借助 RAG 融合扩展检索增强生成：行业部署经验

Luigi Medrano戴尔科技 美国德克萨斯州奥斯汀
Luigi.Medrano@dell.com

Arush Verma戴尔科技 美国德克萨斯州奥斯汀
Arush.Verma@dell.com

Mukul Chhabra戴尔科技 美国德克萨斯州奥斯汀mukul.chhabra@dell.com

arXiv:2603.02153v1 [cs.IR] 2 Mar 2026

摘要

检索增强生成（RAG）系统通常采用检索融合技术，例如多查询检索和倒数排名融合（RRF），以提高文档召回率，其前提是假设更高的召回率能够带来更好的答案质量。尽管这些方法在独立的检索基准测试中表现出稳定的提升，但它们在真实的生产环境约束下的有效性仍未得到充分研究。本研究评估了生产级检索增强生成流水线中的检索融合，该流水线基于企业知识库运行，并具有固定的检索深度、重排序预算和延迟约束。

在多种融合配置下，我们发现检索融合确实提高了原始召回率，但经过重排序和截断后，这些收益大多被抵消。在我们的设置中，融合变体在知识库级 Top-k 准确率上未能胜过单查询基线；在若干配置下，Hit@10 从 0.51 降至 0.48。此外，融合还会因查询改写和候选集扩大而带来额外的延迟开销，却未能相应提升下游效果。

我们的分析表明，在应用符合实际的重排序限制和上下文预算后，以召回率为导向的融合技术呈现出收益递减的趋势。我们得出结论：检索层面的改进并不能可靠地转化为生产环境中的 RAG 系统的端到端收益，并主张采用能够综合考量检索质量、系统效率和下游影响的评估框架。

1 引言

检索增强生成（RAG）系统已成为一种广泛采用的方法，用于将大语言模型（LLM）建立在外部知识库的基础上（Lewis 等人，2020年）。通过基于检索到的文档进行生成，RAG 系统旨在提高事实准确性并减少幻觉。然而，其有效性仍高度依赖检索质量：当缺少相关证据或其排名不佳时，无论模型容量如何，下游生成都会受到限制。

提升检索鲁棒性的一种常见策略是采用检索融合技术，例如多查询检索和倒数排名融合 (RRF) (Cormack 等人，2009年)。通过对查询进行多种改写，或合并不同检索轮次的结果，融合方法旨在提高文档召回率并减少遗漏证据。以往研究表明，这些技术可以单独提升以召回率为导向的指标，因此许多 RAG 流水线都采用了这些技术。

尽管检索融合广受欢迎，但它在生产环境中的 RAG 系统里所产生的端到端影响仍不明确。在实践中，检索还受到其他约束，包括固定的检索深度、有限的重排序能力、上下文窗口预算以及严格的延迟要求。在这些条件下，原始召回率的提升可能会被冗余、冲突上下文或下游引入的截断效应抵消。

本文探讨检索融合能否在生产环境风格的 RAG 场景中提升系统级性能。我们并未提出新的融合算法，而是着重评估检索层观察到的召回率提升，能否在重排序和上下文选择之后转化为有意义的改进。结果表明，尽管融合增加了候选多样性，但应用现实的流水线约束后，这些收益往往会被抵消，最终对知识库级准确率和延迟的影响有限，甚至产生负面影响。

1.1 贡献

本文作出以下贡献：

• 我们对生产环境导向的检索融合方法进行了评估，该方法应用于 RAG 流水线，并在固定检索深度、重新排序预算和延迟约束下运行。

• 我们通过实证分析检索层召回率提升与下游知识库级准确率之间的关系，发现融合带来的召回率提升在重排序和截断后往往会被抵消。

•我们指出了融合带来的实际权衡，包括冗余增加和额外的延迟开销，这些因素限制了融合在真实检索增强生成部署中的有效性。

2 背景

本研究采用检索增强生成（RAG）系统，以支持面向大型、公开的企业知识库进行事实性问答，因为单独使用大语言模型（LLM）并不足够。通过依据检索到的证据来生成内容，检索增强生成（RAG）能够在访问控制、延迟和内容不断变化等运营约束下，生成有依据且可追溯的回答。

在实践中，早期部署表明，端到端性能往往受检索而非生成的限制。简短、不够明确或词汇表达较弱的查询——在真实用户交互中十分常见——即使使用性能强大的混合检索器，也经常无法找到正确的知识库文章。这些检索失败会传递到下游，限制生成器的输出；即便大语言模型能力出色，最终回答仍可能不完整或质量较低。

为解决这些召回失败问题，研究者提出了多查询检索和倒数排名融合（RRF）等检索融合技术，通过聚合来自多个检索信号的结果来扩大文档覆盖范围（Rackauckas，2024年）。先前研究报告称，这些技术能持续提升召回率导向指标，尤其是在查询含糊或较短时表现突出，因此推动了现代 RAG 流水线采用融合技术。

然而，在生产系统中，检索到的候选项需要经过重排序、去重和截断，以适应固定的上下文窗口。这引发了一个问题：融合带来的召回率提升能否经受住下游约束。这些观察促使我们开展本研究：我们不孤立地评估融合，而是考察其在检索层面的优势能否转化为更高的证据可用性，并在生产级 RAG 流水线中带来实际收益。

3 问题陈述

检索融合常用于检索增强生成（RAG）系统，其依据的假设是，提高检索召回率能够带来更准确、更可靠的答案。在企业环境中，部署内部 RAG 系统旨在缩短支持和运维查询的解决时间，因此答案质量至关重要。然而，融合也会因增加检索轮次、融合逻辑和下游处理而带来额外的系统复杂性和延迟。

本研究考察检索融合带来的额外复杂性和延迟，能否为企业规模的内部 RAG 系统带来可衡量的答案质量提升。在生产环境的流水线中，答案质量取决于相关知识库文章是否出现在重排序和截断后的最终 Top-K 排序上下文中；因此，只有当检索层的改进能够在现实的企业级约束下转化为端到端证据可用性的提升时，这些改进才有意义。

我们检验以下假设：

• **H1：** 与强大的混合检索基线相比，检索融合并未显著提升 Top-K证据准确性。

• **H2：** 融合带来的召回率提升大多会被下游重排序和上下文截断抵消。

• **H3：** 融合带来的任何收益都仅限于召回率不足的查询场景，对整个企业系统的性能没有实质影响。

4 方法论

我们在一个企业规模的 RAG 检索流水线中评估检索融合，该流水线旨在尽可能贴近生产部署条件。本方法论的目标并非孤立地优化检索性能，而是评估融合带来的召回率提升能否在后续排序阶段持续保持，并在现实的企业级约束下提高相关证据的可用性。

4.1 系统基线

我们采用标准的混合检索策略，将稀疏词汇匹配（BM25）与稠密语义检索相结合，随后针对每个查询进行交叉编码器重排序，这反映了大规模信息检索系统中的常见做法（Robertson和Zaragoza，2009年）。

• **基线**：混合检索（BM25），随后进行交叉编码器重排序（FlashRank）、去重，并截断至固定的Top-K 上下文（Pradeep 等人，2023 年）。

所有基线共享相同的文档语料库、分块策略、元数据过滤规则、检索深度、重排序模型以及排序超参数。这确保所观察到的差异仅归因于检索融合的有无。在所有实验中，检索深度固定为 $K = 10$，所有下游重排序、融合和截断操作均在这一固定候选预算上进行。

4.2 融合配置

我们评估一种忠实于生产环境的双查询融合策略。对于每个输入查询，检索使用以下方式执行：

• **Q1**：原始用户查询。

• **问题2**：对原始查询进行一次由大语言模型生成的释义或改写。

对于每个查询，混合检索器会返回排名靠前的K 个文档块。所有文档都按固定大小切分为文本块，每块包含 512 个词元，并使用 Granite 嵌入模型进行嵌入。查询改写（问题2）由基于 LLaMA 的语言模型根据问题1生成，所遵循的提示策略见 *A.4.3*。

问题1和问题2的检索结果分别通过基线所用的同一个交叉编码器重排序器独立处理。然后，对两个重排序列表应用倒数排名融合（RRF），以生成融合后的候选集。

重要的是，融合应用于每个查询重排序之后，这反映了生产环境约束：检索输出在聚合之前会分别进行排序。所有下游组件，包括去重、截断和评估，均与基线保持不变，从而确保观察到的差异反映的是融合的影响，而非嵌入、分块或查询生成选择所导致的影响（见第 6 节）。

4.3 数据集和真实值

实验基于包含 $N = 115$ 个企业风格支持查询的合成数据集开展。每个查询均源自一篇已知的知识库（KB）文章，并与其来源文章配对作为真实标注，从而能够在符合实际的文档分布下，对检索行为进行受控评估。检索相关性在知识库文章级别进行评估，而排序则使用分块感知标识符在文本块级别进行。

所有检索到的文本块及其相关元数据，均使用 LangChain 框架提供的标准化文档抽象来表示；该框架用于协调检索增强生成流水线中的检索和排序组件（Kakkerla 等人，2024年）。产品元数据仅用于限定检索范围，不用于排序或评估。

4.4 评估协议

端到端性能使用 Top-K 知识库级命中指标进行评估。对于每个查询，我们确定重排序和截断后，最终 Top-K 个排序后的上下文中是否至少包含一个属于真实知识库文章的文本块。报告的指标包括 Top-1、Top-3 和 Top-10 准确率，并在评估集上进行汇总。

该评估框架用于衡量检索与融合策略是否成功地向下游组件呈现相关证据，而不依赖主观的生成质量评判。

4.5 运行约束

所有实验均在企业部署约束下进行，这些约束会直接影响检索融合的有效性：

• 固定的重排序容量限制了能够进行有效评分的候选项数量。

• 上下文窗口限制要求在生成前截断排序结果。

•基于元数据的检索过滤，以确保产品相关性和访问控制。

5 框架概述

评估 RAG 系统中的检索融合，需要做的不仅仅是单独比较检索指标。尽管融合技术通常以面向召回率的目标为驱动，但其

下游影响取决于检索候选项如何在完整流水线中与排序能力、上下文限制及运行约束相互作用。该评估框架旨在揭示这些相互作用，并找出融合收益未能转化为有意义的系统级改进的具体环节及原因。

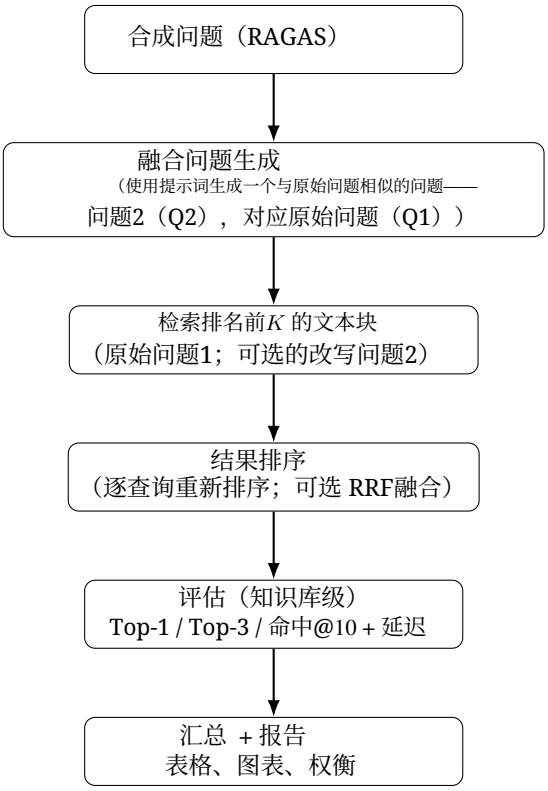

图1：评估流程的简化概览。

本分析围绕在将融合引入贴近生产环境的检索增强生成流水线时反复出现的一系列注意事项展开，包括重排序器饱和、上下文冗余、证据冲突以及运维开销增加。本研究并未将这些影响视为次要问题，而是明确衡量它们对相关证据能否保留在最终排序后的上下文中的影响。

实验设置遵循附录*A* 中所述的检索与排序流水线，并如以下图示（图*1*）。对于每个查询，我们将强混合检索基线与双查询融合变体进行比较，同时保持所有下游组件不变。检索结果按文本块级别排序，使用倒数排名融合进行融合，并在评估前截断至固定的 Top-K 上下文。

主要目标是确定是否融合能在重排序和截断之后提高正确知识库证据的可用性。为此，重点关注 Top-K知识库级命中指标，将其作为下游效用的代理指标。这一选择使融合在检索与排序阶段中的效果得以隔离，而不会将结果与生成特有的变异性混为一谈。

通过围绕具体失效模式组织评估，并独立衡量其影响，该框架能够更精准地评估检索融合何时有益、何时效果被抵消，以及何时会带来负面权衡。

6 结果

我们按照评估框架中确定的主要注意事项来组织并呈现结果。对于每种情况，我们在相同的下游条件下，将融合配置与基线进行比较，并报告检索、排序和运行结果。这些结果共同说明了检索融合在实际流水线约束下的表现，以及观察到的效果与第 3 节所述假设的一致程度。

表 1–3 展示了附录 A.4.3 中所述三种融合提示策略的端到端准确率，其范围从保守的查询改写到明确的多样性最大化。结果采用知识库级 Top-K 命中指标进行报告，反映在固定检索预算下，相关知识库证据在重排序和截断后是否仍然保留。

6.1 重排序后召回率提升被抵消

在所有评估配置中，检索融合始终提升上游召回率，证实了其在检索层的预期效果。然而，当检索候选项经过交叉编码器重排序并截断至固定的 Top-K 上下文后，这些提升并未始终反映在下游知识库级准确率上。如表1、2和3所示，融合变体在 Top-1、Top-3 和 Hit@10 指标上的准确率与基线相当，某些情况下甚至更低。

这些结果与**H2**一致。在固定的重排序和截断约束下，由融合引入的额外候选项往往

方法	Top-1	Top-3	Hit
基线	**30.43%**	37.39%	**51.30%**
基于 RRF 重排序 问题1	29.57%	**38.26%**	47.83%
基于 RRF 重排序 2	15.65%	23.48%	47.83%
倒数排名融合 问题1 问题2	25.22%	**38.26%**	47.83%

表1：融合提示 1 的汇总指标表

方法	Top-1	Top-3	Hit
基线	**29.57%**	36.52%	**50.43%**
基于 RRF 重排序 问题1	**29.57%**	39.13%	46.09%
基于 RRF 重排序2	7.83%	17.39%	46.09%
倒数排名融合 问题1 问题2	27.83%	**40.87%**	46.09%

表2：融合提示 2 的汇总指标表

方法	Top-1	Top-3	Hit
基线	**29.57%**	36.52%	**50.43%**
基于 RRF 重排序 问题1	**29.57%**	40.00%	44.35%
基于 RRF 倒数排名融合2 重排序	10.43%	19.13%	44.35%
倒数排名融合 问题1 问题2	27.83%	**40.87%**	44.35%

表 3：融合提示 3 的汇总指标表

无法进入最终的排序上下文。虽然召回率的提高扩大了候选项池，但下游排序能力似乎限制了这些收益能够影响最终提供给语言模型的证据的程度。

6.2 实际运行的前三名深度上的统计显著性

不同融合变体的 Top-3 准确率差异很小，因此我们进行配对显著性检验，以判断这些变化是真实存在，还是仅仅由噪声造成。我们将每个查询视为二元命中/未命中结果：如果重排序和截断后的最终 Top-3 排序列表中，包含来自真实知识库文章的任意文本块，则该查询记为命中（这与生成时传递给大语言模型的 Top-3 上下文相同）。由于每种方法都在相同的 $N = 115$查询上进行评估，因此在将每种融合方法与基线进行比较时（仅针对问题 1），我们对配对结果使用 McNemar 精确检验。由于我们会在每种提示词类型中进行多次比较，因此应用 Benjamini–Hochberg 校正（假发现率 $= 0.05$）。我们将基于倒数排名融合重排序的问题 2排除在本分析之外，因为它在各种提示词下的表现都明显逊于基线，并不代表可行的部署方案 - 表4 显示，提示词 2/3 的 Top-3 命中率略高（$+2.61$ 至 $+4.35$ 个百分点），但校正后这些提升均不具有统计显著性（均 $p_{\text{adj}} \geq 0.125$）。换言之，采用特定参数进行融合会略微提升检索准确率，但效果不够稳定，无法令人信服地证明在生产环境中值得承担额外的融合开销。

P	方法	Top-3	Δpp	p	p_{adj}
P1	倒数排名融合问题1 问题2	38.26%	+0.87	1.0000	1.0000
P1	重排序on 倒数排名融合 问题1	38.26%	+0.87	1.0000	1.0000
P2	倒数排名融合问题1 问题2	40.87%	+4.35	0.1250	0.1667
P2	重排序开启 倒数排名融合 问题1	39.13%	+2.61	0.2500	0.2500
提示3	倒数排名融合问题1 问题2	40.87%	+4.35	0.1250	0.1250
提示3	对倒数排名融合问题1重排序	40.00%	+3.48	0.1250	0.1250

表 4：配对Top-3显著性与基线对比。

6.3 上下文冗余与冲突证据

融合带来的精确重叠很少，Q1 和 Q2 的检索列表之间几乎没有交集，这意味着改写后的查询（Q2）通常会获取一组不同于原始查询的候选项。表5清楚地说明了这一点：对于提示词 2 和 3，Jaccard 相似度约为0。09，尽管并集扩展到了大约 15 个不同来源。我们报告重叠率和 Jaccard 相似度，是因为它们能说明 Q2 是否确实带来了新内容（低重叠率／低 Jaccard 相似度），而不是仅仅以不同顺序返回相同的文档；并集大小则显示合并 Q1 和 Q2 后候选池扩大的程度。

单凭重叠率低并不是问题——生成改写查询的目的正是如此。问题在于，许多“新”候选项并没有带来真正的新信息。它们往往重复相同的观点（来自知识库相邻部分的近重复文本块），或在相关文章中引入部分相互冲突的指导。这会降低重排序和截断的稳定性，还可能将真正有用的证据挤出 Top-K 窗口。

这也有助于解释为什么基于 RRF 重排序 q2 的表现如此糟糕。使用 RRF 融合问题1和问题2后，再用问题2对融合后的列表进行重排序，评分可能会过度偏向问题2特有措辞（以及改写过程中引入的任何偏移）。实际上，与问题1一致的优质文档可能会被降级或剔除，这与提示词 2 和 3 中基于 RRF 重排序 q2 的 Top-1 大幅下降相吻合。

总体而言，融合拓宽了覆盖范围，但语义冗余和以 Q2 为锚点的重排序可能会降低最终上下文的质量，这与 **H2** 一致。

P	Ov@10	Jac@10	Un@10	Uniq@10
P1	2.87	0.279	13.20	7.82
P2	1.09	0.087	15.29	7.77
P3	1.10	0.094	14.99	7.77

表 5：精确的 Q1–Q2 重叠（Top-10）。

6.4 对端到端证据可用性的有限影响

尽管上游召回率持续提升，融合并未在整个评估集上带来 Top-K 知识库级准确率的稳定提升。在若干配置中，与基线相比，融合略微降低了 Top-1 和 Top-3 准确率，这表明新增候选项可能会干扰排序效果，而非使其提升。

这些发现与 **H1** 一致。应用下游约束后，检索层的改进并不能可靠地转化为更好的端到端证据可用性。在此实验设置中，不同融合提示变体和重排序锚点的结果均一致，这表明观察到的现象并非由某一种配置选择所导致。

6.5 延迟与流水线开销

融合会带来额外延迟，原因包括增加检索轮次、融合逻辑以及更多下游处理。如表 6所示，虽然中位延迟仍在可接受的运行范围内，但在各次实验中，尾延迟始终高于基线。随着检索深度和查询量增加，这种影响会进一步累积。

指标	平均耗时（秒）
融合问题生成	0.89
基线检索时间（问题1）	54.60
检索时间（Q1 + Q2 合并）	65.98
倒数排名融合时间	0.012
FlashRank 时间（$K = 10$）	0.26

表 6：所有数据点的平均延迟组成部分。

我们观察到，次级融合流程会额外产生 0.89 秒的开销，包括融合问题生成、倒数排名融合以及额外的重排序轮次。在处理两个查询而非一个查询时，检索延迟也会增加，这反映了额外嵌入和搜索操作所带来的成本。

尽管假设中未明确记录延迟，但这些结果通过证明融合会产生可测量的运营成本，却未带来相应的证据准确性提升，从而为 **H1** 提供了背景。在吞吐量和响应能力至关重要的生产环境中，这种权衡进一步限制了融合的实际价值。

6.6 查询切片分析

按查询特征进行细分后可以发现，对于一小部分召回稀缺查询，融合带来了有限的收益，尤其是对于基线检索器未能在前-K 个结果中呈现相关知识库文章的查询。在这些情况下，查询改写偶尔能够呈现出基线检索器遗漏的相关证据。

这些观察在一定程度上支持**H3**。尽管融合能带来局部收益，但这些收益仅适用于少数工作负载，且不会实质性影响系统整体性能。随着语料库覆盖率和排序能力的提升，融合的边际效用会逐渐减弱，这进一步印证了其效果高度依赖具体上下文这一结论。

7 结论

本研究在一个贴近生产环境的检索增强生成检索与排序流水线中，对检索融合进行了系统评估。尽管融合能够稳定提升上游召回，但我们的结果表明，这些收益往往会被下游重排序和上下文约束抵消；在若干配置中，融合还会导致可用证据减少、延迟增加。

通过针对重排序器饱和、上下文冗余和运维开销等具体流水线瓶颈评估融合，我们证明，以召回为导向的改进并不会自动转化为下游效用。在成熟的企业系统中，排序能力和上下文选择是影响端到端性能的主导因素，其重要性往往超过扩大检索广度所带来的收益。

我们的研究结果表明，融合的收益高度依赖具体情境，且主要局限于召回率不足的查询场景。对于大多数企业工作负载，融合会增加系-

统成本，却无法显著提升证据可用性。**虽然融合可能使个别检索指标略有改善，但这些收益往往不足以抵消其带来的额外系统复杂性、计算开销以及生产环境中的运营成本。我们的评估中观察到的下游收益有限，这表明，对于大多数企业检索增强生成部署而言，更简单、易于维护的架构或许能提供更优的成本效益权衡。**

参考文献

Gordon V. Cormack、Charles L. A. Clarke 和 Stefan Buettcher。2009年。倒数排名融合优于 Condorcet 方法和单独的排名学习方法。载于 第*32*届 *ACM SIGIR* 信息检索研究与发展国际会议论文集，第 758–759 页。Sahil Es、Jithin James、Kumar Shridhar Pradeep、Salman Khan 和 Brandon Talley。2023年。Ragas：检索增强生成的自动评估。*arXiv* 预印本 *arXiv:2309.15217*。Shiva Kumar Kakkerla、Krishnaveni Yalla、Pujitha Pathipaka、Yashwanthini Irukulla、Sriharshini Panthangi 和 Rohith Adupa。2024年。使用 LangChain 框架和生成式人工智能的问答模型。载于 *2024* 智能系统与机器学习会议（*ISML*），第 200–205 页。Patrick Lewis、Ethan Perez、Aleksandra Piktus、Fabio Petroni、Vladimir Karpukhin、Naman Goyal、Heinrich K¨ uttler、Mike Lewis、Wen-tau Yih、Tim Rockt¨aschel、Sebastian Riedel 和 Douwe Kiela。2020年。面向知识密集型自然语言处理任务的检索增强生成。载于 神经信息处理系统进展，第 33 卷，第 9459–9474 页。Kumar Shridhar Pradeep、Pablo Moreno 和 Srikanta Muthukrishnan。2023年。FlashRank：使用轻量级语言模型实现超快速重排序。*arXiv* 预印本 *arXiv:2311.07179*。Zackary Rackauckas。2024年。RAG-Fusion：检索增强生成的新思路。《自然语言计算国际期刊》，13(1):37–47。Stephen Robertson 和 Hugo Zaragoza。2009年。概率相关性框架：BM25 及其扩展。信息检索基础与趋势，3(4):333–389。

A 实验流水线与评估方法论

本附录记录了一个端到端流水线，该流水线用于评估在大型企业知识库上运行的检索增强生成（RAG）系统中的检索、融合和重排序策略。这里的目标是确保透明：输入了哪些数据、产生了哪些中间产物、如何应用排序，以及如何计算指标。主论文侧重于结果；本附录则详述这些结果背后的机制。

A.1 术语与核心对象

本附录通篇使用以下术语。

A.1.1 问题集

该**问题集**是用于探测检索系统的客户支持类查询集合。在本研究中，问题通过**使用 RAGAS 合成生成**。合成生成能够在大规模进行受控实验，同时仍可生成类似真实支持交互的查询（E 等人，2023年）。

A.1.2 知识库（KB）

该**知识库（KB）**是一个大型文本语料库，包含经过整理的领域知识，例如故障排除指南、已知问题和操作流程。知识库以书面形式呈现主题专家（SME）的知识，并作为所有实验的检索语料库。

A.1.3 知识库文章与文本块

一篇**知识库文章**是聚焦于特定主题的单份文档。为了进行检索，每篇知识库文章都会拆分为更小的文本片段，称为**文本块**。每个文本块都存储为**LangChain 文档**对象。

采用文本块划分，是因为检索模型通常比起完整文档，能更可靠地匹配较小且语义连贯的片段，尤其是在文章较长或涵盖多个子主题时。

A.1.4 索引

我们的 **索引** 是一个知识库文章集合，这些文章通过业界标准的嵌入模型将文本转换为向量。上述知识库文章文本块存储在此索引中。在检索阶段，检索器会执行嵌入后的查询（使用相同的嵌入模型）与索引之间的相似性搜索。

A.1.5 检索结果文档

A**检索结果**是一个 LangChain 文档，表示一个检索到的文本块。每个结果包含：

- 页面内容：检索返回的分块文本
- 元数据：描述文本块来源的字段，至少包括其**来源知识库文章**和**块索引**

A.1.6 Top-K结果

对于每个查询，检索器都会返回**Top-**K 个文档（文本块）的排序列表。K 是我们定义的值，在本例中 K=10。将多个查询的结果记录在同一个列表中时，我们采用固定的位置约定：

位置	含义
$1 \dots K$	问题1检索结果
$K{+}1 \dots 2K$	问题2检索结果

表7：K 个检索结果

A.2 数据集构建与真实值

A.2.1 查询数据集

每次实验运行都基于一个数据集，其中每一行对应一个查询实例。每行至少包含：

- 问题：自然语言查询文本（通过 RAGAS 合成生成）
- 真实标注：生成该问题所依据的“真实”知识库文章（第 A.2.2 节）

实际上，可能还会包含其他元数据（例如产品类别标签、查询长度）。这些元数据仅用于筛选和语料库范围限定（第A.4.1节），不会直接用作排序信号。

A.2.2 真实标注标签

相关性是在**知识库文章层面**定义的。每个问题都关联一个或多个有效的“真实标注”知识库文章：

• 知识库文章 ID：被认为适用于该查询的一个或多个知识库文章标识符

在评估期间，如果检索到的结果中，排名前-K 的结果包含任意一篇真实知识库文章（通过其任意一个文本块检索到），则视为**命中**。

A.3 文档表示

A.3.1 分块策略

所有知识库文章在索引之前都会经过预处理，拆分为文本块。每个文本块都表示为一个包含页面内容（文本）和元数据的 LangChain 文档。元数据包括：

• 来源知识库：该文本块所属知识库文章的标识符

• 块索引：该文本块在文章中的位置

• 用于筛选和访问控制的可选字段（如适用）

A.3.2 分块感知标识符

由于多个文本块可能来自同一篇知识库文章，我们将文本块视为检索的原子单位。为避免在融合或重排序过程中意外地将多个相关文本块合并为单个条目，我们为每个文本块分配一个唯一标识符：

格式 来源 kb::chunk=文本块索引	**示例**
KB123456::chunk=0	

表 8：知识库文本块引用的来源标识符格式。

这个分块感知标识符会贯穿检索、融合和重排序过程。重要的是，来自同一篇知识库文章的多个文本块出现在同一个 top-K 列表中是正常的。我们希望在重排序过程中保留文本块的位置，以确保在使用同一篇知识库文章中的多个文本块来组织答案时，不会丢失其中任何信息。

A.4 检索阶段

A.4.1 元数据过滤

通过轻量级的元数据过滤层，将检索范围限定在知识库的相关部分。这样做是出于实际运营需要，而非理论考量：限制候选集可以提高精确率并降低延迟，因为这能排除明显不属于查询目标产品类别（或其他高层级分组）的文章。

筛选应用于之前的检索（即约束可搜索子集），且不会直接改变评分或排序逻辑。

A.4.2 用于融合实验的双查询检索

为评估融合策略，我们使用两种查询表述进行检索：

• **原始查询（问题1）**：原始问题文本

• **融合查询（问题2）**：由大语言模型生成的重新表述（例如释义），旨在找出其他匹配证据

对于每个查询（$Q1$以及可选的 $Q2$），检索器会以 LangChain 文档对象的形式返回排名前 K 的文本块。排序后的输出会按照前文所示的位置约定进行串联。

A.4.3 融合问题生成提示词

为了更好地解决我们的问题陈述，我们决定使用三种不同的融合生成提示词进行实验。这些提示词会与原始问题一同作为指令发送给大语言模型。随后，大语言模型会根据指令中定义的改写策略，生成一个相似的问题，并提供一些有关该问题的信息（如产品类别等）。

```
< 指令 | >
你是一名技术查询改写器。
你将收到一条技术
查询、案例详情和产品
信息。
重新表述查询，但不要添加
有关产品或案例的详情。

指南：
- 只给出输出，不要提供你的
相关推理
- 不要添加任何额外细节
仅限于输入中
input
– 保持直接简洁
</指令>
```

图2：提示类型1

< |instruction| >你是一名技术查询改写器。你的职责是改写给定的技术查询，以用于故障排查。请利用给定的信息改写技术查询，使其适用于RAG系统。指南：- 仅提供改写结果，不要说明理由 - 不要添加输入中未提供的任何额外细节 - 让查询精准且注重细节，但不要写得太长。</instruction>

图 3：提示类型2

< |instruction| >你是一名检索策略师。你的任务是生成多样且高质量的技术查询改写，以最大限度提高 RAG 融合流水线中的文档召回率。你只能返回一条高质量的改写。指南：- 仅提供输出，不要说明推理过程 - 除输入中提供的内容外，不要添加任何额外细节</instruction>

图 4：提示类型3

A.5 重排序与融合流水线

实验流水线评估了一个由多个阶段组成、类似生产环境的排序栈。并非每种配置都会用到所有阶段，但输出都会以一致的方式记录，以便直接比较。

A.5.1 按查询重排序（基线）

为了模拟在应用任何融合之前先对检索输出进行精炼的企业设置，每个查询的检索列表都使用 FlashRank 模块独立重新排序。由此得到：

- 基线：问题1检索结果的重排序列表
- 问题2基线：问题2检索结果的重排序列表

每个列表都以分块感知标识符构成的有序序列形式存储。如果来自同一篇知识库文章的多个文本块排名靠前，则会保留这些文本块。问题1的检索结果重新排序后形成的列表，就是我们的**基线**数据点。请参阅下方的表8、9和10。

A.5.2 倒数排名融合（RRF）

倒数排名融合（RRF）用于将多个排序列表合并为单一的融合排名。在本研究中，倒数排名融合应用于已经重排序的列表（基线、问题2基线）

对于每个排序列表 L_i，排名为 r 的文本块 d 贡献如下：

$$\text{score}(d) \mathrel{+}= \frac{1}{k+r}$$

其中，k 是平滑常数。

实现的关键特性：

- 融合在**分块级别** 上进行，并使用分块感知标识符。
- 每个文本块在每个列表中最多贡献一次（如果出现重复项，则使用最佳排名）。
- 输出是一个融合有序列表，记作倒数排名融合 q1 q2。 - -

A.5.3 融合后重排序

为了单独考察融合是否能从额外的语义细化中获益，我们会选择性地使用 FlashRank 对融合列表中的前-K 项进行重新排序。根据用作重排序锚点的查询表述，这会产生两种变体：

- 基于 RRF_重排序（问题1）：以问题1为锚点，对融合列表重新排序
- 基于 RRF_重排序（问题2）：以问题2为锚点，对融合列表重新排序

此阶段有助于区分：

- 单独**融合**的效果，与
- 融合 **+额外重排序**。

有关这些列表的评估，请参见下文表 8、9 和 10。

A.6 记录的输出

对于每个查询对（Q1、Q2）（问题）实例，实验会记录：

- 每种流水线配置的排名输出列表（逐查询重新排序、RRF融合列表、融合后重新排序）

- 检索延迟测量
- 重排序延迟测量
- 融合问题生成延迟测量（启用时）

所有排序列表均以有序序列的形式存储分块感知标识符，即使文本块文本或元数据随时间发生变化，也能确保评估的一致性。

A.7 评估方法

A.7.1 基于分块级排序的知识库级评估

尽管检索和排序以文本块为单位进行，但正确性是在知识库文章级别定义的。为了评估结果，我们通过移除分块后缀，将每个分块感知标识符映射回其对应的知识库文章：

$$\text{源知识库::文本块=索引} \quad \rightarrow \text{源知识库}$$

如果属于任意真实知识库文章的任意文本块出现在该方法的前 K 个排序结果中，则该方法对该查询**命中**。

这种设计符合基于检索增强生成的对话模型的工作方式。从正确文章中检索到任何有力的支持段落，通常就足以支持下游答案生成，即使具体文本块有所不同。

A.7.2 指标

对于每种排序方法，我们计算：

- **Top-1 准确率**：真实知识库文章排名第 1 的查询所占比例
- **Top-3 准确率**：真实知识库文章排名在第 1–3 位的查询所占比例
- **Hit@10（Top-10 命中率）**：真实知识库文章排名在第 1–10 位的查询所占比例

指标按每个查询计算，然后在整个数据集上汇总。

A.7.3 评估产物

主要的评估成果是一张**汇总指标表**，每种方法占一行，并设有 Top-1、Top-3 和 Hit@10 三列。这张表用于论文正文中的结果报告与比较。

方法	Top-1	Top-3	Hit
基线	0.3043	0.3739	0.5130
基于 RRF 重排序 问题1	0.2957	0.3826	0.4783
基于 RRF 重排序 2	0.1565	0.2348	0.4783
倒数排名融合 问题1 问题2	0.2522	0.3826	0.4783

（a）融合提示 1 的汇总指标表

方法	Top-1	Top-3	Hit
基线	0.2957	0.3652	0.5043
基于 RRF 重排序 问题1	0.2957	0.3913	0.4609
基于 RRF 重排序 倒数排名融合2	0.0783	0.1739	0.4609
倒数排名融合 问题1 问题2	0.2783	0.4087	0.4609

(b) 融合提示 2 的汇总指标表

方法	Top-1	Top-3	Hit
基线	0.2957	0.3652	0.5043
基于 RRF 重排序问题1	0.2957	0.4	0.4435
基于 RRF 重排序倒数排名融合2	0.1043	0.1913	0.4435
倒数排名融合问题1 问题2	0.2783	0.4087	0.4435

(c) 融合提示 3 的汇总指标表

A.8 鲁棒性与故障处理

我们采取了多项措施，确保迭代过程受到控制且保持一致。为维护实验完整性并避免混合部分输出：

- 检索调用在发生暂时性故障时会重试（例如超时或后端不稳定）
- 超过最大重试预算后仍失败的查询将被跳过并记录日志
- 下游阶段（重排序和融合）假定检索成功，从而防止未完整填充的列表污染指标

A.9 可复现性

在检索器参数、重排序模型和融合超参数均固定的情况下，流水线具有确定性。分块感知标识符为各阶段提供稳定的比较单位，并防止来自同一知识库文章的多个文本块所包含的证据意外合并。