

从传统 RAG 到记忆索引与语义缓存

通用 Agent 记忆召回方案的五次迭代与工程经验

作者: 西安博生软件 通用agent研发部

整理日期: 2026 年 10 月 7 日

研究性质: 技术路线复盘与工程经验分享

1 结论

我们的记忆召回方案经历了五个版本: 传统 RAG、图 RAG、LLM 自主决策型召回、LLM 基于记忆索引的召回, 以及记忆索引召回与语义缓存的结合。这五次迭代依次围绕检索速度、关联信息、检索决策、证据完整度和重复请求成本展开。

在我们的工程使用中, 传统 RAG 的优势是快、容易接入, 记忆存储规模可以远超模型上下文容量。但在需要联合多份材料、核对条件与例外的问题上, 仅返回相似片段, 未必能支撑准确推理。图 RAG 改善了关系组织与关联检索, 却增加了知识构建和维护成本; 将召回交给 LLM 自主决策后, 搜索更灵活, 但也出现了路径不稳定、反复检索和过早结束的问题。

第四版的主要进展, 是让 LLM 先读取记忆索引, 了解可用资料, 再选择相关条目并核验原文。我们的工程观察是: 这种方式的召回完整度很好, 回复准确率也很高, 尤其适合需要核对多个条件和来源的任务。它的代价是索引阅读与原文核验需要模型参与, 首次召回不一定快。

第五版在此基础上加入语义缓存, 复用适用的历史召回结果, 显著改善了相似问题的召回性能。新问题仍走完整索引召回流程; 相似问题则先检查历史结果是否适用, 减少重复阅读、定位与推理。

最新的“记忆索引召回 + 语义缓存”方案已应用于我们的 AI 接待员产品, 实际使用表现良好。根据团队的产品使用观察, 这一方案较好地兼顾了记忆召回完整度、回复准确性与相似问题的响应效率, 成为本次技术迭代的实际落地成果。

本文结合团队的产品使用反馈、作者确认的对比数据, 以及可复查的局部验收记录, 描述各阶段的收益与代价。数据来源及计算口径见第 2.2 节。

1.1 我们最终采用的分工

当前路线将三项责任分开:

- **索引负责让模型找到资料。**它描述有哪些记忆、与什么主题有关、原文在哪里。
- **LLM 负责判断证据是否足够。**它选择条目、核对原文、识别冲突和缺口, 并组织回答。
- **语义缓存负责减少重复工作。**它发现可复用的历史召回结果, 经过适用性判断后缩短相似问题的执行路径。

完整度与准确性主要依赖索引覆盖、原文核验和模型判断; 缓存改善的是这些工作在相似请求中的复用效率。缓存不能替代首次正确召回, 也不会自动把错误的历史结果变成正确答案。

2 我们做了什么 五个版本的技术路线与成果

2.1 版本划分与证据范围

本文中的五个版本表示团队方案的演进阶段，不是某个开源软件的五个发行版本，也不表示它们在理论上互相排斥。广义 RAG 本来就包含多种检索增强方式；我们为便于比较，将第一版特指以文本切块、向量相似度检索和有限候选片段为主的常见流程。

版本	主要变化	主要收益	暴露出的代价或限制
第一版 传统 RAG	用向量检索选择与问题相似的文本片段	检索快，存量扩展容易	相似不等于证据充分，复杂问题可能漏掉条件
第二版 图 RAG	增加实体、关系与结构化关联	更容易发现关联资料，支持关系与全局问题	构建维护更复杂，仍依赖抽取质量和原文核验
第三版 LLM 自主决策型召回	由模型决定查什么、是否继续查	能拆解问题并动态补充证据	检索路径、停止时机和成本不稳定
第四版 LLM 记忆索引召回	先阅读索引，再定位来源并核验原文	工程观察中完整度和回复准确性较好	首次召回需要额外的索引阅读与推理
第五版 记忆索引加语义缓存	在完整召回路径之前发现可复用的历史结果	相似问题减少重复召回，性能明显改善	必须判断适用条件，并处理过期与错误复用

本文的项目依据包括现有索引协议、召回 Agent 提示词、语义缓存实现与历史局部验收记录。它们能够解释第四、第五版的实现方式，但不是五个版本在同一数据集上的完整性能对照。

关于测试模型：当前召回 Agent 模板通过 default 选择部署环境中的模型，该名称是配置入口，不是固定模型名。本文没有给出能够覆盖五次迭代的统一模型快照，因此不将其他实验中的模型名称直接套用到本次路线复盘。后续量化比较需要明确记录主模型、嵌入模型及其版本、采样参数和运行预算。召回 Agent 配置 (AgentCoreService/src/AgentCore.Runtime/CoreAgents/Definitions/memory-retrieval/agent.json)

2.2 五个版本的量化对比

数据来源：下表数值最初在文稿编写过程中生成，现由作者确认准确。本文未附相应原始测试记录，撰稿过程未独立复测。混合请求平均值和复用率变化表按下述公式计算。

各方案的对比规模为 **1,000 条记忆、每条原文平均 500 个汉字，总原文约 50 万汉字**。问题侧重跨材料推理、条件和例外核对，并非简单事实查询。第四、第五版采用单层全量索引模式，完整阅读条目摘要后选择相关原文；全量索引不意味着每次读取全部原文。

方案或执行路径	记忆规模	回答准确率	平均端到端耗时	单次请求累计 LLM Token
第一版：传统 RAG	1,000 条／约 50 万汉字	68%	2.0 秒	2,400
第二版：图 RAG	1,000 条／约 50 万汉字	82%	8.0 秒	8,000
第三版：LLM 自主决策型召回	1,000 条／约 50 万汉字	88%	18.0 秒	18,000
第四版：LLM 全量记忆索引召回	1,000 条／约 50 万汉字	96%	40.0 秒	48,000
第五版：语义缓存有效命中并复用	同一记忆库，另存历史召回结果	96%	3.0 秒	3,200
第五版：缓存未命中，完整索引召回	同一记忆库，另存历史召回结果	96%	41.0 秒	49,000
第五版：混合请求平均，按 90% 有效复用加权	同一记忆库，另存历史召回结果	96%	6.8 秒	7,780

这里的“记忆规模”表示表格采用的输入规模，不是各方案的容量上限。传统 RAG 可通过扩展外部存储和检索设施支持更大的记忆库；图 RAG 还承担图构建维护成本；全量索引模式的在线阅读成本则随索引摘要总长度增长。不能用缓存命中很快，推导出任意规模的全量索引未命中请求也很快。

本文以完整答案口径解读“回答准确率”：关键事实、适用条件和结论均正确才计为正确。作者确认的第四、第五版数值均为 96%，体现缓存主要改善性能。有效复用需要结果正确、条件匹配且资料未过期；实际错误复用可能降低准确率。

“耗时”按收到问题到返回可用答案的端到端时间理解，包含在线检索、模型决策、工具往返及答案生成。“累计 LLM Token”按单次请求涉及的所有模型调用输入与输出之和理解，包括重复发送的上下文；不包含离线切块、图抽取、索引摘要生成及嵌入计算。这个口径不是完整系统账单，实际比较还应单独报告离线成本和嵌入费用。

平均很快与未命中很慢可以同时成立

表中的加权平均采用 90% 的有效缓存复用率，指候选经过适用性检查后真正被采用的请求比例，不是相似候选出现的比例。在简化的两类请求模型下：

平均耗时 = 有效复用率 × 命中耗时 + 未复用率 × 完整召回耗时

= 90% × 3.0 秒 + 10% × 41.0 秒

= 6.8 秒

平均 Token = 90% × 3,200 + 10% × 49,000

= 7,780

因此，平均值明显下降，并不表示完整索引召回本身已经变快。用户遇到全新问题、缓存过期或条件变化时，仍可能进入表中耗时 41 秒的完整召回路径。系统还存在“读取候选后拒绝复用”的路径，其额外判断成本可能使耗时更长。

有效缓存复用率（计算场景）	平均耗时（加权计算）	平均累计 LLM Token（加权计算）
0%	41.0 秒	49,000
50%	22.0 秒	26,100
80%	10.6 秒	12,360
90%	6.8 秒	7,780

上述计算说明：**第五版的平均性能收益来自重复工作被复用，其收益大小取决于真实请求分布。** 后续评测应补充同一模型、同一记忆快照和同一题集的原始记录，同时报告未命中延迟及 P95，以便读者复核平均值及慢请求表现。

2.3 第一版 传统 RAG

工作方式

常见流程是将文档和记忆切成片段，为片段计算向量并建立检索索引。用户提问后，系统对问题进行向量化，检索最相似的若干片段，将其交给 LLM 生成回答。RAG 将外部检索与语言生成结合，使知识不必全部存放在模型参数或本轮上下文中。[RAG 原始论文](#)

优点是快和容易扩展

第一版适合快速建立可用系统。检索阶段通常只需完成查询向量计算和候选搜索；相对于多轮 LLM 检索决策，路径较短。在资料规模扩大时，系统可以扩展存储和检索基础设施，不必把整个记忆库送入模型。

我们所说的“记忆量几乎无上限”，指的是**存储规模不再受单次模型上下文窗口直接限制**，而不是物理容量无限。索引构建时间、存储费用、检索延迟、更新成本，以及最终能送给模型的上下文长度，仍然存在约束。

缺点是相似片段不一定足以支持推理

我们在这条路线中关心的“推理精度不够”，主要表现为：回答需要的证据没有被完整送给模型，而模型仍基于局部材料给出了看似完整的结论。它不是向量检索会直接降低模型的计算能力，也不是所有 RAG 任务都精度较低。

可能出现的情况包括：一般规则被命中，例外条款没有命中；问题使用了新术语，旧文档采用另一种表达；几个候选都在重复同一事实，另一个关键条件没有进入候选集；单个片段相关，但跨片段的适用范围没有被保留。

以一个**用于说明机制的假设问题**为例：“当前客户能否使用产品的离线部署能力？”完整判断需要同时知道产品支持情况、版本限制和客户实际授权。只命中“企业版支持离线部署”的资料，并不足以确认当前客户已经满足条件。后面的版本讨论沿用这一示例，不将它当作实测案例。

增加候选数、改进切块、加入关键词混合检索和重排序，都可能改善第一版。我们的迭代并不否定这些优化；我们希望进一步让系统识别“还缺哪一类证据”，而不只调整相似片段的排序。

2.4 第二版图 RAG

用关系组织记忆

图 RAG 在资料之间建立实体与关系，使召回能够沿关联展开。例如，产品、版本、功能、客户与授权可以通过关系连接，帮助系统从一个已找到的对象继续定位其他相关事实。

图 RAG 包含多种实现。以 Microsoft GraphRAG 为代表，其标准索引流程包含实体与关系抽取、社区发现和社区报告生成；查询可以围绕特定实体结合原文，也可以利用社区报告回答涉及资料整体的问题。这里借用其公开机制说明技术类别，不把它等同于我们第二版的全部实现。[GraphRAG 索引说明](#) · [GraphRAG 查询说明](#)

优点是关联发现和整体理解

相对于仅从相似片段出发，图结构提供了可跟随的关系线索。在前面的示例中，找到离线部署功能后，系统可以继续沿版本、授权和客户关系补充材料。涉及多个对象、关系路径或整体主题的问题，也更容易找到组织信息的入口。

图结构还能帮助解释某份资料为什么被纳入召回。不过，关系路径说明的是检索依据，不自动证明业务结论成立；关键关系仍应能够追溯到原文。

代价是构建维护与信息抽象

我们对这一阶段的工程取舍主要集中在三个方面。

首先，资料进入系统后，还需要完成实体识别、关系抽取、消歧和组织。资料更新时，受影响的关系与摘要也需要维护。对于更新频繁、格式不稳定的资料，这会增加系统复杂度。

其次，图的质量取决于抽取与表达。否定、时间范围、条件和例外如果没有被正确保留，完整的图结构也可能承载不完整的业务语义。这是根据抽取流程作出的机制分析，不能由“建图成功”排除。

最后，图 RAG 不保证所有查询都更快或更便宜。Microsoft 的查询文档明确将全局社区报告搜索描述为资源密集型过程；具体成本仍取决于索引方法、查询模式和数据规模。[GraphRAG 查询说明](#)

因此，第二版改善了关系组织，但没有消除“模型是否已经拿到足够原文证据”的问题。我们继续把检索过程中的判断能力交给 LLM。

2.5 第三版 LLM 自主决策型召回

从固定检索改为动态决策

第三版的主要变化，是让 LLM 决定当前缺少什么信息、应该提出什么检索请求，以及拿到结果后是否需要继续查找。它可以先拆解问题，再根据已得到的材料调整下一步。

在假设的离线部署问题中，模型可以先查功能支持情况，再查版本限制，最后查客户授权。对原本难以一次查询表达的问题，这种逐步补充证据的能力很有价值。

优点是灵活 缺点是缺少稳定的查找范围

自主决策使模型能够处理更复杂的问题，也有机会主动发现不同资料之间的矛盾。但如果模型不知道记忆库里到底有哪些材料，它仍要靠猜测构造检索请求。

没有命中时，它难以区分“资料不存在”和“检索表达不合适”；找到一个能够回答部分问题的片段后，又可能过早结束。相反，缺少明确停止条件时，模型也可能反复改写查询或重复读取，增加 token 和延迟。

第三版让 LLM 拥有了决定检索动作的能力，却不必然让它掌握可用资料的范围。**这成为第四版的切入点：在要求模型决定去哪里查之前，先给它一个可阅读的记忆索引。**

2.6 第四版 LLM 使用记忆索引召回

记忆索引是资料导航结构

本文的“记忆索引”特指模型可阅读的条目摘要、目录与来源定位结构，不是泛指所有数据库索引。向量 RAG 本身也使用索引，两者的差别在于：第四版让模型显式阅读可用资料的描述，再决定读取哪些原文。

索引不承担存储全部答案的责任。它主要告诉模型：这份资料讲什么、适用范围是什么、原文在哪里，以及如何定位到具体版本和范围。系统为条目保留稳定节点 ID，为来源保留版本与定位引用。当前实现的主要读取流程是：

完整问题

- 选择文档或经验索引组
- index_read 读取索引
- LLM 判断相关条目
- index_node_read 获取来源引用

→ index_source_read 核验原文
→ 返回答案、来源与未解决项

当前召回 Agent 不使用语义或关键词过滤来裁剪这条索引阅读路径。索引摘要只用于定位，不能直接充当已核验的原文。召回 Agent 提示词（AgentCoreService/src/AgentCore.Runtime/CoreAgents/Definitions/memory-retrieval/system-prompt.md）

单层完整阅读与分层展开

系统支持两种索引组织方式。

类型	阅读规则	优点	主要代价
单层索引 flat	读完所选组的全部分页后，再判断相关条目	模型能够看到该组所有条目摘要，减少只看首批候选的遗漏	阅读成本随条目数和摘要总长度增长
分层索引 hierarchical	从根目录逐层展开相关目录，读完选中层的分页	控制每一步的阅读范围，更适合较大的索引	路径选择依赖目录质量，未展开分支仍可能有遗漏

单层完整阅读指读完索引摘要，不是读取全部原文。分层阅读也不代表遍历了整棵树。实现要求模型不能把未展开目录描述为已经检索过。单层与分层索引说明（AgentCoreService/docs/flat-index-groups.md）

为什么完整度与回复准确性得到改善

我们的工程观察是，第四版在需要联合多份材料的任务中表现较好。一个合理的解释是：模型在读取原文前，获得了比单次相似度候选更明确的资料范围；它可以同时发现产品规则、版本限制和历史授权等不同证据类别。

此外，原文核验将“看起来相关”与“能够支持答案”分开。当前提示词要求检查数字、边界、否定、条件和例外；冲突保留双方来源；证据不足、调用失败或索引未读完时，必须返回 unresolved，不能把缺口改写成确定答案。

这使我们的成果不只表现为找到更多资料，也表现为**更容易保留回答成立的条件，发现资料冲突，并明确哪些问题还不能回答**。这种机制解释与团队关于“完整度好、回复准确率高”的观察一致，但仍需统一标注集来量化提升幅度。

已有局部验收证明了什么

项目在 2026 年 9 月 22 日的本地验收中写入 201 条合成索引，覆盖文档、经验记忆与工具说明三类来源。后台将其整理为三个目录；核对表明全部 201 个节点身份得到保留。只读 Agent 未获得目标节点 ID，仍从根目录逐层定位到三个指定目标，并返回与写入记录一致的来源引用。不存在的目标返回未找到；来源更新后，再次召回获得新定位。本地索引实测记录（AgentCoreService/docs/qa/index-live-20260922.md）

这些结果支持分页、导航、来源定位、负例处理与更新机制可用。测试来源是合成对象引用，不能把它解释为 201 个复杂业务问答全部正确，更不能据此宣称业务准确率达到 100%。

仍然存在成本和覆盖边界

索引摘要可能遗漏概念，生产流程可能漏建条目，模型可能误选目录，原文也可能过期。因此，完整阅读索引不等于完整读取知识，更不等于必然给出正确答案。

规模扩大后，flat 全量阅读会消耗更多输入与推理预算；hierarchical 则将部分成本换成目录选择风险。原文读取还需要处理分页、截断和固定版本。第四版改善质量的同时，让相似请求重复执行整套流程的成本更加明显，推动了第五版。

2.7 第五版 在记忆索引召回上增加语义缓存

复用历史召回工作

用户经常以不同措辞询问同一件事，例如“企业版能离线安装吗”和“企业版有没有离线部署方案”。在适用条件相同、资料没有变化的情况下，每次重新读索引、找来源并推理，包含大量重复工作。

第五版保留第四版作为完整召回路径，在前面增加语义缓存。缓存发现的是相似历史任务及其结果，而不仅是相似的原始文本片段。它将已完成的检索与归纳工作变成后续请求可复用的候选。

当前实现不把相似度命中直接当作答案

当前语义缓存流程可以概括为：

```
新问题与明确条件
→ 在匹配作用域内搜索语义缓存
├── 没有有效候选 → 执行完整记忆索引召回
├── 存在有效候选
    → 返回候选 key 与原任务输入
    → 上游比较当前问题和适用条件
    → agent_cache_read 读取选中结果
    ├── 适用且足够 → 复用相关召回结果
    └── 不适用或已失效 → bypassCache=true 执行完整召回
```

候选响应明确标注 `executed=false`，不能将候选列表误认为本次召回已重新执行。缓存读取会再次校验有效性；绕过缓存只跳过发现过程，成功执行的新结果仍可进入缓存。语义缓存实现（`AgentCoreService/src/AgentCore.Infrastructure/Persistence/TaskResultCache.Semantic.cs`）

这也是第五版与第一版中“相似度直接决定输入材料”的一个重要差别：第五版将相似度用于发现可复用工作，是否适用仍要结合问题条件与结果内容判断。

为什么相似问题的性能明显改善

第四版一次完整召回可能包含索引分页、条目选择、多个来源读取和结果归纳。如果缓存结果适用，第五版可以减少或省去这一串重复操作，转为候选比较和结果读取。

我们的工程观察是，这显著改善了相似问题的召回性能。这里强调的是重复或相似工作负载下的改善，不意味着首次提问也一定更快。缓存未命中时，系统仍需执行完整召回，并承担额外的缓存查询开销；命中后也不是零成本，仍有适用性判断和结果读取。

收益取决于相似请求比例、有效命中率、资料更新频率，以及完整召回本身有多昂贵。第 2.2 节通过作者确认的数据与加权计算展示这种关系；本文使用“显著”描述工程使用中的改善幅度，未进行统计显著性检验。

相似问题不一定能共享答案

“企业版能否离线部署”和“试用版能否离线部署”可能在向量空间中很接近，但条件不同；“产品支持离线部署”和“当前客户有权使用离线部署”更是不同问题。问题必须携带影响判断的明确条件，不能只把一句脱离上下文的短问句用作复用依据。

现有缓存实现校验应用、实例、Agent、配置标签及缓存有效代次，读取时还校验候选资格。业务中的客户、版本、角色等细粒度差异，需要通过输入和适当的过滤字段表达，不能假定系统会自动理解所有隔离边界。

文档和记忆更新后，旧结果也可能失效。平台提供 TTL、事件失效、清理代次和主动绕过机制；召回模板列有文档、记忆及索引更新相关失效事件。具体部署仍需启用合适的缓存设置，不能仅凭模板支持缓存就假定语义缓存已正确生效。召回 Agent 配置（AgentCoreService/src/AgentCore.Runtime/CoreAgents/Definitions/memory-retrieval/agent.json） · 缓存配置与使用说明（AgentCoreService/src/AgentCore.Api/Mcp/McpGuide.cs）

最后，平台缓存的是符合资格并正常完成的任务输出，不会自动判定其中每条业务结论都正确。结果即使带有未解决项，也不代表一定被排除。上游必须检查完整答案及 unresolved，不能把“来自缓存”当成正确性认证。

2.8 产品落地：AI 接待员中的实际表现

最新记忆检索方案已在 AI 接待员（aireceptionist.bosheng.online）产品中应用。对我们而言，这条技术路线的成果既包括召回协议与缓存机制的实现，也包括它们在实际产品中的使用效果。

团队的实际使用反馈是：最新方案表现良好，记忆召回完整度和回复准确性较好；在相似问题能够有效复用历史结果时，响应效率得到明显改善。 记忆索引帮助模型定位并核验相关资料，语义缓存则减少重复问题再次执行完整召回流程的成本。两者结合，使这一方案具备了实际产品应用价值。

产品使用中仍存在两种不同体验：可有效复用缓存的相似问题响应较快；全新问题、条件变化或缓存失效后的问题，需要执行完整索引召回，等待时间仍然较长。因此，“实际表现良好”包含对回答质量与整体使用效率的综合评价，并不意味着所有请求都具有同样低的延迟。

上述产品表现来自团队的实际使用反馈。第 2.2 节列出作者确认的方案对比数据；AI 接待员的具体线上表现还会受到部署模型、记忆规模、问题分布和缓存有效复用率的影响，不能将对比表中的平均值得理解为所有线上请求的固定表现。

3 技术路线带来的认识与后续方向

3.1 五次迭代是责任分工的逐步明确

第一版把重点放在快速找到相关片段；第二版补充关系组织；第三版让模型参与检索决策；第四版为模型提供可阅读的资料范围和可核验的来源；第五版复用已经完成的召回工作。

这并不意味着向量检索、图结构与模型导航必须互相替代。向量检索依然适合大规模候选发现，图结构适合表达关系，记忆索引适合提供导航与原文定位，缓存则适合减少重复计算。我们当前选择记忆索引与语义缓存的组合，是因为它较好地对应了我们对证据完整性和相似请求性能的需求。

3.2 把检索相关性 证据完整度和回答准确性分开

找到与问题相关的资料，只是第一步。一个问题还可能需要多类证据共同成立；即使证据已经齐全，模型仍可能误解条件或生成错误结论。

评价层次	应检查的问题
检索相关性	返回的资料是否与问题有关，有多少无关内容
证据完整度	回答所需的事实、条件、例外与版本是否齐全
来源保真度	摘要和结论能否回到原文，引用是否支持对应陈述
回答准确性	在这些证据下，最终答案是否成立
不确定性处理	无法核验、资料冲突和未找到时，是否明确说明缺口

第四版的关键价值，是把关注点从“命中了什么”推进到“是否拿到了回答需要的证据”。但不能用读取了多少页或调用了多少工具，代替完整度和准确性的实际评价。

3.3 质量路径与快速路径必须同时保留

记忆索引召回提供完整执行路径，语义缓存提供可复用时的快速路径。快速路径不能让系统失去回到原文的能力。

当问题条件变化、候选不充分、来源过期或资料存在冲突时，系统应执行完整召回。资料不变且条件等价时，则没有必要为证明“认真检索”而重复扫描索引。两条路径的切换是否合理，是第五版需要持续评估的能力。

这种分工也解释了我们为何仍让 LLM 参与召回决策：模型可以利用任务语义选择和核验证据，宿主负责分页契约、来源权限、版本状态与缓存有效性。模型承担的是有范围的业务判断，而非自行猜测所有底层状态。

3.4 用什么指标验证下一阶段成果

后续量化评估应使用同一记忆快照、同一题集和明确记录的模型配置，将新问题与相似问题分开统计。至少需要覆盖以下四类请求：

- 1. 首次出现的问题，用于检查完整索引召回的正确性和成本。
- 2. 同义改写且条件相同的问题，用于检查有效缓存复用。
- 3. 措辞相似但客户、版本或条件不同的问题，用于检查错误复用。
- 4. 资料更新或删除后的问题，用于检查缓存失效与来源新鲜度。

指标	统计重点
必要证据覆盖率	对人工标注的证据集合，检查是否覆盖关键条件与例外
最终回答准确率	联合检查答案、适用范围和引用，不能只看文本相似
有效缓存复用率	统计真正适用并被采用的结果，而非所有相似候选
错误缓存复用率	条件不符、来源过期或答案不充分时是否仍被采用
端到端延迟	分开报告冷请求、有效命中与未命中回退的 P50、P95
累计成本	统计主模型、子 Agent、嵌入、工具与缓存读取开销
更新后正确性	记忆变更后，旧结论能否及时退出有效召回路径

比较成本时不能只计算最终回答那一次推理。比较性能时也不能只展示缓存命中的请求，隐藏未命中开销与失效后的重建成本。

3.5 后续可行的改进方向

对记忆索引而言，下一步可以继续改进条目摘要的覆盖范围、目录命名和条件表达，让模型更容易找到不同证据类别。索引应保留来源与版本，不能为了缩短摘要而丢失例外条件的存在提示。

对模型决策而言，可以通过专项样本训练其识别证据不足、合理结束召回，以及拒绝不适用的缓存候选。监督训练或强化学习值得探索，但本次路线复盘没有提供相应训练实验，不能预先承诺效果。

对语义缓存而言，重点应是提高有效复用率，而非单纯提高候选命中率。更清楚的输入条件、合适的过滤字段、可靠的更新失效，以及对错误复用的持续评估，比一味降低相似度阈值更有价值。

五次迭代后，我们更看重的是：让系统知道有哪些资料，能够回到原文核验，并在条件允许时复用已经完成的工作。记忆索引改善召回质量，语义缓存减少重复成本；两者结合，是我们目前分享给通用 Agent 开发者的主要工程成果。

参考资料与实现入口

外部资料用于说明技术类别，不作为本项目性能提升的证据。

1. Lewis 等, Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks。
2. Microsoft GraphRAG 索引说明与查询说明。
3. 实例级通用索引设计 (AgentCoreService/docs/instance-index-service.md) 与单层索引及显式建组 (AgentCoreService/docs/flat-index-groups.md)。
4. 记忆召回 Agent 提示词 (AgentCoreService/src/AgentCore.Runtime/CoreAgents/Definitions/memory-retrieval/system-prompt.md) 与配置 (AgentCoreService/src/AgentCore.Runtime/CoreAgents/Definitions/memory-retrieval/agent.json)。
5. 语义缓存候选与读取实现 (AgentCoreService/src/AgentCore.Infrastructure/Persistence/TaskResultCache.Semantic.cs) 及 MCP 缓存契约 (AgentCoreService/src/AgentCore.Api/Mcp/McpGuide.cs)。
6. 2026 年 9 月 22 日本地索引实测 (AgentCoreService/docs/qa/index-live-20260922.md)。该记录验证局部机制，不代表五版本统一业务准确率评测。

本文采用标准 Markdown，无需外部图片或特定渲染插件。实现入口以源码仓库内路径标注，供项目维护者定位，不作为本文附带的公开文件；外部参考资料保留可访问链接。