arXiv:2611.33124v1 [cs.CL] 24 Nov 2026

SAG-Mem-1.0 技术报告

Yuchao Wu*, **XingCheng Liang, Xuanze Wang,Yini Huang, Yue Wu**Zleap AIjomy@zleap.com, lensen@zleap.com, wangxuanze@zleap.com, huangyini@zleap.com, wuyue@zleap.com

摘要

长期对话记忆需要检索散布于各个对话会话中的互补事实，包括与原始问题相似度有限的中间证据。现有方法通常依赖派生的记忆表示或持久化关系结构，这会引入构建成本，并可能丢失源数据细节。我们提出 SAG-Mem，这是一种对话记忆框架，通过在互不重叠的文本块中保留完整的对话消息，并通过查询时词汇扩展发现互补证据。记忆构建无需模型推理，而是使用词法索引，不进行基于大语言模型的抽取、摘要或语义关系构建。在查询时，BM25 检索初始候选项，证据评估器识别有用的文本块，选定证据中的词汇线索则引导额外检索轮次。检索到的对话还会进一步标注其中所支持的相对时间表达式的日历解释，在保留原始措辞的同时保留时间粒度。我们还提出 SAG-Mem-Lite，这是一种替代配置，以基于嵌入的候选评分取代证据评分；在两个基准测试中，与完整上下文和所有基于记忆的基线相比，该配置均实现了更低的运营成本和较低的查询延迟。在 LoCoMo 和 LongMemEval 上，SAG-Mem 的总体得分分别为 83.08 和 80.20，在两个基准测试中均为所评估方法中的最高分。在 LongMemEval 上，其得分超过最强对比基线 7.20 分。SAG-Mem-Lite 的得分分别为 77.09 和 77.20。这些结果表明，对话保留与证据引导的检索扩展是长期对话记忆的一种有效设计。[1]

关键词 长期对话记忆 · 检索增强生成 · 证据引导检索 ·词汇查询扩展 ·时间锚定

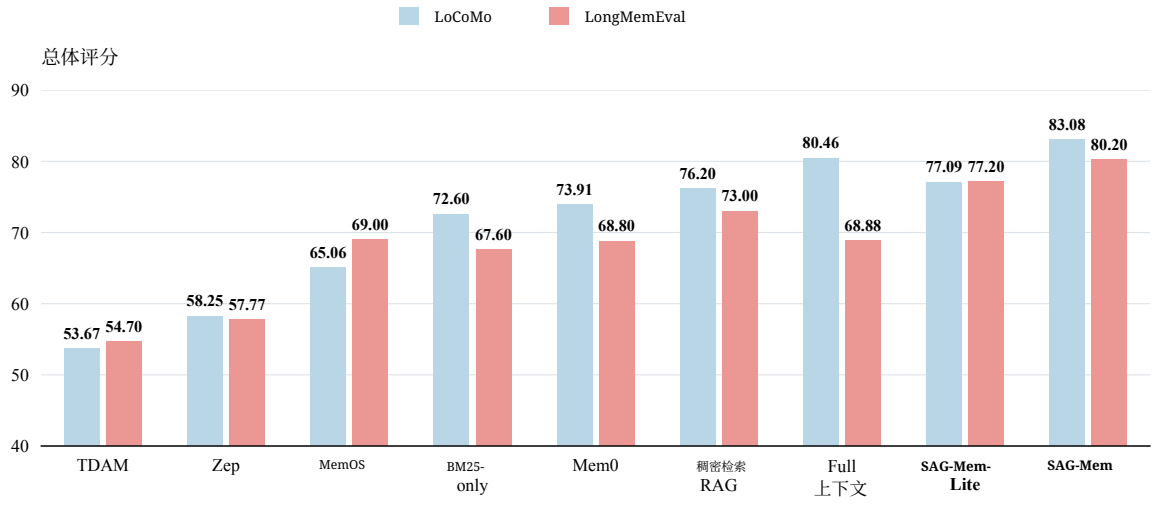

图 1：以 Qwen3.6-35B 为骨干模型时，不同记忆方法在 LoCoMo 和 LongMemEval 上的总体评分（%）。SAG-Mem 在两个基准测试中均取得最高总体评分，在 LoCoMo 上达到 83.08%，在 LongMemEval 上达到 80.20%。

*联系方式： jomy@zleap.com。[1]基准测试与评估代码：
https://github.com/Zleap-AI/SAG-Mem-Benchmark

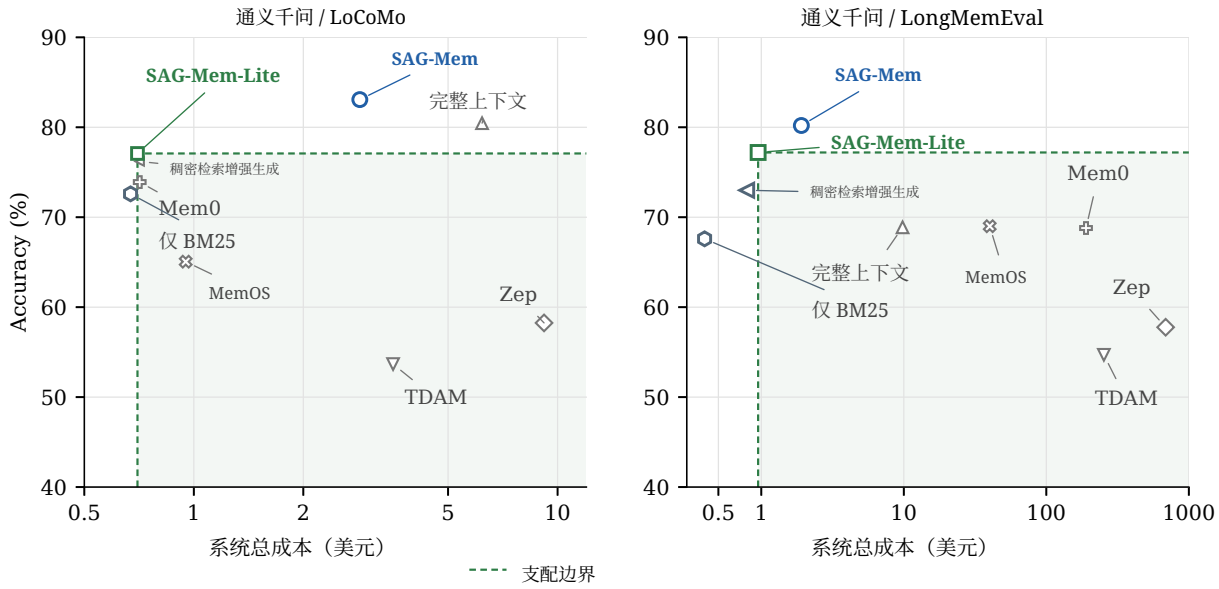

图 2：在 Qwen3.6-35B-A3B-FP8 下，使用表3和5中的数值绘制的总体准确率与总运行成本关系图。绿色虚线表示 SAG-Mem-Lite 的成本和准确率，阴影区域突出显示成本更高但准确率不高于该方法的方法。两个面板均使用对数成本坐标轴。

1 引言

长期对话记忆使语言模型能够利用过往交互中积累的信息回答问题 [Packer 等人， 2023]。随着对话历史不断增长，检索增强生成（RAG）提供了一种访问相关信息的方式，而无需将完整历史置于模型的上下文中 [Lewis 等人， 2020]。然而，回答一个问题可能需要分布于多个会话中的互补事实，以及与之相关的对话与时间上下文 [Maharana 等人， 2024年， Wu 等人， 2025]基于与原始问题相似度的检索可能会遗漏中间证据，而这类证据的相关性只有在找到其他事实后才会显现 [Qi 等人，2019年，Trivedi 等人，2023]。这促使我们利用检索到的证据指导进一步搜索，超越初始候选池。

记忆系统通过采用不同的表示形式和组织方式来解决这一问题。Memory-Bank [Zhong 等人生成事件摘要和用户画像，而 RMM [Tan 等人在多个粒度上总结交互内容。Mem0 [Chhikara 等人、 2025] 提取并整合关键信息，形成持久记忆条目。在关系组织方面，A-MEM [Xu 等人、2026a年] 将记忆笔记相互关联，并更新其上下文表示；而 Zep [Rasmussen 等人、 2025] 则将提取出的实体和事实组织到时序知识图谱中。这些表示有助于检索，但生成和更新它们需要模型推理，并会在记忆摄入过程中产生持续的 token成本。此外，这些表示中被省略的细节，之后可能仍是回答问题或跨会话关联证据所必需的。近期关于 Raw-RAG 的结果 [Bao 等人、 2026] 表明，在原始对话历史上进行稠密检索和重排序，能够优于所比较的智能体记忆系统，这凸显了保留源信息以及推导答案所需支持关系的重要性。这些观察结果促使我们采用一种设计：直接存储源对话，并利用检索证据引导后续搜索，以获取互补信息。

使用小型阅读器模型时，时间解释也是一个值得关注的问题 [Chu 等人，2024年，Fatemi 等人， 2024]。诸如 昨天 和 上个月 这样的表达，其所指取决于原始消息的日期，而非当前问题的日期 [Rasmussen 等人， 2025]。即使检索到了正确的证据，阅读器也必须确定恰当的参考日期，并在回答前转换相对时间表达。时间归一化提供了一种成熟的方法，可通过标准化的日历表示明确表达此类内容 [Strötgen 和 Gertz，2010年，Chang 和 Manning， 2012]。这促使我们为受支持的表达提供明确的日历解释，以减少阅读器所需进行的时序推理。

这些考量引出了三个研究问题。

RQ1：记忆构建。 在跨越多个会话的长对话中，直接存储对话能否在保留准确问答所需原始细节的同时降低成本?

RQ2：证据发现。 证据引导的词汇扩展能否跨会话找回互补事实，包括与原始查询匹配度较低、但推导答案所必需的中间证据?

RQ3：时序解释。 通过根据源消息日期解析相对表达式，确定性日期标注能否帮助小型阅读器模型提升时序问答能力?

基于 SAG2 [Wu， 2026], 我们提出了 **SAG-Mem**，将证据引导的词汇检索应用于对话记忆。它在每个会话内以互不重叠的文本块为单位索引完整消息，在不进行模型推理的情况下保留说话人、日期和来源标识。查询时，BM25 [Robertsonand Zaragoza， 2009] 检索候选项，JEV [Almeida， 2026] 对其在回答问题中的贡献进行评分。从高评分文本块中确定性选出的术语会驱动一次额外的 BM25 搜索，随后在共享证据上下文中，由 JEV 对保留的候选项和新候选项进行重新评分。

SAG-Mem-Lite 沿用相同的文本块和词汇扩展方式，并以使用 bge-large-en-v1. 计算的查询–分块余弦相似度取代 JEV。5 [Xiao 等人， 2024]。它会为每个文本块预先计算一个嵌入，并支持最多两轮扩展。两种配置都会返回带有相对时间表达式日期标注的源对话。系统会指示阅读器依据这些证据作答，并在证据不足时弃答。

SAG-Mem 和 SAG-Mem-Lite 的主要贡献可概括如下:

- 我们提出了一种对话保留式记忆设计，避免使用基于大语言模型的抽取和持久语义关系构建；同时通过日期标注明确呈现有依据支持的相对时间表达式，并保留原文和时间粒度。
- 我们开发了基于种子的词汇扩展方法，以发现初始检索候选池之外的补充证据；在 SAG-Mem 中使用基于 JEV 的证据评估，在 SAG-Mem-Lite 中使用基于嵌入的候选评分。
- 我们在 LoCoMo 和 LongMemEval 上评估了两种配置。如图 1所示，SAG-Mem 在两个基准测试上的总体评分分别为 83.08 和 80.20，在所有受评估的方法中均排名最高；SAG-Mem-Lite 的评分分别为 77.09 和 77.20。

2 问题陈述

我们研究基于过往交互历史的长期对话式问答。设 $\mathcal{H} = (S_1, \ldots\ldots, S_N)$ 表示由 N 个按时间顺序排列的会话组成的历史，其中每个会话 $S_i = (m_{i,\,1}, \ldots\ldots, m_{i,\,n_i})$ 都包含一系列带有相关日期信息的消息。每条消息都包含其原始文本、说话人和来源标识。给定关于 $\mathcal{H}$的问题 q，任务是生成有记录的交互内容作为依据的答案。所需证据可能跨越多个会话，涉及中间事实，并取决于源消息的时间语境。

在我们的设定中，每条记忆记录都是一个对话块，包含来自单个会话的连续完整消息。这些文本块互不重叠，并保留原始消息文本和元数据。设 $\mathcal{C}$ 表示由此得到的来源块集合。记忆系统根据这些文本块构建记忆存储库 $\mathcal{M} = \text{Build}(\mathcal{H})$。给定证据预算 K，系统从 $\mathcal{C}$中检索一个有序的文本块列表:

$$\mathcal{E}_q = \text{Retrieve}(q, \mathcal{M}), \qquad |\mathcal{E}_q| \le K, \tag{1}$$

其中 K 限制传递给阅读器的文本块数量。

让 $g(\mathcal{E}_q)$ 渲染选定的源对话及其说话人、日期、会话和消息元数据。此渲染还会附加受支持的相对时间表达式的日历解释，使用相关源消息日期作为参考点，同时保留原始措辞和时间粒度。阅读器生成答案，具体如下

$$\hat{a} = \text{Reader}(q, g(\mathcal{E}_q))\,. \tag{2}$$

检索目标是在 q 预算范围内 K找回能够共同支持回答问题的互补证据。这包括连接跨会话信息或提供定位更多证据线索的中间事实，即使它们与原始问题的直接相似度有限。答案应与源对话、相关时间条件以及证据所支持的精确程度保持一致。当检索证据不足时，包括所请求的信息不在 $\mathcal{H}$中的情况，预期的回应是承认现有信息不足。

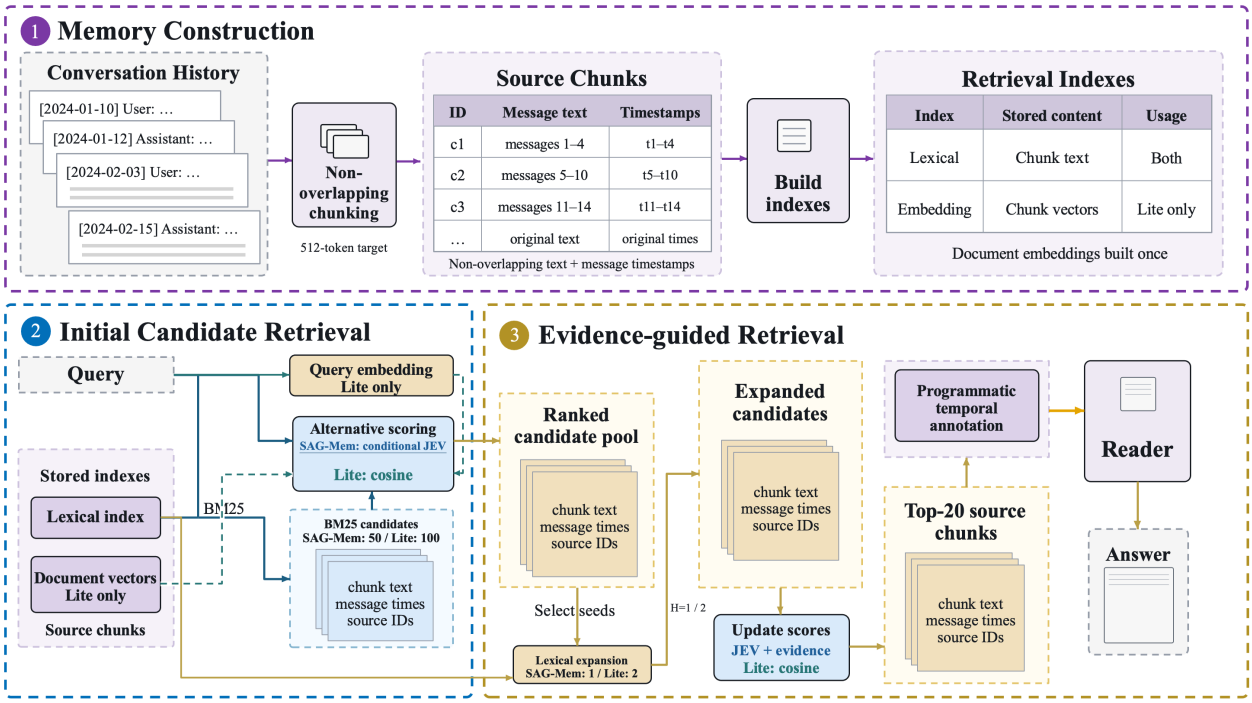

图3：流水线概览。完整消息被分组成源链接分块，检索证据引导进一步的词汇搜索，选定的消息在答案生成之前会接受确定性时间澄清。标记为“Lite”的元素属于第 3.5 节所述的变体。每个文本块中的消息数量仅用于说明；文本块长度遵循第 3.1 节的512令牌预算。

表1：各阶段如何应对第 1 节提出的问题。

问题	阶段	设计选择
RQ1	记忆构建	完整消息逐字存储，并采用追加写入
RQ2	初始检索和证据引导检索	根据所选文本块中的词语进行第二轮词汇搜索，此外
	检索	针对所收集证据进行重新评分
RQ3	答案生成	检索后进行确定性的日期标注，同时保留源文本措辞

3 SAG-Mem框架

第 1 节促成了三项设计决策：存储完整对话而非提取的记忆，在查询时发现跨会话链接，以及在读取前澄清相对时间表达式。这三项决策都遵循同一原则：将每项解释行为推迟到所需信息变得可用时再进行。在写入时，问题尚未知晓，因此对话以逐字形式存储。在查询时，迄今检索到的证据表明下一步应查找什么，因此在此时发现事实之间的链接。在读取时，已知阅读器将看到哪些消息，因此仅针对这些消息在此时解析相对时间。

本节按照流水线顺序（图2 ）实例化第 2 节中的三个组件。第3节3.1介绍构建；第3.2 和3.3 介绍公式（1）中的检索；第3.4 介绍公式（ g 2）中的渲染和阅读器。表1总结了各阶段所解决的问题。随后，第3.5 介绍 SAG-Mem-Lite：它保留存储库和检索流程，但替换评分模型并调整搜索预算。

3.1 记忆构建

记忆构建实现了 构建。基于提取的记忆会在写入时决定保留哪些细节 [Zhong 等人，2024年，Chhikara 等人，2025]。由于系统无法预知之后的问题需要什么，SAG-Mem 不做此类判断，而是按对话原本的说法存储内容。

完整消息块。 我们按源顺序处理 $\mathcal{H}$，并将连续消息分组成文本块，每个文本块都位于单个会话内，且彼此不重叠。打包时遵循 512 个令牌的软性预算：如果添加下一条完整消息会超出预算，则先关闭当前文本块。令牌数使用 bge-large-en-v1.5 分词器计算 [Xiao 等人， 2024] 并包含渲染后的标头和特殊标记。分词器仅用作打包时的长度度量，不参与索引或评分。消息始终保持完整，任何超过 512 个令牌的单条消息都会单独构成一个文本块。因此，检索单元始终是原始对话中的一个片段；这种粒度选择会同时影响检索和阅读 [Pan 等人， 2025]。

渲染文本与溯源信息。 每条消息都会连同其说话人和源日期一起渲染，因此二者都会成为文本块文本的一部分。SAG-Mem 不存储任何派生陈述：第 2 节所述的记忆记录实际上就是这样一个文本块及其溯源信息，

$$c_i = (x_i, \mathcal{R}_i), \qquad c_i \in \mathcal{C}, \tag{3}$$

其中， x_i 是渲染后的源文本，而 $\mathcal{R}_i$ 包含会话标识符、消息标识符以及该文本块所涵盖的完整消息跨度。

源日期。 源日期记录消息发出时的时间，而消息所描述的事件可能发生在另一时间。消息在有自身日期时保留该日期，否则继承会话日期。相对时间表达式在存储时保持逐字形式。它们所依赖的日期会与其一同存储，因此可以等检索之后再对其进行解析（第 3.4节）。

仅追加维护。 新会话中的文本块会追加到 $\mathcal{M}$，并根据规范化文本将其加入词法索引。更正内容与其所修订的陈述共存，各自具有独立的来源和日期。存储区既不会将二者合并，也不会为其分配有效期区间；检索到其中任何一项时，阅读器都会将其作为带日期的源文本接收。对于 SAG-Mem 而言，整体构建过程仅包括分词和建立索引，不涉及模型推理；Lite 变体在此基础上为每个文本块增加一个嵌入（第 3.5节）。

3.2 初始候选项检索

构建过程仅生成逐字文本块及其词法索引，因此连接事实是 检索 的任务。检索遵循 SAG2 [Wu，即 2026], 一种基于词法索引、无需嵌入的多跳检索器。我们采用其 BM25 候选池、基于 JEV 的证据评分、将得分最高的文本块选为种子、扩展词项加权、逐词项交错检索，以及对前五个 BM25 匹配结果的保护机制。我们将检索单元改为完整消息块，并将预算改为 B=50 和 H=1；同时，我们将每一轮（包括最后一轮）的候选池定义为第 U_t 3.3 节的集合，不再像 SAG2 那样用更早轮次的候选项重新填充最终候选池。检索分为两步。首先，本节找出在词法上与问题重叠的文本块，并从中选出种子；随后，第 3.3 节从这些种子出发，触达仅凭问题无法触达的桥接块 [Xionget al.， 2019]。

词法候选池。 我们从词法索引中检索一个初始候选池 C_0，最多包含 B=50 个文本块，使用 BM25 [Robertson and Zaragoza， 2009]。 B 用于限定每次词汇搜索（包括本次初始搜索以及第 3.3 节每一轮扩展）纳入的文本块数量。查询由问题中的词项构成，并在停用词移除和规范化后以析取形式组合。可搜索内容为文本块文本，来源标识则作为元数据。

证据评分。 BM25 排名反映词项重叠，而种子选择需要判断某个文本块是否有助于回答问题。因此，我们使用证据评估器 JEV 对候选项重新排序 C_0，证据评估器 JEV [Almeida， 2026]。JEV 会判断候选项自身的文本是否提供了回答问题所需的事实 q，包括连接其他证据的中间事实。初始评分为

$$d_i^{(0)} = J_\theta\big(q, x_i; E_0, V_0(i)\big), \qquad E_0 = \emptyset, \tag{4}$$

其中 E_0 是显式证据列表， $V_0(i)$ 是与 c_i 同一请求中可见的候选段落集合。此时尚未选出任何证据，因此 E_0 为空，但请求中的其他候选项仍然可见。附录 A介绍了如何将候选池打包到请求中。

种子。 s=10 评分最高的文本块构成扩展种子集合 Z_0。种子有两种作用：它们既是最终列表的候选项，也为下一轮检索提供词语。

3.3 证据引导检索

初始阶段会得到种子，即与问题重叠且证据评分最高的文本块。仍然缺少的是那些与种子共享词语、但与问题本身共享词语较少的事实。

从问题到证据。 考虑这个问题：“Maya 的导师住在哪里？”会话 S_1 中包含 Maya 的一句话：“我的导师是 Leo。”这句话与问题有重叠之处，可以作为种子。会话 S_7 包含 Leo 的一句话：“我

住在京都。”这句话与问题仅共享动词“住”，既没有提到 Maya，也没有提到导师，因此使用 q 发起的搜索很难匹配到它。不过，种子中包含“Leo”一词，而 S_7呈现的文本中也有这个词，因为说话人姓名是文本块的一部分。再次使用种子中的词语进行搜索，就能找到起桥接作用的事实。基于图的系统会在写入时 [Rasmussen 等人，2025年，Xu 等人，2026a年], 存储这条边；迭代检索器则通过生成中间推理或改写后的查询来找到它 [Trivedi 等人， 2023]。这里，系统会在需要时直接从文本中检索出这条事实，因此构建过程能够保持轻量。

扩展词项。 扩展词项直接取自种子块的文本，因此该步骤是确定性的，且完全基于词汇。不同于生成式查询扩展 [Wang 等人， 2023], 它不需要生成模型。它类似于伪相关反馈 [Rocchio， 1971], 但词项仅来自种子块；这些种子块是由评分函数而非 BM25 排名选出的。词项权重结合了 SAG2 中定义的四种信号：词频、其在语料库中的文档频率、其所在句子与问题之间的重叠程度，以及其种子的评分。最多保留 20 个词项；完全由问题词组成的词项会被丢弃，因为第一次检索已经使用了这些词。

收集新候选项。 每个扩展词项都会检索与之匹配且此前检索未返回的文本块，并按照结合问题词与扩展词项的查询进行排名。在第一轮中，这些文本块位于 C_0之外；在多轮运行的第 t 轮中，它们位于 $C_0 \cup \cdots \cup C_{t-1}$之外，因此不会重复检索任何文本块。各词项的列表按排名交错，并去重为至多 B 个新候选项 C_t，这些候选项在第一轮中构成 C_1 。交错检索使每个词项都能发挥作用，从而避免某个高频词占满配额。如果没有可用词项或没有找到新候选项，则保留当前排名，并在此处停止检索。

基于证据重新评分。 新候选项进入候选池，仅仅是因为它与某个种子共享一个术语，这并不意味着它有用。它仍然必须接受评判，而现在这一评判可以利用截至目前找到的证据。第二轮候选池为 $U_1 = P \cup Z_0 \cup C_1$，其中 P包含 C_0 按 BM25 顺序排列的前五个文本块，而 C_0 中位于$P \cup Z_0$ 之外的成员将离开候选池。共享的证据上下文是 $E_1 = Z_0 \cup \text{First}_8(C_1)$，新候选项按检索顺序排列，JEV 会对 U_1中的每个成员重新评分：

$$d_i^{(1)} = J_\theta\big(q, x_i; E_1, V_1(i)\big), \tag{5}$$

其中， $V_1(i)$定义为本轮请求的 $V_0(i)$。 E_1 在本轮的所有决策之间共享，并且可能包含正在评分的候选项。在这个示例中，“我住在京都。”与“我的导师是 Leo。”一同接受评判，从而显现出其作为中间事实的角色。

最终排序。 候选池按照最新的 JEV 分数排序，平分时依次以检索排名和分块标识符作为次级排序依据。返回前 K=20 个文本块，作为 $\mathcal{E}_q$，见公式（1）。算法1 说明了轮次数量一般时的处理流程H。SAG-Mem 运行单轮，其中算法的种子、新候选项、候选池和证据分别为 Z_0、 C_1、U_1和 E_1 上述内容。

3.4 答案生成

答案生成负责实现渲染 g 以及式（2）中的阅读器。检索到的文本块保留原始措辞，而在阅读器查看它们之前，还需进行一步解释：将相对时间表达式转换为日历日期。

检索后的时间澄清。选定的文本块 $\mathcal{E}_q$ 按照检索顺序呈现，并附上完整的消息、说话人、源日期以及会话和消息标识符。确定性渲染器会在每个受支持的相对时间表达式后插入括号标注的日期，同时保留原始措辞。对于日期为 2025/05/12 的消息，“昨天”会变为“昨天（2025/05/11）”。渲染器采用模式匹配和日历运算，其做法遵循 SUTime 等基于规则的时间标注器 [Chang 和 Manning、 2012] 以及 HeidelTime [Strötgen 和 Gertz、 2010]。月级和年级表达式保留其粒度，不受支持的表达式则保持不变。

我们在检索后执行这一步，原因有二。首先，只有阅读器将要查看的消息才需要标注，而且每条消息都带有其发送日期，因此可以用规则来完成语言模型仍容易出错的计算 [Fatemi 等人、 2024]。其次，标注只存在于阅读器的上下文中：索引、分块标识符和检索分数均保持不变，原始措辞也仍可用。

日历规则。 渲染器采用以下约定。诸如“上周”这样的周级表达表示从星期一到星期日的一个日历周。诸如“两周前”这样的计数表达则表示按天计算的偏移量，此处为 14 天，并不表示一个日历周区间。星期表达式遵循另一条规则：英语中的“last”或“next”加星期几，指严格早于或晚于源日期的最近一个对应日期。近似表达式仍保持近似，引号和代码中的文本保持不变。日期采用 2025/05/12 的形式，行内注释采用 yesterday (2025/05/11) 的形式。这些规则适用于英语和中文表达，不过本报告实验中的对话使用英语。

算法 1 证据引导检索，改编自 SAG2 [Wu, 2026]。评分$(c;\ E)$ 由证据列表 E计算得出，是评分函数。第 t 轮的候选池为 $U_t = P \cup Z_0 \cup \cdots \cup Z_{t-1} \cup C_t$。 E_t 针对 t=1指定，这是 SAG-Mem 的默认设置；SAG-Mem-Lite 会忽略证据列表。SAG-Mem 使用 JEV 分数和 H=1；第3.5 节介绍 SAG-Mem-Lite 的设置。

要求： 问题 q，存储 $\mathcal{M}$，评分函数 Score，预算 B、 H、 s、 K
确保： $\mathcal{E}_q$ 与 $|\mathcal{E}_q| \le K$

$C_0 \leftarrow$ BM25$(q, \mathcal{M}, B)$ ▷ 初始词汇池
使用 Score$(c; \emptyset)$ 为每个 $c \in C_0$ 评分
$P \leftarrow$ C_0 中 BM25 顺序的前五个文本块； $U \leftarrow C_0$
按评分排列的 C_0 的前 s 个文本块 $Z_0 \leftarrow$ ▷ 初始种子
针对 $t = 1, \ldots, H$ **do**
  $T \leftarrow$ 扩展术语$(Z_{t-1},\ q)$ ▷ 最多 20 个术语
  $C_t \leftarrow$ 词汇扩展$\big(T,\ q,\ \mathcal{M} \setminus (C_0 \cup \cdots \cup C_{t-1})\ B\big)$ ▷仅限未见过的块
  如果 $T = \emptyset$**或** $C_t = \emptyset$ **那么**
    中断 ▷ 保留当前排名
  结束条件判断
  $U \leftarrow P \cup Z_0 \cup \cdots \cup Z_{t-1} \cup C_t$ ▷ 候选池 U_t
  $E_t \leftarrow$ 第 t 轮的证据列表，包含 $E_1 = Z_0 \cup$First$_s(C_1)$ ▷ 仅供 JEV 只读
  使用 Score$(c; E_t)$ 为每个 $c \in U$ 评分
  如果 $t < H$ **那么**
    $Z_t \leftarrow$ 按评分排列的 $U \setminus (Z_0 \cup \cdots \cup Z_{t-1})$的前$s$ 个文本块 ▷ 个新种子
  结束如果
结束循环
按评分排序，然后按检索排名排序，最后按分块标识符排序
返回 $\mathcal{E}$中的前 K 个 U_q

阅读器。阅读器接收 q 以及呈现的上下文 $g(\mathcal{E}_q)$，并生成$\hat{a}$。按照第2节的要求，它需要简洁地回答，并在证据不足时拒绝作答。它的任务是结合源消息及其日期进行解读，包括跨会话的引用和更新。

3.5 SAG-Mem-Lite

在上述流水线中，查询时唯一需要进行模型推理的检索步骤是 JEV 评分。SAG-Mem-Lite 保留存储、词汇搜索、扩展过程和时间呈现不变，将 JEV 替换为嵌入相似度，并使用自身的搜索预算。检索期间唯一的模型调用是对问题进行一次嵌入。

余弦评分。SAG-Mem-Lite 使用 bge-large-en-v1 对问题和文本块进行编码。5 [Xiao 等人将 2024] 编码为 L2 归一化向量。文本块向量在构建时计算一次，这是 SAG-Mem-Lite 在构建阶段增加的唯一模型成本。向量仅用于评分，而 x_i 仍然是被搜索和阅读的文本。问题在查询时编码一次。评分为

$$d_{\text{Lite}}(c_i) = \bar{e}_q^{\top} \bar{e}_i, \tag{6}$$

这等于余弦相似度，因为两个向量都经过了归一化。

无证据评分。 在算法 1中，SAG-Mem-Lite 将 评分 设为 d_{Lite}。此评分仅取决于固定的问题向量，而忽略证据列表，因此每个文本块只评分一次，重新评分也就简化为对新候选项评分。

更广泛的搜索。 候选项来自与 SAG-Mem 相同的两个来源：初始 BM25 搜索和词汇扩展。与稠密检索 [Karpukhin 等人的做法不同， 2020], 这里的嵌入仅用于对这些候选项排序，不会在整个存储库中运行最近邻搜索。由于余弦评分开销低，SAG-Mem-Lite 将节省下来的预算用于扩大搜索范围：每次词汇搜索最多检索 B=100 个文本块，最多进行 H=2 轮扩展，采用 SAG2 的默认预算。扩展词项的权重依据种子的评分确定；由于余弦值可能为负数，因此将评分映射到 $(d_{\text{Lite}} + 1)/2$，使其保持非负。该映射是单调的，因此种子仍保持原有的余弦排序；映射后的评分也以与 JEV 分数相同的方式参与词项权重计算。

跨轮次候选池。 第一轮之后，会从当前候选池中尚未用作种子的文本块里，按余弦分数最多选出 s 个后续种子。第 t 轮的候选池为 $U_t = P \cup Z_0 \cup \cdots \cup Z_{t-1} \cup C_t$：保留最初的前五个 BM25 匹配结果和所有先前的种子，并纳入本轮的新候选项。因此，每轮都会重新构建候选池

；既不在 P 中、也未被选为种子的前几轮候选项会从候选池中移除，正如C_0 中位于 $P \cup Z_0$ 之外的成员在第一轮后会被移除。因此，第 t 轮的候选池大小最多为 $5 + ts + B$，候选池只会随每轮的 s 个种子而增长，而不会纳入所有先前轮次的候选项。第二次扩展找到新候选项时，第一轮的候选项只有被选为种子才能进入最终列表，而第二轮的每个候选项都可直接参与竞争；若未找到新候选项，则停止检索并保留 U_1。最终列表按余弦相似度排序，最多返回 K=20 个文本块。

替换的代价。 这种替换的代价是，SAG-Mem-Lite 的评分不再取决于所收集的证据：一个只有在种子旁边才显得有用的桥接块，仅根据其与问题的相似度进行排序。因此，这两个变体可能会以不同方式对相同的候选项进行排序，进而选择不同的种子。在仅依据问题相似度选择种子的情况下，SAG-Mem-Lite 的扩展更接近于对重新排序列表进行的伪相关反馈。

总而言之，这两个变体共享存储、词汇搜索、扩展过程和时间呈现，但在候选项的评分方式以及预算方面存在差异 B 和 H。因此，对二者的比较实际上是对两种配置的比较，而不仅仅是对评分函数的比较。第4节给出了它们的配置，并从准确率和成本两个方面对二者进行了评估。

4 实验

4.1 实验设置

数据集和模型。 我们在 LoCoMo 的纯文本设置上进行评估 [Maharana 等人， 2024]（不包括图像说明）以及 LongMemEval-S 基准测试 [Wu 等人， 2025], 其中分别包含 1,986 个和 500 个问题。对于答案生成（阅读器）和 LLM 评审（评判器） [Zheng 等人， 2023], 我们采用两种配置：`Qwen3.6-35B-A3B-FP8` [Qwen 团队， 2026] 以及 `DeepSeek-V4.1-Flash-1017` [Xu 等人，2026年]。两种配置均禁用思考模式和提供商回退，并将温度设为 0。

基线。 我们将 SAG-Mem 和 SAG-Mem-Lite 与三组基线进行比较：（1）**完整上下文**，即将完整的对话历史直接传递给阅读器；（2）**基于记忆的系统**，包括 Zep [Rasmussen 等人， 2025], Mem0 [Chhikara 等人， 2025], TencentDB-Agent-Memory（TDAM） [腾讯云， 2026], 以及 MemOS [Li 等人， 2025]；以及（3）**标准 RAG 基线**，其运行于原始对话历史之上，具体包括**仅 BM25** 和 **稠密检索增强生成**。所有基线均在相同的阅读器和评判器配置下复现。表2汇总了所有软件版本和模型标识符。

评估指标 。 我们从三个维度评估性能 ：

- **准确率：** 使用数据集特定语义匹配和弃答提示，通过 LLM 评审判断二元答案的正确性；排除少数 API 故障，不计入分母。
- **检索召回率：** 针对 $K \in \{5, 10, 20\}$ 的会话级和轮次级 Recall@K，基于真实证据标注进行评估，涵盖 1,973 个带标注的 LoCoMo 问题和 LongMemEval 的全部 500 个问题。
- **效率：** 衡量货币成本和端到端查询延迟，统计基于预构建索引的查询时检索和阅读器生成，同时排除离线索引和评判环节。

默认配置。 记忆构建和分块遵循第 3 节中的规范。在两个基准测试中，SAG-Mem 每轮最多检索 $B = 50$ 个候选项，最多进行 $H = 1$ 轮词汇扩展，而 SAG-Mem-Lite 保留 SAG2 的默认预算，即 $B = 100$，最多进行 $H = 2$ 轮。两个变体均选择$s = 10$ 个扩展种子，保留最多 20 个扩展词项，并将排名前 $K = 20$ 个文本块传递给 Reader。这些默认设置在所有主要评估中均保持不变，不进行按数据集调参；我们将在第 4.6 节评估围绕此默认配置的检索参数敏感性。

统一成本核算。 运营成本包括记忆构建、查询时检索和 Reader 答案生成，但不包括离线 LLM 评审；对于每个不同的对话历史，构建成本仅计一次。所有系统均根据记录的令牌数量，并采用官方 API 定价进行计费，同时计入抽取 LLM、搜索/Reader LLM、证据评估器，以及构建时和查询时嵌入的提示缓存命中折扣。重复使用的检索日志会将其完整逻辑成本计入每个独立的评估配置。

4.2 总体效果

通义千问配置下的总体比较。表3 报告了 Qwen 配置下所有评估系统的端到端答案准确率。总体而言，SAG-Mem 和 SAG-Mem-Lite 均无需依赖基于 LLM 的记忆提取或显式图构建，就取得了出色的表现，由此我们得出三项关键结论：

表2：所有评估系统所使用的软件版本和模型标识符。

系统 / 角色	组件	版本或模型标识符
Zep	记忆后端	Graphiti 0.29.3
Mem0	记忆后端	Mem0 2.2.1
MemOS	记忆后端	MemOS 2.0.32
TDAM	记忆后端	TencentDB-Agent-Memory（GitHub，2026年11月）
仅 BM25	词法检索	BM25 [Robertsonand Zaragoza， 2009]
稠密检索增强生成	嵌入模型	text-embedding-bge-large-en-v1.5
SAG-Mem / Lite	检索引擎	SAG2 0.3.0
SAG-Mem	证据评估器	typesafe/jev-1.14-20261014
消融实验（第 4.4节）	证据评估器	Zev-1 (knowline-35b-a3b-fp8)

(1) SAG-Mem 在两个基准测试中均取得最高的总体准确率，而 SAG-Mem-Lite 始终名列表现最佳的方法之列。两个变体的表现均显著优于所有基于记忆的基线，体现了保留完整源消息并在检索时发现关联的有效性。

(2) 与基于原始对话历史运行的标准 RAG 基线方法相比，SAG-Mem 和 SAG-Mem-Lite 均持续优于仅 BM25 和稠密检索增强生成，这证实了证据引导的词汇扩展和候选项评分能够带来超越单轮检索的收益。

(3) 相较于完整上下文，SAG-Mem 在 LoCoMo 上保持优势，并在 LongMemEval 上取得显著提升；在 LongMemEval 中，冗长的多会话历史会降低完整上下文阅读的效果。尽管 SAG-Mem-Lite 在 LoCoMo 上不及完整上下文，但优于该数据集上的所有其他基于记忆的方法和 RAG 基线方法，并在 LongMemEval 上大幅超越完整上下文。

类别级性能。 为考察不同检索与呈现组件对各类任务的贡献，我们进一步分析了表 3中通义千问配置下的类别级细分结果。总体而言，SAG-Mem 和 SAG-Mem-Lite 在需要跨会话证据关联和时间锚定的类别上优势最为显著：

(1) 在 LoCoMo 上，SAG-Mem 在时序推理方面明显领先于所有基线方法，体现了检索后时间澄清的优势。在多跳问题上，SAG-Mem 显著优于所有基于记忆的方法和标准 RAG 基线，并接近完整上下文；我们在第 4.5 节分析了这一尚存的多跳差距。在对抗性问题上，包括 SAG-Mem 和 SAG-Mem-Lite 在内的检索系统，其拒答表现始终优于完整上下文。
(2) 在 LongMemEval 上，SAG-Mem 在多会话、时间类和单会话用户问题上表现优异，而两个变体在单会话助手查询上的准确率均达到满分。同时，SAG-Mem-Lite 在知识更新方面表现最佳，超过了基于记忆的基线和 SAG-Mem，这表明，对词汇扩展后的候选池进行固定查询嵌入评分，在定位更新后的事实陈述方面尤其有效。

DeepSeek 下的跨模型一致性。 为了考察在更强的大语言模型配置下选择性检索是否仍有益处，我们针对 DeepSeek，对 SAG-Mem 和 SAG-Mem-Lite 与完整上下文参照进行重点跨模型比较，而不重新评估完整的基线套件（其成本主要由写入时的记忆提取决定）。详细结果和跨模型结果见表 3 和表 4，其中阅读器和评判器均切换为所示的大语言模型配置：

(1) SAG-Mem 再次在两个基准测试上取得最高总体评分；在 LoCoMo 上以微弱优势超过完整上下文，同时在 LongMemEval 上保持大幅领先。在 DeepSeek 配置下，SAG-Mem 在 LoCoMo 多跳问题上的表现略胜完整上下文，并在时间类和对抗类别上保持明显优势。在 LongMemEval 上，SAG-Mem 和 SAG-Mem-Lite 在大多数类别中继续大幅领先完整上下文。
(2) 从 Qwen 配置切换到 DeepSeek 后，三种评估方法的总体评分均有所提高，这反映了阅读器和评判器模型的联合变化。完整上下文在 LongMemEval 上取得了最大幅度的提升，在 LoCoMo 上的提升也超过了 SAG-Mem，而 SAG-Mem-Lite 在 LoCoMo 上的增幅最大。因此，在 DeepSeek 配置下，SAG-Mem 相对于完整上下文的领先优势有所缩小，在 LoCoMo 上仅略占优势。即便如此，在 LongMemEval 较长的多会话历史中，SAG-Mem 和 SAG-Mem-Lite 仍然显著优于完整上下文。

表 3：LoCoMo 和 LongMemEval 上的端到端答案准确率（%）。**粗体** 和下划线分别表示每种大语言模型配置中的最高和次高评分（当第一名并列时，不对次高评分添加下划线）。API请求失败的问题不计入分母。Ove.：总体；M-H：多跳；O-D：开放域；S-H：单跳；Temp.：时间；Adv.：对抗；M-S：多会话；K-U：知识更新；S-P/S-A/S-U：单会话偏好/助手/用户。

方法	LoCoMo						LongMemEval						
	Ove.	M-H	O-D	S-H	温度	优势	Ove.	M-S	K-U	温度	S-P	S-A	S-U
大语言模型配置：*Qwen3.6-35B-A3B-FP8*													
完整上下文	80.46	**78.37**	**42.71**	**90.13**	64.85	82.96	68.88	54.13	69.23	65.15	37.93	**100.00**	91.43
Zep	58.25	46.09	32.29	53.50	42.99	91.47	57.77	42.10	61.04	54.14	50.00	67.86	85.71
TDAM	53.67	45.39	29.16	62.66	33.64	61.66	54.70	40.60	71.80	34.60	16.70	91.10	87.10
Mem0	73.91	59.92	34.37	74.31	71.65	92.15	68.80	55.64	74.36	63.16	**63.33**	76.79	94.29
MemOS	65.06	54.61	31.25	64.09	54.21	88.57	69.00	57.89	79.49	70.68	60.00	53.57	91.43
仅 BM25	72.60	48.20	27.10	82.50	54.80	91.90	67.60	46.60	74.40	64.70	40.00	**100.00**	91.43
稠密检索增强生成	76.20	67.00	35.40	82.00	57.90	**93.00**	73.00	65.40	76.90	62.40	53.30	**100.00**	90.00
SAG-Mem-Lite	77.09	64.89	37.50	81.69	68.54	90.81	77.20	61.65	**85.90**	72.18	60.00	**100.00**	95.71
SAG-Mem	**83.08**	75.89	40.62	88.35	**79.44**	89.46	**80.20**	**72.93**	78.21	**76.69**	53.33	**100.00**	**98.57**
大语言模型配置：*DeepSeek-V4.1-Flash-1017*													
完整上下文	83.43	79.08	**51.04**	**92.39**	76.95	80.94	74.40	60.90	**87.18**	67.67	30.00	**100.00**	97.14
SAG-Mem-Lite	81.22	75.18	48.96	87.40	76.32	**83.86**	81.60	69.17	85.90	78.95	**63.33**	**100.00**	98.57
SAG-Mem	**83.99**	**79.79**	44.79	90.61	**82.24**	**83.86**	**82.60**	**72.93**	83.33	**79.70**	**63.33**	**100.00**	**100.00**

表4：通义千问Qwen3.6-35B-A3B-FP8与DeepSeek-V4.1-Flash-1017之间的跨模型比较。阅读器和评判器均使用所示的大语言模型配置。评分为总体准确率（%）； Δ 表示DeepSeek减去通义千问后的百分点差。

方法	LoCoMo			LongMemEval		
	Qwen	DeepSeek	Δ	Qwen	DeepSeek	Δ
完整上下文	80.46	83.43	+2.97	68.88	74.40	+5.52
SAG-Mem-Lite	77.09	81.22	+4.13	77.20	81.60	+4.40
SAG-Mem	83.08	83.99	+0.91	80.20	82.60	+2.40

4.3 系统效率：成本和延迟

货币成本与令牌分解。图 2 总结了准确率与成本之间的权衡，表 5 报告了 Qwen 配置下所有评估系统的货币总成本和各阶段令牌分解，并分别统计写入时的提取与嵌入，以及查询时的检索和阅读器生成。总体而言，随着对话历史规模扩大，省去写入时的大语言模型提取可显著降低成本。由此我们得出三项关键观察：

(1) 基于记忆的基线会产生高昂的写入时大语言模型提取成本；在 LongMemEval 上，由于必须为数百条相互独立的历史记录建立索引，这部分成本占总支出的主要部分。相比之下，SAG-Mem 和 SAG-Mem-Lite 均无需消耗大语言模型提取令牌，因此在 LongMemEval 上的总成本比基于记忆的系统低一到两个数量级。在 LoCoMo 上，许多查询共用少量历史记录，写入时提取的成本更容易分摊；SAG-Mem-Lite 仍比所有基于记忆的基线都便宜，而 SAG-Mem 将预算用于查询时的 JEV 评分，其成本介于较轻量的基于记忆的基线（Mem0 和 MemOS）与较重的基线（TDAM 和 Zep）之间。

（2）在两个基准测试中，与完整上下文和所有基于记忆的基线相比，SAG-Mem-Lite 的总成本最低；其成本与标准 RAG 基线相当，同时准确率更高。由于它依赖轻量级词法扩展和预先计算的分块嵌入，其在 LongMemEval 上的主要开销来自一次性的嵌入构建，而这项成本可在后续查询中有效分摊。相比之下，SAG-Mem 完全无需嵌入成本，并将大部分预算用于查询时的 JEV 证据评分，因此在 LoCoMo 上的成本高于 SAG-Mem-Lite，但在 LongMemEval 上仍远低于完整上下文和所有基于记忆的基线。

（3）在两个基准测试中，选择性检索都能减少读取器令牌的使用量，相较于完整上下文尤为明显。在 LongMemEval 上，这种减少为两个变体都带来了显著的总成本节省。在 LoCoMo 上，在采用相同的未缓存阅读器费率（包括查询时 JEV 评分）的情况下，SAG-Mem 的预估总成本比完整上下文低 54%。

表5：Qwen 配置下的运营成本（美元）和令牌明细。提取大语言模型和搜索/读取大语言模型以百万令牌（M）为单位报告。Emb. 表示嵌入成本（美元）。SAG-Mem 在 LoCoMo 和 LongMemEval 上分别使用 53.31M 和 36.63M 个 JEV 输入和输出令牌。JEV 成本计入总成本。粗体和下划线分别表示各基准测试中最低和次低的总成本。

方法	LoCoMo				LongMemEval			
	总计（美元）	提取（百万）	搜索/读取（百万）	嵌入（美元）	总计（美元）	提取（百万）	搜索/读取（M）	嵌入（美元）
完整上下文	6.20	–	61.66	–	9.80	–	59.47	–
Zep	9.18	48.69	4.91	0.0024	687.49	4257.52	1.08	0.372
TDAM	3.53	26.59	3.14	0.0030	253.41	1353.21	3.33	0.491
Mem0	0.71	7.00	1.23	0.0041	189.86	3659.63	0.41	0.772
MemOS	0.95	1.26	3.64	0.0028	40.07	233.43	1.08	0.481
仅 BM25	**0.67**	–	6.38	–	**0.40**	–	3.70	–
稠密检索增强生成	0.71	–	6.62	0.0032	0.79	–	2.18	0.539
SAG-Mem-Lite	0.70	–	6.45	0.0036	0.95	–	3.33	0.583
SAG-Mem	2.86	–	7.36	–	1.91	–	3.81	–

端到端查询延迟。 表 6 将预构建索引上的平均查询时延迟拆分为检索时长和 Reader 答案生成时间：

(1) SAG-Mem-Lite 的端到端延迟较低，响应速度快于完整上下文、Zep、MemOS 和两个标准 RAG 基线方法，这得益于其 Reader 生成速度快于完整上下文。
(2) SAG-Mem 的端到端延迟高于 SAG-Mem-Lite 和各基线方法，主要原因是在检索过程中进行查询时 JEV 证据评分，以及 Reader 生成时间较长。

表 6：通义千问在 LoCoMo 上的端到端查询延迟（秒），分解为预构建索引上的查询时检索和 Reader 答案生成。粗体和下划线分别表示每列中的最低值和次低值。

方法	检索（秒）	生成（秒）	总计（秒）
完整上下文	**0.0012**	1.0429	1.0441
Zep	0.1908	0.7748	0.9656
TDAM	0.0616	0.7753	0.8369
Mem0	0.0890	**0.5097**	**0.5987**
MemOS	0.2450	0.7947	1.0397
仅 BM25	0.0085	1.0238	1.0323
稠密检索增强生成	0.0534	1.2213	1.2747
SAG-Mem-Lite	0.2528	0.6528	0.9056
SAG-Mem	2.0590	1.2230	3.2820

4.4 组件消融

证据引导的词汇扩展。 表7 对比了完整的 SAG-Mem检索流水线（$H = 1$）与不进行扩展的对照组（$H = 0$），两者均采用通义千问配置，保留初始 BM25候选项、初始 JEV 评分和检索后时间澄清，同时移除词汇扩展轮次和基于证据重新评分。移除扩展轮次会导致 LoCoMo 和 LongMemEval 上的准确率显著下降，证实仅匹配初始查询会遗漏跨会话问题所需的互补事实。在这两个基准测试上均观察到一致的提升，验证了利用选定的来源块引导词汇扩展轮次并基于证据重新评分，是 SAG-Mem 效果的重要因素。

证据评估器解耦。 为了检验 SAG-Mem 是否完全依赖 JEV，我们用 Zev-1 替换 JEV，同时保持流水线的其余部分不变（表 8）。在 Qwen 配置下，Zev-1 在 LoCoMo 上略胜 JEV，在 LongMemEval 上则不及 JEV，但仍以较大优势胜过表 3中的所有基于记忆的方法和标准 RAG 基线方法。

检索后时间澄清。 表 9评估了在 Qwen 配置下，固定检索到的文本块时，两个基准测试的时间子集上的检索后时间澄清效果。我们比较了三种渲染设置：

表 7：Qwen 配置下 SAG-Mem 的证据引导的词汇扩展消融实验。评分为总体准确率（%）； Δ 表示无扩展（$H = 0$）减去主实验（$H = 1$）的百分点差。

数据集	N	主实验（$H = 1$）	无扩展（$H = 0$）	Δ
LoCoMo	1,986	**83.08**	78.15	-4.93
LongMemEval	500	**80.20**	75.20	-5.00

表 8：Qwen 配置下 SAG-Mem 的证据评估器消融实验，对比 JEV 与 Zev-1。评分为总体准确率（%）；粗体表示各基准测试中的最高评分。

证据评估器	LoCoMo	LongMemEval
JEV	83.08	**80.20**
Zev-1	**83.50**	77.60

主要设置 将规范化日期标头与受支持的相对时间表达式的内联日期注释相结合，**原始日期** 保留原始消息标头且不添加内联注释，而**仅日期** 则对标头进行规范化，同时省略内联注释。

（1）在 LoCoMo 上，对于 SAG-Mem 和 SAG-Mem-Lite，**主要方法**相较于**原始日期**和**仅日期**都带来了显著的准确率提升。相比之下，**仅日期**并不优于**原始日期**，这表明，对于 Qwen 阅读器而言，为相对时间表达式添加明确的内联日期注释至关重要。（2）在 LongMemEval 上，该效果较小：对于 SAG-Mem-Lite，**主要方法**优于两种对照方法；而对于 SAG-Mem，其效果与**原始日期**相当，并略优于**仅日期**。在两个基准测试中，确定性的检索后时间澄清在不修改存储索引或增加额外大语言模型调用的情况下，都能带来稳定或正向的收益。

表 9：在 Qwen 配置下、固定检索到的文本块时，时间子集上的检索后时间渲染消融实验。评分为准确率（%）；粗体和下划线分别表示每行中的最高值和第二高值（并列第一时使用粗体而不加下划线）。

数据集 & 方法	Main	原始日期	仅日期
LoCoMo ($N = 321$) — SAG-Mem	**79.44**	<u>63.55</u>	61.37
LoCoMo ($N = 321$) — SAG-Mem-Lite	**68.54**	<u>56.39</u>	56.07
LongMemEval ($N = 133$) — SAG-Mem	**76.69**	**76.69**	75.19
LongMemEval ($N = 133$) — SAG-Mem-Lite	**72.18**	69.17	<u>70.68</u>

4.5 检索诊断与多跳瓶颈分析

检索层召回率。表10 报告了 SAG-Mem 和 SAG-Mem-Lite 在两个基准测试中的会话级和轮次级 Recall@5/10/20，从而将检索覆盖率与下游阅读器生成相分离。在 LoCoMo 和 LongMemEval 上，SAG-Mem 始终实现较高的会话级和轮次级召回率，在前 5 个文本块中捕获绝大多数支持性证据，并在 $K = 20$时接近饱和。相比之下，SAG-Mem-Lite 在较严格的截断位置下落后于 SAG-Mem，尤其是轮次级召回率；但在$K = 20$时，差距大幅缩小。这种差异体现了 JEV 证据评分将金标准证据集中在排名靠前位置的优势，同时也解释了为何向阅读器提供 20 个文本块后，SAG-Mem-Lite 的端到端准确率仍具竞争力。

表10：SAG-Mem 和 SAG-Mem-Lite 在 LoCoMo（$N = 1$，973 个标注问题）和 LongMemEval（$N = 500$）上的检索层会话级和轮次级 Recall@5/10/20（%）。

方法	数据集	N	会话 R@5 / R@10 / R@20	轮次 R@5 / R@10 / R@20
SAG-Mem	LoCoMo	1,973	94.44 / 96.56 / 97.85	90.60 / 93.09 / 94.15
SAG-Mem-Lite	LoCoMo	1,973	79.45 / 87.91 / 94.14	66.33 / 75.99 / 84.62
SAG-Mem	LongMemEval	500	96.19 / 97.66 / 98.40	91.11 / 94.88 / 95.72
SAG-Mem-Lite	LongMemEval	500	89.98 / 96.22 / 98.22	77.56 / 90.25 / 94.02

多跳瓶颈分析。为诊断第4.2节中通义千问条件下相较于完整上下文仍存在的多跳差距，我们使用 SAG-Mem 在 LoCoMo 多跳子集上进行针对性的检索预算扫描和错误归因（表11和表12）。诊断结果表明，这一差距并非源于未能找到金标准证据，而是反映了返回文本块预算 K 与阅读器端多跳推理之间的相互作用。在默认设置下（$K = 20$，$H = 1$），消息和会话覆盖率已经很高，扩大 K 或增加第二轮词汇扩展（$H = 2$）对总体覆盖率的影响微乎其微。然而，将K 从 20 增至 40 可显著提升准确率，弥合并扭转与完整上下文之间的差距；此后准确率在 $K = 60$达到饱和。这一模式与第 K 节中 LongMemEval 所观察到的对4.6的敏感性一致：即便已检索到大多数标注的金标准轮次，额外的文本块仍能提供周边对话上下文和剩余的部分跳数，帮助通义千问阅读器关联散落的信息，但代价是阅读器上下文变长。增加第二轮扩展（$H = 2$）也能小幅提升准确率，同时增加查询时 JEV 评分开销。如表12所示，在 $H = 1$ 和$H = 2$两种设置下，完全检索遗漏仅占失败案例的一小部分，而绝大多数错误仍出现在金标准证据已包含于阅读器上下文的问题上。结合第4.2节中 DeepSeek 的结果（SAG-Mem 即使在 $K = 20$时也优于完整上下文），这些诊断表明，默认多跳准确率同时受限于 $K = 20$ 上下文权衡和阅读器的推理能力。

表 11：在通义千问配置下，针对 LoCoMo 多跳子集（$N = 282$）的 SAG-Mem 多跳检索预算扫描。粗体表示各指标列中的最高值。每种配置均在独立的端到端运行中进行评估；覆盖率出现轻微的非单调波动，反映了不同 JEV API 调用之间的细微评分差异。

配置	B	H	K	正确数 / 总数	准确率（%）	消息覆盖率（%）	会话覆盖率（%）
k20_h1	50	1	20	214 / 282	75.9	94.6	96.8
k40_h1	50	1	40	226 / 282	80.1	94.3	97.0
k60_h1	60	1	60	227 / 282	**80.5**	**95.8**	**97.2**
k20_h2	50	2	20	220 / 282	78.0	95.1	96.9

表 12：在通义千问配置下，SAG-Mem 在 LoCoMo 多跳子集（$N = 282$）上的错误归因细分。**完全遗漏** 统计未返回任何带注释的金标准证据轮次的错误答案；**证据存在但答案错误** 统计至少检索到一个金标准证据轮次的错误答案（涵盖金标准轮次的完整覆盖和部分覆盖）。

配置	正确	完全遗漏	证据存在但答案错误
$K = 20, H = 1$	214	4	64
$K = 20, H = 2$	220	1	61

4.6 超参数敏感性分析

为考察检索预算如何影响性能，同时避免对所有参数组合进行完整基准测试的 JEV 评估成本，我们在 LongMemEval 的按类别分层的 125 个问题子集上，针对 Qwen 配置下的 SAG-Mem 进行敏感性扫描，以补充第 4.5 节中的 LoCoMo 多跳预算扫描。围绕固定的默认配置（默认：$B = 50$、$H = 1$、K =20），我们调整三个核心检索超参数：每轮候选池大小 B、词汇扩展轮次的最大数量 H，以及传递给阅读器的文本块数量 K （表 13）。总体而言，在这三个维度上调整检索预算都会对子集准确率产生明显影响，而固定的默认设置在准确率与查询时间开销之间提供了均衡的权衡：

（1）在这三个参数中，准确率对返回分块预算 K 最为敏感。将 K 降低至默认值以下会降低准确率，而将 K 加倍则能在该子集上带来最大的单参数增益，但代价是阅读器输入上下文也加倍。

(2) 调整候选池大小 B 和扩展深度 H 会产生不对称的影响。缩小 B 不会改变子集准确率，而禁用词汇扩展（$H = 0$）则会降低准确率，这与第 4.4 节的全基准测试消融实验结果一致。将 B 或 H 加倍，都会使子集表现获得同等幅度的中等提升，不过二者都会增加查询时 JEV 评分开销。

表 13：在 Qwen 配置下，针对按类别分层的 LongMemEval 子集（$N = 125$）对 SAG-Mem 进行超参数敏感性扫描。default 表示固定的默认配置（$B = 50$、$H = 1$、$K = 20$）。粗体表示最高准确率。

配置	B	H	K	正确数 / 总数	准确率（%）
默认	50	1	20	94 / 125	75.2
k10	50	1	10	90 / 125	72.0
k40	50	1	40	100 / 125	**80.0**
b20	20	1	20	94 / 125	75.2
b100	100	1	20	98 / 125	78.4
h0	50	0	20	91 / 125	72.8
h2	50	2	20	98 / 125	78.4

5 相关工作

记忆表示与组织。外部对话记忆在所存内容的粒度及其组织结构方面各不相同 [Hu 等人， 2025]。摘要和反思会生成对过往交互或经历的精简描述，包括事件摘要、用户画像和高层次洞见 [Zhong等，2024年，Park 等人，2023年，Tan 等人， 2025]。其他方法则将显著信息表示为持久事实或独立的话语单元，以保留细节供细粒度检索 [Chhikara等，2025，Zhou 和 Han， 2025]。分层设计将记忆组织为不同层级，并管理层级间的信息传递或整合 [Packer 等人，2023年，Fang 等人， 2026]。时序知识图谱和链接笔记结构会维护记忆及其上下文之间的关联 [Rasmussen 等，2025年，Xu 等，2026a年]。我们的框架以文本块为单位保留完整的对话消息及其元数据，避免基于大语言模型的记忆改写和持久语义关系构建。

记忆检索策略。 检索策略旨在找到原始问题匹配结果之外的支持性证据。相关性反馈和自适应重排序会利用初始结果或评分较高的文档来引导后续搜索 [Lavrenko 和 Croft，2017年，MacAvaney 等人， 2022]。生成的推理步骤、答案草稿和预测续写提供了额外的检索线索 [Trivedi 等人，2023年，Qian 等人，2025年，Jiang 等人，2023]。基于图的检索利用实体与段落的连接关系，从分布在多个文档中的内容中找回证据 [Gutiérrez 等人，2024, 2025]。SAG [Wu 等人， 2026]在查询时通过事件—实体索引中的共享实体连接源段落。SAG2 [Wu， 2026] 将 BM25 检索与决策模型证据评分相结合，并根据选定段落进行词汇扩展，无需离线嵌入计算或基于大语言模型的关系抽取。我们的工作将 SAG2 的证据引导的词汇扩展应用于对话记忆，并评估基于 JEV 和基于嵌入的评分配置。

6 结论

我们介绍了 SAG-Mem 和 SAG-Mem-Lite，这是对话记忆框架的两种配置，能够保留完整的对话消息，并通过基于种子的词汇扩展发现互补证据。SAG-Mem 将无模型记忆构建与基于 JEV 的证据评估相结合，而 SAG-Mem-Lite 则使用预计算的分块嵌入进行候选项评分。两种配置都会为有依据的相对时间表达式添加基于日期的解释，同时保留原文和时间粒度。
在 LoCoMo 和 LongMemEval 上，SAG-Mem 搭配 Qwen3.6-35B 时的总体得分分别为 83.08 和 80.20，搭配 DeepSeek-V4.1-Flash 时分别为 83.99 和 82.60，在全部四种设置中均优于 Full Context 基线。S
AG-Mem-Lite 搭配通义千问时的相应得分为 77.09 和 77.20，搭配 DeepSeek 时为 81.22 和 81.60，同时保持较低的查询延迟，且运营成本低于 Full Context 和所有基于记忆的基线。这些结果表明，在所评估的两种大语言模型配置下，对话保留和证据引导的检索扩展是构建长期对话记忆的有效设计。

参考文献

Charles Packer、Sarah Wooders、Kevin Lin、Vivian Fang、Shishir G Patil、Ion Stoica 和 Joseph E Gonzalez。MemGPT：将大语言模型作为操作系统。*arXiv*预印本 *arXiv:2310.08560*，2023年。

Patrick Lewis、Ethan Perez、Aleksandra Piktus、Fabio Petroni、Vladimir Karpukhin、Naman Goyal、Heinrich Küttler、Mike Lewis、Wen-tau Yih、Tim Rocktäschel 等。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，33：9459–9474，2020年。Adyasha Maharana、Dong-Ho Lee、Sergey Tulyakov、Mohit Bansal、Francesco Barbieri 和 Yuwei Fang。评估大语言模型智能体的超长期对话记忆。载于第62届计算语言学协会年会论文集（第1卷：长篇论文），第13851–13870页，2024年。Di Wu、Hongwei Wang、Wenhao Yu、Yuwei Zhang、Kai-Wei Chang 和 Dong Yu。LongMemEval：对聊天助手的长期交互记忆进行基准测试。载于国际学习表征会议，2025年。Peng Qi、Xiaowen Lin、Leo Mehr、Zijian Wang 和 Christopher D Manning。通过迭代式查询生成回答复杂开放领域问题。载于2019年自然语言处理实证方法会议和第9届国际自然语言处理联合会议（EMNLP-IJCNLP）论文集，第2590–2602页，2019年。Harsh Trivedi、Niranjan Balasubramanian、Tushar Khot 和 Ashish Sabharwal。将检索与思维链推理交错，用于回答知识密集型多步问题。载于第61届计算语言学协会年会论文集（第1卷：长篇论文），第10014–10037页，2023年。Wanjun Zhong、Lianghong Guo、Qiqi Gao、He Ye 和 Yanlin Wang。MemoryBank：通过长期记忆增强大语言模型。载于AAAI人工智能会议论文集，第38卷，第19724–19731页，2024年。Zhen Tan、Jun Yan、I-Hung Hsu、Rujun Han、Zifeng Wang、Long Le、Yiwen Song、Yanfei Chen、Hamid Palangi、George Lee 等。前瞻与回顾：面向长期个性化对话智能体的反思式记忆管理。载于第63届计算语言学协会年会论文集（第1卷：长篇论文），第8416–8439页，2025年。Prateek Chhikara、Dev Khant、Saket Aryan、Taranjeet Singh 和 Deshraj Yadav。Mem0：构建具备可扩展长期记忆、可用于生产环境的人工智能智能体。arXiv预印本 arXiv:2504.19413，2025年。Wujiang Xu、Zujie Liang、Kai Mei、Hang Gao、Juntao Tan 和 Yongfeng Zhang。A-MEM：面向大语言模型智能体的智能体记忆。神经信息处理系统进展，38：17577–17604，2026a年。Preston Rasmussen、Pavlo Paliychuk、Travis Beauvais、Jack Ryan 和 Daniel Chalef。Zep：用于智能体记忆的时序知识图谱架构。arXiv预印本 arXiv:2501.13956，2025年。Zhengye Bao、Shengtao Zhang、Haokai Zhang、Shiqi Lou、Muning Wen 和 Minnan Luo。复杂智能体记忆总是必需的吗？朴素检索增强生成比我们想象的更强。arXiv预印本 arXiv:2610.30755，2026年。Zheng Chu、Jingchang Chen、Qianglong Chen、Weijiang Yu、Haotian Wang、Ming Liu 和 Bing Qin。TimeBench：对大语言模型时序推理能力的综合评测。载于第62届计算语言学协会年会论文集（第1卷：长篇论文），第1204–1228页，2024年。Bahare Fatemi、Mehran Kazemi、Anton Tsitsulin、Karishma Malkan、Jinyeong Yim、John Palowitch、Sungyong Seo、Jonathan Halcrow 和 Bryan Perozzi。Test of Time：用于评估大语言模型时序推理能力的基准测试。arXiv预印本 arXiv:2406.09170，2024年。Jannik Strötgen 和 Michael Gertz。HeidelTime：高质量的时间表达式基于规则抽取与规范化。载于第5届语义评测国际研讨会论文集，第321–324页，2010年。Angel X Chang 和 Christopher Manning。SUTime：用于识别和规范化时间表达式的工具库。载于第八届国际语言资源与评测会议（LREC'12）论文集，第3735–3740页，2012年。Yuchao Wu。SAG2：使用BM25和JEV进行无嵌入多跳检索。arXiv预印本 arXiv:2610.22588，2026年。Stephen Robertson 和 Hugo Zaragoza。概率相关性框架：BM25及其后续发展。《信息检索前沿与趋势》，3(4)：333–389，2009年。doi:10.1561/1500000019。Diogo Almeida。介绍System One Models和JEV。TypeSafe AI Blog，2026年9月。URL https://typesafe.ai/blog/introducing-system-one-models-and-jev。访问日期：2026年11月20日。Shitao Xiao、Zheng Liu、Peitian Zhang、Niklas Muennighoff、Defu Lian 和 Jian-Yun Nie。C-Pack：用于通用中文嵌入的打包资源。载于第47届ACM SIGIR信息检索研究与发展国际会议论文集，第641–649页，2024年。Zhuoshi Pan、Qianhui Wu、Huiqiang Jiang、Xufang Luo、Hao Cheng、Dongsheng Li、Yuqing Yang、Chin-Yew Lin、H Vicky Zhao、Lili Qiu 等。SeCom：个性化对话智能体的记忆构建与检索。载于国际学习表征会议，第2025卷，第91851–91885页，2025年。

Wenhan Xiong、Mo Yu、Xiaoxiao Guo、Hong Wang、Shiyu Chang、Murray Campbell 和 William Yang Wang。用于开放域多跳问答的简单而有效的桥接推理。载于《第二届面向问答的机器阅读研讨会论文集》，第48–52页，2019年。Liang Wang、Nan Yang 和 Furu Wei。Query2doc：使用大语言模型进行查询扩展。载于《2023年自然语言处理实证方法会议论文集》，第9414–9423页，2023年。Joseph John Rocchio。信息检索中的相关性反馈。载于 Gerard Salton 编，《SMART 检索系统：自动文档处理实验》，第313–323页。Prentice-Hall，1971年。Vladimir Karpukhin、Barlas Oguz、Sewon Min、Patrick Lewis、Ledell Wu、Sergey Edunov、Danqi Chen 和 Wen-tau Yih。面向开放域问答的稠密段落检索。载于《2020年自然语言处理实证方法会议论文集》（EMNLP），第6769–6781页，2020年。Lianmin Zheng、Wei-Lin Chiang、Ying Sheng、Siyuan Zhuang、Zhanghao Wu、Yonghao Zhuang、Zi Lin、Zhuohan Li、Dacheng Li、Eric Xing 等。使用 MT-Bench 和 Chatbot Arena 评判 LLM 评审。神经信息处理系统进展，36：46595–46623，2023年。Qwen 团队。Qwen3.6-35B-A3B：智能体编程实力，如今向所有人开放，2026年4月。网址：https://qwen.ai/ blog?id=qwen3.6-35b-a3b。Anyi Xu、B Li、Bangcai Lin、Bing Xue、BingCheng Xian、Bingzheng Xu、Bochao Wu、Bowei Zhang、Boyi Deng、CC Yu 等。DeepSeek-V4.1-Flash：突破 KV 缓存压缩的极限。arXiv 预印本 arXiv:2609.19969，2026年。腾讯云。TencentDB 智能体记忆。https://github.com/TencentCloud/TencentDB-Agent-Memory，2026年。GitHub 代码库。访问日期：2026年11月2日。Zhiyu Li、Shichao Song、Hanyu Wang、Simin Niu、Ding Chen、Jiawei Yang、Chenyang Xi、Huayi Lai、Jihao Zhao、Yezhaohui Wang 等。MemOS：大语言模型中记忆增强生成（MAG）的操作系统。arXiv 预印本 arXiv:2505.22101，2025年。Yuyang Hu、Shichun Liu、Yan wei Yue、Guibin Zhang、Boyang Liu、Fangyi Zhu、Jiahang Lin、Honglin Guo、Shihan Dou、Zhiheng Xi 等。人工智能智能体时代的记忆。arXiv 预印本 arXiv:2512.13564，2025年。Joon Sung Park、Joseph O'Brien、Carrie Jun Cai、Meredith Ringel Morris、Percy Liang 和 Michael S Bernstein。生成式智能体：人类行为的交互式模拟体。载于《第36届 ACM 用户界面软件与技术年会论文集》，第1–22页，2023年。Sizhe Zhou 和 Jiawei Han。面向大语言模型智能体长期对话记忆的简单而强大的基线方法。arXiv 预印本 arXiv:2511.17208，2025年。Jizhan Fang、Xinle Deng、Haoming Xu、Ziyan Jiang、Yuqi Tang、Ziwen Xu、Shumin Deng、Yunzhi Yao、Mengru Wang、Shuofei Qiao 等。Lightmem：轻量高效的记忆增强生成。载于《国际学习表征会议》，第2026卷，第98706–98729页，2026年。Victor Lavrenko 和 W Bruce Croft。基于相关性的语言模型。载于《ACM SIGIR 论坛》，第51卷，第260–267页。ACM，美国纽约，2017年。Sean MacAvaney、Nicola Tonellotto 和 Craig Macdonald。基于语料库图的自适应重排序。载于《第31届 ACM 国际信息与知识管理会议论文集》，第1491–1500页，2022年。Hongjin Qian、Zheng Liu、Peitian Zhang、Kelong Mao、Defu Lian、Zhicheng Dou 和 Tiejun Huang。Memorag：通过全局记忆增强检索增强提升长上下文处理能力。载于《2025年 ACM Web 会议论文集》，第2366–2377页，2025年。Zhengbao Jiang、Frank F Xu、Luyu Gao、Zhiqing Sun、Qian Liu、Jane Dwivedi-Yu、Yiming Yang、Jamie Callan 和 Graham Neubig。主动检索增强生成。载于《2023年自然语言处理实证方法会议论文集》，第7969–7992页，2023年。Bernal J Gutiérrez、Yiheng Shu、Yu Gu、Michihiro Yasunaga 和 Yu Su。HippoRAG：面向大语言模型、受神经生物学启发的长期记忆。神经信息处理系统进展，37：59532–59569，2024年。Bernal Jiménez Gutiérrez、Yiheng Shu、Weijian Qi、Sizhe Zhou 和 Yu Su。从检索增强生成到记忆：大语言模型的非参数持续学习。arXiv 预印本 arXiv:2502.14802，2025年。Yuchao Wu、Junqin Li、XingCheng Liang、Yongjie Chen、Yinghao Liang、Linyuan Mo 和 Guanxian Li。SAG：结合查询时动态超边的 SQL 检索增强生成。arXiv 预印本 arXiv:2606.15971，2026年。

一种 JEV 请求打包方法

候选池通过一个或多个 JEV 请求进行评分，而集合 $V_t(i)$ （公式（4）和（5））取决于候选池如何被打包到这些请求中。

候选池顺序。 客户端按候选池现有顺序遍历。对于初始候选池 C_0，顺序为 BM25 顺序。对于重新评分候选池 U_1，顺序依次为 P 的五个文本块、种子，然后是按检索顺序排列的新候选项。
令牌估算。 客户端使用自己的 `o200k_`基础 计数器累积令牌估算，该计数器不同于第 3.1 节中用于分块的 BGE 计数。估算值超过 64,000 后，客户端会开启一个新请求。
共享状态。 同一请求中的候选项共用一个状态，其中包含问题和显式证据列表；在有状态模式下，还包含该请求中所有候选项的段落以及通用评分指令。此状态不会在请求之间共享。在有状态模式下，如果状态超过 32,000 个令牌，或整个请求超过 64,000 个令牌，则当前候选项列表会减半并重新打包。这两个阈值是本地客户端的设置，而不是模型的限制。

打包确定性……请求会并发发送，分数则根据候选项标识符写回。因此，客户端侧的打包以及随之而来的 $V_t(i)$，由候选池及其顺序固定，不受返回的分数影响。当候选项或证据本身需要拆分时，候选项保留其部分分数中的最高分，而存储的文本块则保持完整。这确保了客户端侧请求构建的确定性；调用远程 JEV API 的不同端到端运行，在种子选择边界附近仍可能出现轻微的浮点分数波动，从而导致不同独立运行之间的覆盖率略有变化（表 11）。