTIC XPATH 支持的树状记忆模块，以适当的粒度仅检索相关结构化数据，从而实现高效的记忆访问与更新。

图 2展示了一个示例工作流程。用户创建为期 3 天的 ACL 会议行程后，提出请求：“在会议日程最满的那天安排一次咖啡休息时间。”系统首先将该请求转换为基于记忆模式的 SEMANTIC XPATH 查询，然后执行查询并对候选项进行排序。排名最高的结果是第 *2* 天，其中包含三项会议相关活动。随后，系统将检索到的节点及其对应子树用于下游生成，在第 *2* 天加入一项咖啡休息活动，并通过在结构化记忆中创建新的版本分支来记录此次更新。

功能。 图3展示了 SEMANTICXPATH CHAT 系统的屏幕截图，并通过同一个运行示例说明其主要功能。有关系统的详细演示，请参阅演示视频[2] 。

对话视图（左侧）显示对话历史。每轮对话都包含操作按钮，可将用户带到右侧面板中的结构化记忆视图和执行视图。

记忆视图（中间）将结构化记忆可视化。高亮路径表示与当前用户请求相关的路径。（例如，第 *2* 天和新增的咖啡休息时间均已高亮。）

执行视图（右侧）展示了逐步执行和评分过程。用户可以选择一个查询步骤来查看其执行详情，并点击候选节点以查看语义相关性分数。我们在 语义XPATH 查询语言和执行详情见第 3 节。

[1] 试用实时演示：https://semanticxpathchat.com。

[2] https://www.youtube.com/watch?v=l-DrEr4P5JA

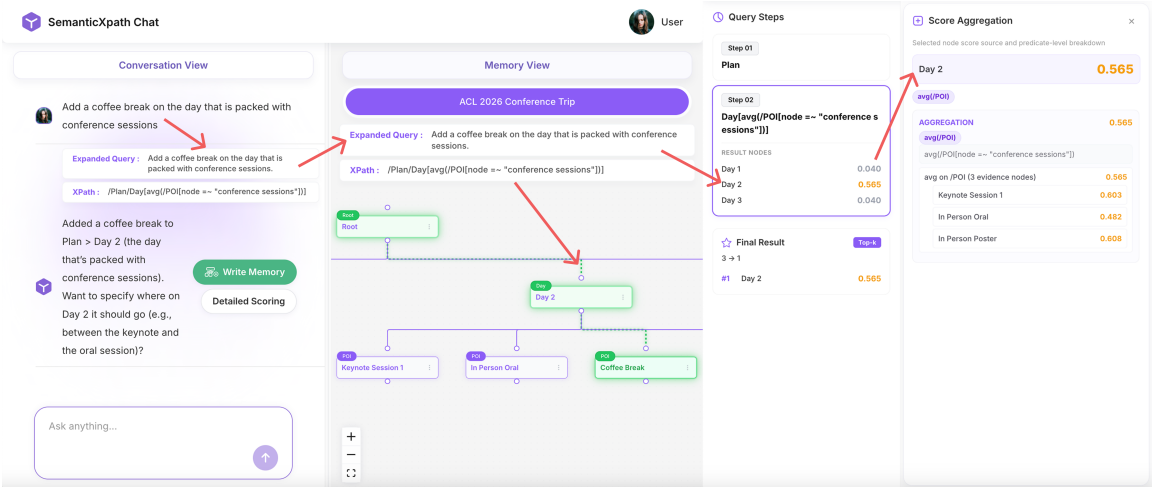


图3：SEMANTICXPATH CHAT 演示[1]。 **左侧：** 对话视图，用户请求在 会议场次排满的那天安排一段咖啡休息时间。 **中间：** 记忆视图，突出显示检索到的 第 *2* 天以及新插入的咖啡休息时间。 **右侧：** 执行视图，展示查询执行的逐步过程和评分详情。

3 语义 XPath

语义 XPATH 是一个树结构记忆模块，它采用 XPath 风格的查询语言，仅检索与任务相关的结构化数据，从而高效地访问和更新记忆。图 2 展示了整体流程。

3.1 数据模型

我们将结构化对话记忆表示为一棵有根树

$$T = (V, E, r),$$

其中每个节点 $v \in V$对应一个对话状态单元，E 表示父子关系，$r \in V$为根。每个节点都关联一个节点类型以及一组与该节点相关联的文本属性。

数据模型由模式所约束，该模式源自对话的自然层级结构和任务信息，并规定了节点类型及其属性。

ACL 旅行示例：“3 天 ACL 会议行程”的模式示例（图 2）为：

行程 → 版本 → 天 → 兴趣点。

- 行程是任务实例根节点。
- 版本记录修订历史（例如初始计划，以及添加咖啡休息时间后的更新版本）。
- 天 和 兴趣点 是特定于任务的。每一天都包含会议活动（例如“欢迎招待会”）。

3.2 查询语法

语义 XPath 文法

查询
$Q ::= \varnothing \mid S \varnothing \mid S\ Q$

Step
$S ::= A\ N\ [R]\ [P]$

轴导航器
$A ::= / \mid //$

节点选择器
$N ::=$ 节点类型 $\mid *$

位置选择器
$R ::= [i] \mid [-i] \mid [i{:}j]$

语义相关性算子
$P ::= \frac{(P+P)}{\max(P,P)} \mid P \cdot P \mid \min(P,P) \mid 1 - P \mid$ 局部 $\mid \mathrm{Agg}(S)$

局部语义相关性
局部::= [属性_名称 ≈字符串] | [节点 ≈字符串]

聚合语义相关性聚合 ::=平均 | 最小 | 最大 | 几何平均

查询 Q（符号 Q）是由多个步骤组成的序列。每个步骤（符号 S）包含四个组成部分：

- 轴导航器（符号 A）用于确定如何遍历结构化记忆。/ 表示选择子节点，// 表示选择后代节点。
- 节点选择器（符号 N）用于指定具体的节点类型或通配符。

- 可选的位置选择器（符号 R）。
- 一个可选的语义相关性算子（符号 P），用于为匹配的节点分配语义相关性得分。

我们通过两个具体示例进行说明。

ACL 旅行示例（续）： 查询 Q_1 针对请求“在会议场次最密集的天安排一次咖啡休息时间”可以翻译为：

```
//天[平均值(/兴趣点[节点~="会议"])]
```

Q_1 首先匹配任意深度的所有 天节点（$A = //$, $N =$ 天）。为了体现某一天被会议场次“排得很满”这一概念，它使用一个聚合语义相关性算子 $Agg(S')$为每一天评分，该算子聚合兴趣点级别的兴趣点[局部]证据。使用平均值可以得到一个衡量当天会议密集程度的密度式度量。

ACL 旅行示例（续）。查询 Q_2 针对“第三天只保留研讨会，移除所有其他活动。”可以翻译为：

```
//天[3]/兴趣点[¬[节点~="研讨会"]]
```

Q_2 首先使用位置选择器，//天[3], 以匹配结构化记忆中的第三天。然后，在兴趣点层级应用局部语义相关性算子（局部 = [节点 $\approx$ “研讨会”]），根据每个兴趣点的描述与“研讨会”的匹配程度为其评分。最后，语义相关性算子 $1 - P$ 再反转该分数，以定位所有不匹配“研讨会”的兴趣点，从而确定要移除的活动。

3.3 查询执行

SEMANTIC XPATH 查询执行维护一个加权节点集 $W \subseteq V \times [0, 1]$，其中每一对（$u$，$w$）都为一个节点 $u \in V$赋予权重 w。

求值函数。查询由单个递归求值函数 E 执行：$\mathcal{X} \times \mathcal{W} \to \mathcal{W}$，其中 $\mathcal{X}$遍历语法所定义的格式良好的语法对象（例如查询、步骤和步骤组件）。

执行从根节点初始化为 $\mathsf{E}(Q,\ \{(r,\ 1)\})$。随后进行查询级递归：

$$\mathsf{E}(Q, W) = \begin{cases} W, & \text{if } Q = \varnothing, \\ \mathsf{E}(S, W), & \text{if } Q = S\varnothing, \\ \mathsf{E}\big(Q', \mathsf{E}(S, W)\big), & \text{if } Q = SQ'. \end{cases}$$

步骤 S 和步骤组件（A，N，R，P）的其余情况定义见附录 A。

ACL 旅行示例（续）。考虑 SE- MANTIC XPATH 查询 Q_1：

```
//天[avg(兴趣点[节点~="conference"])]
```

Q_1 被解析为一个后缀为空的单步骤查询，$Q_1 = S\varnothing$其中该步骤 $S = ANP$带有 $A = //$、$N =$ 天、 $P = avg(S')$以及$S' =$兴趣点[节点~="conference"]。随后按如下步骤执行（图 2）：

1. **初始化：** 令 $W_0 = \{(r,\ 1)\}$。将执行初始化为$\mathsf{E}(Q,\ W_0)$。2. **化简查询：** $\mathsf{E}(Q,\ W_0) = \mathsf{E}(S,\ W_0)$3. **应用轴导航器：** 令 $W_1 = \mathsf{E}(//\ W_0)$

$$\begin{aligned} \mathsf{E}(Q, W_0) &= \mathsf{E}\big(NP,\ \mathsf{E}(//, W_0)\big) \\ &= \mathsf{E}(NP, W_1). \end{aligned}$$

这里，轴导航器将根节点映射到其所有后代节点。

4. **应用节点选择器：** 令 $W_2 = \mathsf{E}(天\ W_1)$

$$\begin{aligned} \mathsf{E}(NP, W_1) &= \mathsf{E}\big(P,\ \mathsf{E}(Day, W_1)\big) \\ &= \mathsf{E}(P, W_2). \end{aligned}$$

在此，节点选择器将根节点的后代节点限制为天节点。

5. **应用语义相关性：**

$$\mathsf{E}(P, W_2) = \mathsf{E}\big(avg(S'), W_2\big).$$

6. **评估内部递归：** $\mathsf{E}\big($平均每个天级别的兴趣点计算相关性分数 d（即其子节点兴趣点与会议的相关程度）。对于第 *2* 天，兴趣点的语义相关性分数分别为 0.603、0.482 和 0.608，针对兴趣点[节点~=“会议”]，，其平均值给出的天级别分数为 0.565。

7. **终止：** 递归在基本情形 $\mathsf{E}(\varnothing, \cdot)$ 处结束，选出包含三项会议活动的第 *2* 天 作为排名最高的匹配项。

4 评估

我们从两项指标评估 SEMANTIC XPATH 与两种基线方法（上下文内存和检索增强生成）相比的表现：（1）**平均通过率**，即系统检索到正确结构化数据并生成满足真实约束的输出的测试请求所占比例；（2）**令牌用量**，即每个请求的大语言模型令牌成本总量。

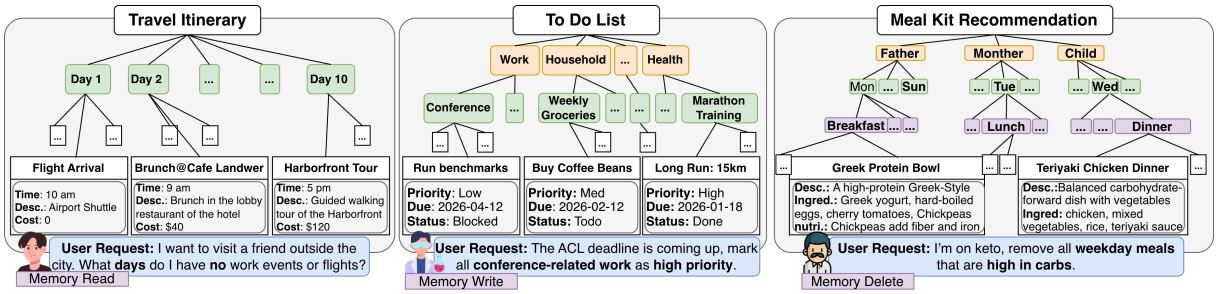


图4：三个对话式人工智能场景中的结构化记忆示例：旅行行程、待办清单和膳食套餐推荐，以及涵盖记忆读取、记忆写入和记忆删除的代表性用户请求。

研究问题1：在单轮静态结构化内存快照设置下，SEMANTICXPATH 在各种对话式人工智能任务（例如，图4）中与现有方法相比表现如何？

研究问题 2：在需要动态维护记忆的多轮设置中，各方法在需要跨版本检索修订历史的交互中的表现如何？轮数又会如何影响令牌用量？

4.1 实验设置

评估领域。我们考察三个涉及结构化长期记忆的对话式人工智能场景，各场景的简化结构见图4。

- **旅行行程**：为期七天的旅行计划，每天包含餐厅和兴趣点选项，并标注描述、预估费用和偏好。
- **待办清单**：结构化待办清单，任务按 5 个类别（如工作、家务和个人事务）进行组织。每个类别包含相关项目，这些项目由若干任务组成，并标注描述、进度状态、优先级和截止日期。
- **餐盒推荐**：面向家庭的7天膳食计划，每位成员在每个用餐时段可获得三个选项，每个选项均标注食材与营养。

每个领域都有一个领域专属模式和一个精心整理的交互集，由20个单轮对话和5个多轮对话组成。

配置。我们使用 GPT-5 mini 和 Gemini 3 Flash 作为大语言模型骨干来评估所有方法。对于平面检索增强生成，我们使用 Qwen3-Embedding-8B 嵌入模型。

对于 SEMANTIC XPATH，我们考虑两种语义相关性评分器：（1）基于语义相似度的评分器，它使用 Qwen3-Embedding-8B 嵌入计算节点内容与语义相关性条件之间的余弦相似度；（2）基于蕴含关系的评分器，它使用 facebook/bart-large-mnli 估计节点内容蕴含相关性条件的概率。

4.2 评估结果

研究问题1：单轮评估。我们报告了跨三个领域平均的单轮对话结果（图 5，详见附录B中的领域专属结果）。如图 5左侧所示，SEMANTIC XPATH的通过率与上下文内方法相当，二者均显著优于平面检索增强生成基线。

平面检索增强生成往往返回缺乏层级推理的表层语义匹配。例如，对于请求“天气预报显示第 7 天有大雨。哪些活动是户外的？”，它会检索整个行程中的户外兴趣点，而不是将结果限定在第 7 天。

上下文内方法的令牌用量大约比另外两种方法高 5×（右侧图 5）。因此，我们可以得出结论：SEMANTIC XPATH 的效果与上下文内基线相当，同时所需令牌大幅减少。

研究问题 2：多轮评估。在图 6中，对于需要访问先前对话历史的请求，上下文内基线的通过率有所下降，因为上下文越长、越复杂，就越容易误判相关信息。相比之下，SEMANTIC XPATH 仅检索相关的结构化数据，使通过率保持在接近单轮设置的水平。

在图 7中，我们评估了各轮对话中的令牌用量。

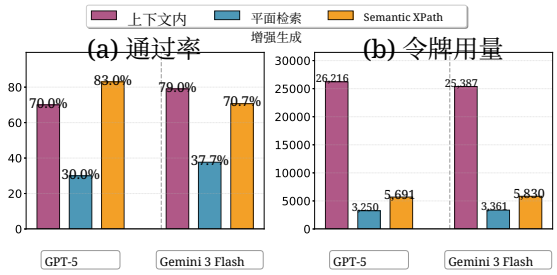


图 5：三种方法的单轮评估。我们报告了三个领域的平均通过率（**左侧**）和令牌用量（**右侧**）的平均值。

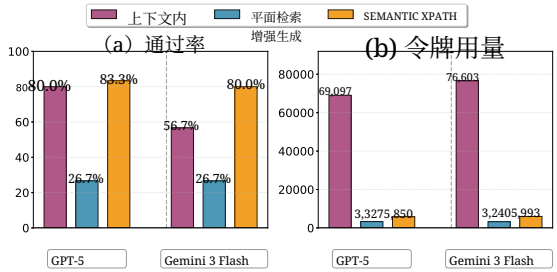


图 6：三种方法的多轮评估。我们报告三个领域的平均通过率（**左**）和令牌用量（**右**）的平均值。

上下文内方法会在每一轮将用户请求和智能体响应追加到输入中，因此令牌用量持续增加；而 SEMANTIC XPATH 会过滤掉与当前请求无关的信息，从而保持令牌用量稳定。

5 相关工作

外部语料库与对话记忆。关于大语言模型智能体记忆的先前研究，通常将记忆视为一个外部语料库，即独立于对话而存在的静态文档或段落集合（Sarthi 等人，2024 年；Chen 等人，2023 年；Li 等人，2025 年；Gupta 等人，2025 年；Wen 等人，2024 年，2025 年；Liu 等人，2025 年）。这种设定不同于对话记忆：在对话记忆中，记忆体现智能体当前执行某项任务的工作状态，并通过交互不断更新，因此本质上具有动态性和可变性（Sun 和 Zeng，2025 年；Rezazadeh 等人，2024 年）。我们关注的是对话记忆。

扁平与结构化对话记忆。在对话记忆领域，早期方法主要将交互历史存储为扁平的文本记录，并采用轻量级格式，或以压缩摘要的形式记录过往交互（Zhong 等人，2024 年；Wang 等人，2025 年；Packer 等人，2023 年；Lu 等人，2023 年）。近期研究则引入了结构化

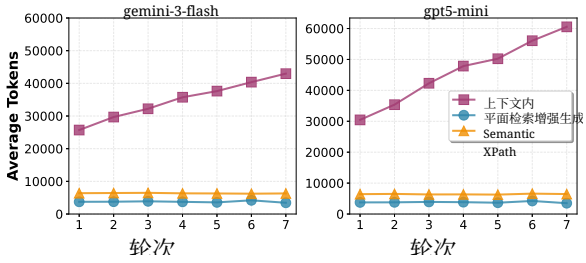


图 7：多轮对话式人工智能场景中令牌用量与轮数的关系。我们提出的 SEMANTIC XPATH 在各轮中保持令牌消耗稳定，而上下文内基线的令牌用量则随着对话历史增长而持续增加。

记忆表征，将信息组织成层级结构，以支持跨越多个抽象层级的交互（Sarthi 等人，2024 年；Chen 等人，2023 年；Rezazadeh 等人，2024 年；Sun 和 Zeng，2025 年；Liet al.，2025 年；Guptaetal.，2025 年）。我们的工作聚焦于结构化对话记忆。

抽象式与组合式结构化记忆层级结构。结构化对话记忆可以组织成抽象式层级结构，其中高层级单元是对低层级内容的有损摘要；也可以组织成组合式层级结构，以结构化方式组织与任务相关的信息（例如，行程 → 天 → 活动）。虽然以往大多数方法都侧重于抽象式层级结构（Sarthi 等人，2024 年；Chen 等人，2023 年；Rezazadeh 等人，2024 年；Sun 和 Zeng，2025 年；Li 等人，2025 年；Gupta 等人，2025 年），组合式层级结构在面向任务的对话式人工智能中往往更为自然，因为它们反映了交互和任务的层级表示，且每个层级都存储不同的信息（例如，行程与各项活动）。因此，我们聚焦于**组合式层级结构**。

6 结论

本文提出了 SEMANTIC XPATH，这是一种面向对话式人工智能智能体的树形结构化记忆模块。实验表明，与平面检索增强生成基线相比，SEMANTIC XPATH 的表现更出色，同时使用的令牌远少于上下文内存。我们还推出了 SEMANTICXPATH CHAT，这是一个用于 SEMANTIC XPATH 的实时演示，可将结构化记忆和查询执行的细节可视化。除了为行程规划等面向任务的场景提供引人入胜的长期用户体验外，SEMANTIC XPATH 还是一个实用框架，可帮助对话式人工智能开发者构建更高效的结构感知记忆系统。

伦理考量

我们提出的方法 SEMANTIC XPATH，是一种面向对话式人工智能系统的树形结构化记忆模块。与任何基于大语言模型的方法一样，该系统可能会继承底层大语言模型的偏见、事实错误或推理错误。此外，尽管我们采用的结构化检索方法提升了效率和相关性，但并不能保证事实准确性。因此，在高风险或安全关键的场景中，用户应避免过度依赖生成的输出。

参考文献

Howard Chen、Ramakanth Pasunuru、Jason Weston 和 Asli Celikyilmaz。2023 年。走入记忆迷宫：通过交互式阅读突破上下文限制。*arXiv* 预印本 *arXiv:2310.05029*。

James Clark 和 Steve DeRose。1999。XML 路径语言（XPath）版本 1.0。万维网联盟推荐标准。

Nilesh Gupta、Wei-Cheng Chang、Ngot Bui、Cho-Jui Hsieh 和 Inderjit S Dhillon。2025 年。大语言模型引导的层次化检索。*arXiv* 预印本 *arXiv:2510.13217*。

Patrick Lewis、Ethan Perez、Aleksandra Piktus、Fabio Petroni、Vladimir Karpukhin、Naman Goyal、Heinrich Küttler、Mike Lewis、Wen-tau Yih、Tim Rocktäschel 等 1 人。2020年。面向知识密集型自然语言处理任务的检索增强生成。神经信息处理系统进展，33:9459–9474。

Rui Li、Zeyu Zhang、Xiaohe Bo、Zihang Tian、Xu Chen、Quanyu Dai、Zhenhua Dong 和 Ruiming Tang。2025 年。CAM：一种关于基于大语言模型的阅读理解中智能体记忆的建构主义视角。载于神经信息处理系统进展（神经信息处理系统大会）。

Nelson F Liu、Kevin Lin、John Hewitt、Ashwin Paranjape、Michele Bevilacqua、Fabio Petroni 和 Percy Liang。2024 年。迷失在中间：语言模型如何利用长上下文。计算语言学协会汇刊，12：157–173。

Yifan Liu、Qianfeng Wen、Jiazhou Liang、Mark Zhao、Justin Cui、Anton Korikov、ArminToroghi、Junyoung Kim 和 Scott Sanner。2025 年。通过结合大语言模型相关性判断的高斯过程回归进行自然语言推荐的多模态条目评分。预印本，arXiv：2510.22023。

Junru Lu、Siyu An、Mingbao Lin、Gabriele Pergola、Yu- lan He、Di Yin、Xing Sun 和 Yunsheng Wu。2023 年。MemoChat：调整大语言模型，使其利用备忘录进行一致的长程开放域对话。预印本，arXiv:2308.08239。

Charles Packer、Vivian Fang、Shishir_G Patil、Kevin Lin、Sarah Wooders 和 Joseph_E Gonzalez。2023 年。Memgpt：迈向作为操作系统的大语言模型。

Alireza Rezazadeh、Zichao Li、Wei Wei 和 Yujia Bao。2024 年。从孤立对话到层次化模式：面向大语言模型的动态树记忆表示。*arXiv*预印本 *arXiv:2410.14052*。

Parth Sarthi、Salman Abdullah、Aditi Tuli、Shubh Khanna、Anna Goldie 和 Christopher D Manning。2024 年。RAPTOR：用于树组织检索的递归抽象处理。发表于第十二届学习表征国际会议。

Haoran Sun 和 Shaoning Zeng。2025 年。用于大语言模型智能体高效长期推理的分层记忆。*arXiv* 预印本 *arXiv:2507.22925*。

Qingyue Wang、Yanhe Fu、Yanan Cao、Shuai Wang、Zhiliang Tian 和 Liang Ding。2025 年。递归摘要使大语言模型具备长期对话记忆。*Neurocomputing*，639:130193。

Qianfeng Wen、Yifan Liu、Justin Cui、Joshua Zhang、Anton Korikov、George-KirollosSaad 和 Scott San- ner。2025 年。一种简单而有效的自然语言推荐阐述式查询重构方法。预印本，arXiv:2510.02656。

Qianfeng Wen、Yifan Liu、Joshua Zhang、George Saad、Anton Korikov、Yury Sambale 和 Scott Sanner。2024 年。面向旅行目的地推荐中广泛且间接查询的阐述式子主题查询重构。预印本，arXiv:2410.01598。

Xuechen Zhang、Xiangyu Chang、Mingchen Li、Amit Roy-Chowdhury、Jiasi Chen 和 Samet Oymak。2024 年。选择性注意力：通过有原则的上下文控制增强 Transformer。神经信息处理系统进展，37:11061– 11086。

Yizhe Zhang、Siqi Sun、Michel Galley、Yen-Chun Chen、Chris Brockett、Xiang Gao、Jianfeng Gao、Jingjing Liu 和 William B Dolan。2020年。DialoGPT：面向对话回复生成的大规模生成式预训练。载于计算语言学协会第*58*届年会论文集：系统演示，第270–278页。

Wanjun Zhong、Lianghong Guo、Qiqi Gao、He Ye 和 Yanlin Wang。2024 年。MemoryBank：通过长期记忆增强大型语言模型。载于 *AAAI* 人工智能会议论文集。

A 查询执行

A.1 步骤求值

$$\mathsf{E}(S,W)=\begin{cases}\mathsf{E}\big(NRP,\,\mathsf{E}(A,W)\big), & \text{if } {}^{S=}_{ANRP},\\ \mathsf{E}\big(NP,\,\mathsf{E}(A,W)\big), & \text{if } {}^{S=}_{ANP},\\ \mathsf{E}\big(NR,\,\mathsf{E}(A,W)\big), & \text{if } {}^{S=}_{ANR},\\ \mathsf{E}\big(N,\,\mathsf{E}(A,W)\big), & \text{if } {}^{S=}_{AN}.\end{cases}$$

A.2 轴选择器求值

定义轴扩展运算符:

$$\mathsf{E}(/,W)=\Big\{(u,w)\;\Big|\;{}^{\exists(v,w)\in W,}_{u\in\mathsf{C}(v)}\Big\},$$
$$\mathsf{E}(//,W)=\Big\{(u,w)\;\Big|\;{}^{\exists(v,w)\in W,}_{u\in\mathsf{D}(v)}\Big\}.$$

其中，$\mathsf{C}(v)$ 和 $\mathsf{D}(v)$ 分别表示节点 v 的子节点集合和后代节点集合。

轴选择器的求值函数定义如下:

$$\mathsf{E}(A,W)=\begin{cases}\mathsf{E}(/,W), & \text{if } A=/,\\ \mathsf{E}(//,W), & \text{if } A=//.\end{cases}$$

A.3 节点选择器求值

定义节点选择器运算符:

$$\mathsf{E}(\mathsf{NodeType},W)=\Big\{(u,w)\;\Big|\;{}^{(u,w)\in W,}_{\kappa(u)=\mathsf{NodeType}}\Big\},$$
$$\mathsf{E}(*,W)=W.$$

其中，$\kappa(u)$ 返回 u 的节点类型。

节点选择器的求值函数定义如下:

$$\mathsf{E}(N,W)=\begin{cases}\mathsf{E}(\mathsf{NodeType},W), & \text{if } {}^{N=}_{\mathsf{NodeType}},\\ \mathsf{E}(*,W), & \text{if } N=*.\end{cases}$$

A.4 位置选择器求值

定义位置选择器运算符:

$$\mathsf{E}([i],W)=[i](W),$$
$$\mathsf{E}([-i],W)=[-i](W),$$
$$\mathsf{E}([i{:}j],W)=[i{:}j](W),$$

其中，$[i](\cdot)$、$[-i](\cdot)$ 和 $[i{:}j](\cdot)$ 按拓扑序对 W 执行位置选择。

位置选择器的求值函数定义如下:

$$\mathsf{E}(R,W)=\begin{cases}\mathsf{E}([i],W), & \text{if } R=[i],\\ \mathsf{E}([-i],W), & \text{if } R=[-i],\\ \mathsf{E}([i{:}j],W), & \text{if } R=[i{:}j].\end{cases}$$

A.5 语义相关性算子求值我们定义一个语义相关性评分函数 $\mathsf{Rel}:\ V\times\Phi\to[0,1]$，它会根据语义相关性条件 $P\in\Phi$为节点 u 分配一个分级相关性分数。

局部语义相关性。定义局部语义相关性的相关性评分函数:

$$\mathsf{Rel}(u,\mathtt{Local})=\mathsf{Atom}_{\mathrm{loc}}(u,\mathtt{Local}),$$

其中，$\mathsf{Atom}_{\mathrm{loc}}(u,\text{局部})\in[0,1]$ 通过语义评分函数，根据特定属性或完整节点表示进行计算。

聚合语义相关性。若 P :=Agg(S')，其中内层步骤 S' 的表面形式为 $S':=N\ R\ P'$，令 $\mathcal{E}^{(u)}_{S'}\subseteq V$ 表示从 u对 S' 进行结构化执行后得到的证据节点集。聚合相关性评分函数递归定义如下:

$$\mathsf{Rel}(u,\mathsf{Agg}(S'))=\mathsf{Agg}\Big(\{\,\mathsf{Rel}(x,P')\mid x\in\mathcal{E}^{(u)}_{S'}\,\}\Big),$$

其中，Agg 是一种聚合算子，例如 min、max、avg 或 gmean。

语义相关性算子的求值函数定义如下:

$$\mathsf{E}(P,W)=\{(u,\ w\cdot\mathsf{Rel}(u,P))\mid(u,w)\in W\}.$$

组合语义相关性。对于一元算子 $1-P$，我们定义

$$\mathsf{Rel}(u,\,1-P)=1-\mathsf{Rel}(u,P),$$

及相应的语义相关性算子

$$\mathsf{E}(1{-}P,W)=\Big\{(u,\ w{\cdot}\mathsf{Rel}(u,1{-}P))\;\Big|\;{}_{(u,w)\in W}\Big\}.$$

对于二元组合条件，设 $\odot\in${平均值、乘积、最小值、最大值} 为$[0,1]$, 上的二元算子，对于任意节点 u 和相关性条件P_1、P_2，定义

$$\mathsf{Rel}(u,\,P_1\odot P_2)=\odot\big(\mathsf{Rel}(u,P_1),\,\mathsf{Rel}(u,P_2)\big).$$

相应的组合式语义相关性算子为

$$\mathsf{E}(P_1{\odot}P_2,W)=\Big\{(u,\ w{\cdot}\mathsf{Rel}(u,P_1{\odot}P_2))\;\Big|\;{}_{(u,w)\in W}\Big\}.$$

语义相关性运算符的求值函数定义如下:

$$\mathsf{E}(P,W)=\begin{cases}\mathsf{E}(\mathsf{Local},W), & \text{if } {}^{P=}_{\mathsf{Local}},\\ \mathsf{E}(\mathsf{Agg}(S'),W), & \text{if } {}^{P=}_{\mathsf{Agg}(S')},\\ \mathsf{E}(1-P',W), & \text{if } {}^{P=}_{1-P'},\\ \mathsf{E}(P_1\odot P_2,W), & \text{if } {}^{P=}_{P_1\odot P_2},\end{cases}$$

模型	方法	评分器	行程		待办清单		食材包		Avg.
			Pass	令牌通过		令牌通过		令牌通过		令牌
GPT-5 mini	上下文内	–	55.0	12,136	80.0	5,444	75.0	61,069	70.0	26,216
	平面检索增强生成	–	25.0	3,137	40.0	2,722	25.0	3,892	30.0	3,250
	语义 XPath *Sem*		75.0	5,548	64.0	5,937	60.0	7,013	66.3	6,166
	Semantic XPath	蕴含	85.0	5,161	84.0	5,184	80.0	6,728	83.0	5,691
Gemini 3 Flash	上下文内	–	95.0	12,846	72.0	10,465	70.0	52,849	79.0	25,387
	平面检索增强生成	–	35.0	3,349	48.0	2,934	30.0	3,800	37.7	3,361
	语义 XPath 语义		65.0	5,454	68.0	5,710	60.0	6,966	64.3	6,043
	语义 XPath	蕴含 75		5,240	72.0	5,605	65.0	6,644	70.7	5,830

表 1：不同方法的单轮评估。语义 表示 SEMANTIC XPATH 的语义相似度评分器，蕴含 表示其蕴含评分器。我们报告了在 GPT-5 mini 和 Gemini 3Flash 作为骨干模型时，三个领域——旅行行程、待办清单和膳食套餐推荐——的平均通过率和令牌用量。

模型	方法	评分器	行程		待办清单		食材包		Avg.
			Pass	令牌	Pass	令牌通过		令牌通过		令牌
GPT-5 mini	上下文内	–	90.0	28,342	70.0	20,398	80.0	158,552	80.0	69,097
	平面检索增强生成	–	20.0	3,250	50.0	3,043	10.0	3,686	26.7	3,327
	语义 XPath 语义		50.0	5,664	100.0	6,132	60.0	7,087	70.0	6,294
	语义 XPath	蕴含	80.0	5,820	100.0	5,649	70.0	6,079	83.3	5,850
Gemini 3 Flash	上下文内	–	60.0	31,639	70.0	23,602	40.0	174,567	56.7	76,603
	平面检索增强生成	–	20.0	3,164	50.0	2,959	10.0	3,596	26.7	3,240
	语义 XPath 语义		60.0	5,451	100.0	5,872	70.0	6,954	76.7	6,092
	语义 XPath	蕴含	60.0	5,671	100.0	5,526	80.0	6,782	80.0	5,993

表 2：各方法的多轮评估。语义 表示语义相似度评分器，蕴含 表示 SEMANTIC XPATH 的蕴含评分器。我们报告了在 GPT-5 mini 和 Gemini 3Flashas 两种骨干模型下，三个领域——旅行行程、待办清单和膳食套餐推荐——的平均通过率和令牌用量。

B 补充结果

在单轮和多轮结果中（表 1 和 表 2），上下文内存均取得了较高的通过率，但令牌用量非常高，尤其是在配餐套装推荐查询中。SEMANTIC XPATH 在保持具有竞争力的准确率的同时，大幅降低了令牌成本；对于 GPT-5 mini 和 Gemini 3 Flash，蕴含评分器的通过率始终优于语义相似度评分器。平面检索增强生成仍是成本最低的基线，但通过率明显较低。

C 案例研究

除了图 2所示的单轮请求外，真正的对话式人工智能系统还必须支持长期、面向任务的交互。用户可能在多轮对话中修改计划，之后再返回到早期版本，甚至切换到不同的计划。关键挑战在于，尽管编辑持续进行、历史记录不断增长，系统能否在不同修订版本和计划中可靠地检索到正确的信息。

下面，我们通过一个示例计划来演示一个案例在相同的 ACL 旅行示例中，用户提出的计划修订请求：

ACL 旅行示例（续）我们展示两个与修订历史交互的请求。
请求 1：“取消参观海报会议的安排，因为我需要参加客户会议。”
查询：

```
//兴趣点[节点~="海报"]
```

省略了 10 条后续请求： · · ·
请求2：“等等，海报会议的时间是什么来着？我也许能赶上。”
查询：

```
//版本[节点~="删除海报会议"]//兴趣点[节点~="海报"]
```

结果： SEMANTIC XPATH 检索删除编辑所创建的版本，从该修订上下文中找回被移除的海报会议条目，并返回其预定时间。相比之下，上下文较长时，上下文内存会失效。