RAGPerf：检索增强生成系统的端到端基准测试框架

Shaobo Li*
shaobol2@illinois.edu
伊利诺伊大学
厄巴纳, IL, USA

Yirui Zhou*
yiruiz2@illinois.edu
伊利诺伊大学
厄巴纳, IL, USA

Yuan Xu
yuanxu4@illinois.edu
伊利诺伊大学
厄巴纳, IL, USA

Kevin Chen
kpchen2@illinois.edu
伊利诺伊大学
厄巴纳, IL, USA

Daniel Waddington
Daniel.Waddington@ibm.com
IBM 研究院
USA

Swaminathan Sundararaman
swami@ibm.com
IBM 研究院
USA

Hubertus Franke
frankeh@us.ibm.com
IBM 研究院
USA

Jian Huang
jianh@illinois.edu
伊利诺伊大学
厄巴纳, IL, USA

arXiv:2603.10765v1 [cs.PF] 11 Mar 2026

摘要

我们介绍了一种基于检索增强生成的人工智能系统基准测试框架RAGPerf的设计与实现，用于刻画检索增强生成流水线的系统行为。为实现详细的性能分析和细粒度的性能剖析，RAGPerf将检索增强生成工作流解耦为多个模块：嵌入、索引、检索、重排序和生成。用户可以灵活配置各模块的核心参数，并检查这些参数对端到端查询性能和质量的影响。RAGPerf配备了工作负载生成器，通过支持多种数据集（如文本、PDF、代码和音频）、不同的检索与更新比例以及查询分布，来模拟真实场景。RAGPerf还支持不同的嵌入模型、主流向量数据库（如 LanceDB、Milvus、Qdrant、Chroma 和 Elasticsearch），以及用于内容生成的不同大语言模型。它会自动收集性能指标（即端到端查询吞吐量、主机/GPU内存占用和 CPU/GPU利用率）以及准确性指标（即上下文召回率、查询准确率和事实一致性）。我们通过一系列全面的实验展示了 RAGPerf 的功能，并在 GitHub 上开源了其代码库 [1]。我们的评估表明，RAGPerf带来的性能开销可以忽略不计。

1 引言

大语言模型（LLM）的发展彻底改变了智能数据处理，推动了大量人工智能应用的实现。然而，在用户查询需要领域专属知识或私有数据的场景中，大语言模型的能力仍然有限。为此，检索增强生成（RAG） [30, 34, 52]应运而生，用于通过外部知识库补充大语言模型。借助 RAG，我们可以将最新知识融入大语言模型的内容生成过程，为科学研究、法律证据开示和金融分析等各种特定领域提供准确的响应。

当我们为领域专属应用部署 RAG 流水线时，了解其系统瓶颈和设计权衡至关重要。然而，由于整个 RAG 流水线所涉及的计算栈十分复杂，在现实部署场景下研究其性能影响和查询质量既困难又耗时。此外，RAG 流水线通常包含多个组件（即嵌入、索引、检索、重排序和生成），这些组件之间的交互使开发人员难以确定最优配置。

因此，为应对上述挑战，开发人员需要一个可复现的、面向基于RAG的人工智能系统的端到端基准测试框架。遗憾的是，现有的检索增强生成基准测试未能实现这一目标（详见表2中的详细比较）。大多数现有的检索增强生成基准测试都专注于特定应用的语义指标 [25, 46, 53, 61, 68, 91, 104], ，很少有先前研究考察过RAG流水线的性能行为。它们无法捕捉完全集成的RAG流水线中的运行时性能和潜在资源争用。此外，其中许多缺乏灵活性，不允许用户使用不同的数据集、嵌入模型，甚至不同的向量数据库来自定义RAG流水线。

受 YCSB [14] 或 OLTP-bench [18]等云服务基准测试框架的启发，我们认为 RAG 生态系统需要专门的基准测试框架来表征 RAG 的端到端性能。本文提出了一种基于RAG的人工智能系统基准测试框架，名为 RAGPerf，用户可以借助它：(1) 通过标准接口轻松配置 RAG 流水线，并考察不同系统配置对端到端性能和查询质量的影响；(2) 通过工作负载生成器中的请求速率、查询/更新比率、批次大小等不同参数对现实场景进行建模；(3) 自动收集 RAG 流水线不同阶段的丰富性能统计数据。具体而言，RAGPerf 具备以下独特功能。

首先，RAGPerf 可通过其工作负载生成器（§3.2）模拟现实世界中知识库操作的动态行为。RAGPerf 支持多种插入、更新和删除操作，并可配置访问模式，因此能够对常见场景进行建模，例如文档集合持续增长、内容定期

*共同第一作者。[1]RAGPerf 的代码库可在 https://github.com/platformxlab/RAGPerf 获取

修订、删除过时或无效条目，以及这些行为同时发生的混合情形。这种灵活性使 RAGPerf 能够捕捉真实的查询，并研究在不同应用场景中，数据新鲜度、索引开销和端到端性能会受到怎样的影响。

其次，RAGPerf 能够准确捕捉不同流水线阶段中系统配置变化的影响（§3.3）。它允许用户更改系统中定义的核心参数，从嵌入维度到向量索引方法，从检索策略到重排序算法。它会系统地衡量这些变化对性能和查询质量的影响。借助性能分析结果，用户可以了解每种配置调整后的性能表现，并据此调整系统参数，从而在不同的工作负载和部署设置下做出数据驱动的决策。

第三，RAGPerf 旨在提供准确的测量结果，同时将性能分析开销保持在较低水平（§3.4）。它收集两组互为补充的指标：性能指标和准确性指标。系统性能指标包括对硬件资源利用率（例如 GPU/CPU 利用率、内存消耗和 I/O吞吐量）的细粒度测量，以及端到端性能指标（例如延迟和吞吐量）。准确性指标用于评估生成答案的质量，例如上下文召回率、查询准确率和事实一致性，并支持用户分析系统效率与输出质量之间的权衡。这些指标共同为理解 RAG 流水线的系统行为奠定了坚实基础，可为其未来部署提供指导。RAGPerf 包含一个可灵活适配的系统性能监视器，该监视器独立于 RAG 流水线运行，因此其性能分析不会影响常规的 RAG 行为。RAGPerf 支持系统级和组件级性能分析，同时仅引入可忽略不计的运行时开销（参见我们的评估：§5.8）。

最后，RAGPerf 被实现为一个模块化且可扩展的基准测试框架，并配备了定义明确的接口（§3.5）。RAG 流水线被分解为用于数据分别是嵌入、索引、检索、重排序和生成。为支持灵活的流水线构建，RAGPerf 仅定义每个组件的输入和输出接口，不对内部实现施加限制。每个组件都有自己的可配置设置。默认情况下，RAGPerf 支持多种嵌入模型、重排序模型、生成模型和向量数据库，如我们的评估所示（§4 和 §5）。

为反映真实世界检索增强生成系统的性能特征，RAGPerf 收集了多样化的数据集（表 3），这些数据集体现了真实世界内容的异质性，并涵盖不同类型的 RAG应用。RAGPerf 默认提供覆盖整个 RAG 流水线的一组最新模型，包括嵌入、重排序和生成（表 4）。它还支持 LanceDB、Milvus、Qdrant、Chroma 和 Elasticsearch 等常用向量数据库（表 5）。我们通过一组全面的实验展示了 RAGPerf 的能力，并表明 RAGPerf 能够以极小的应用性能负面影响，对不同 RAG应用进行端到端的性能分解。我们还展示了 RAGPerf 能够量化各种系统配置（例如可用的 CPU 核心、CPU内存和 GPU 内存）以及不同 RAG 配置（例如各种批次大小、嵌入维度和索引方案）对性能的影响。我们已在 https://github.com/platformxlab/RAGPerf 开源 RAGPerf，供公众使用。

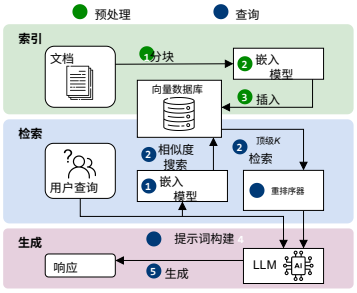


图 1：端到端 RAG 工作流。

2 背景与动机

在本节中，我们介绍基于RAG的人工智能系统的背景。随后，我们讨论构建端到端基准测试框架的动机，以考察性能和设计权衡。

2.1 基于 RAG 的人工智能系统架构

大语言模型通常基于公开数据集进行预训练和微调，因此在通用问答方面表现出色。然而，当用户查询需要领域专属知识或私有数据才能作答时，其效果会受到限制。检索增强生成（RAG）[44] 已成为一种利用外部知识库补充语言模型的方法。借助 RAG，大语言模型能够整合最新知识，在特定领域场景（例如科学、法律和特定商业领域）中提供准确的响应。

如图 1所示，典型的检索增强生成系统由三个阶段组成：索引阶段用于准备知识库，而在处理用户查询时则会进行检索和生成。接下来，我们将分别介绍这三个阶段。

索引。为了实现快速、准确的知识检索，并为大语言模型提供相关上下文，必须对外部知识库进行预处理，将原始数据转换为可搜索的表示。具体而言，首先将原始数据切分为较小的文本块（ ），以支持细粒度检索，并尽量减少后续检索和生成过程中引入的无关上下文。根据输入数据的模态和结构，以及下游应用的需求，分块可以采用不同粒度，从基于文本的分段（例如固定长度的文本片段、若干句子或段落）到模态专属单元，例如图像区域、视频片段或音频片段。随后使用嵌入模型将生成的文本块映射为固定维度的稠密向量（ ）。这些向量在高维向量空间中捕捉文本块的语义含义。在这一空间中，语义相似的文本块彼此距离更近，为基于相似度的检索奠定基础。接着将生成的向量导入向量数据库（ ），并在高维嵌入空间中构建专用索引结构，以实现高效且可扩展的搜索。索引方法分为多个类别，从基于分层图的方法（例如 HNSW）到基于分区的方案（例如 IVF）。有时还会采用标量量化（SQ）和乘积量化（PQ）等压缩技术，在保持检索准确率的同时减小索引大小并提高内存效率。

表1：流行的基于检索增强生成的应用。

类别	应用	特征
对话式 AI	搜索引擎 [7, 9, 99]	高并发用户请求，需要短响应时间。
	客户支持机器人 [22, 29, 66, 98]	
	个人助理 [19, 94]	
企业智能	法律证据开示 [3, 6, 15]	包含表格/表单的结构化数据。高精确度和准确率要求。
	财务报告分析 [10, 51, 93]	
	医疗支持 [89, 97, 103]	
多模态语义搜索	视频检索 [8, 38, 56, 60, 78]	包含大量负载的非结构化数据。
	医学影像诊断 [75, 82, 83]	
	会议摘要 [5, 43]	

检索。收到用户查询后，系统首先使用索引阶段所用的同一嵌入模型对用户输入的提示词进行编码（ ）。在实际应用中，一些 RAG 流水线还会引入轻量级辅助模型，调整搜索方向，以进一步提高检索准确率。例如，该模型可以通过拆解或消除原始查询的歧义，或生成经过优化的查询表述，来增强查询表示与检索内容之间的对应关系。随后，利用生成的查询向量表示执行相似性搜索，从向量数据库中检索最相关的 top-K 个上下文（ ）。相似性搜索通常基于近似最近邻（ANN）算法，这类算法以牺牲部分检索准确率为代价换取更高的计算效率，从而能够在大规模、高维嵌入空间中实现低延迟搜索。为了进一步提高检索精确率，一些流水线还会加入重排序步骤，对从向量数据库中检索到的候选上下文重新评估（ ）。在此步骤中，通常会使用表达能力更强的模型，该模型往往会同时处理完整查询内容及其相关上下文，对候选上下文重新排序，并过滤掉相关性较低或含噪声的结果，然后再将结果传递给下游语言模型。**生成**。最后，检索增强生成系统将原始用户查询与检索到的上下文合并为一个提示词（ ），通常遵循预定义的系统模板或用户定义的提示格式，以确定如何组织检索信息。随后将该提示词传递给语言模型，由其结合查询和提供的上下文生成最终响应（ ）。

2.2 基于检索增强生成的应用

检索增强生成已从一种实验性技术发展为不同应用领域中的关键基础设施。我们在表1中总结了常见的检索增强生成应用。检索增强生成最普遍的应用是通用对话式人工智能，例如经过搜索引擎增强的聊天机器人或实时助手。该系统能够处理网络规模的数据摄取，同时在高并发用户请求下保持短响应时间。检索增强生成流水线需要平衡高频向量更新以确保知识的时效性，同时处理并发检索请求，避免延迟增加。

检索增强生成的另一个常见用途，是支持金融、法律和医学等专业企业领域中的有据可依的问答。这类企业检索增强生成系统通常会集成一个内部知识库，其中包含复杂的非结构化数据。例如，金融分析系统会从财报和表格中提取细粒度的数值数据；法律发现工具则需要在庞大的合同库中进行高召回率检索。与开放领域的人工智能应用不同，这些应用必须处理复杂的文档布局，例如多栏 PDF 和嵌入式表格。同时，这些应用通常还面临严格的隐私要求，并须达到高精确度标准。

表 2：现有的检索增强生成系统基准测试。

基准测试	可配置流水线	End-到端	系统指标	语义指标
BEIR [85]				
RAGBench [25]				
TruthfulQA [53]				
RAGTruth [61]				
FELM [104]				
VectorDBBench [105]				
VESPA [67]				
LegalBench [68]				
OmniEval [91]				
AutoRAG [46]				
RAGPerf				

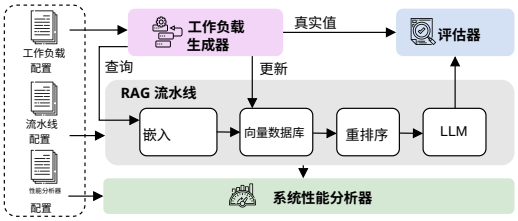
随着多媒体内容的增长，RAG 系统也在迅速发展，以支持超越简单文本的跨模态交互。典型示例包括视觉知识检索、视频语义搜索（例如，从视频中检索相关片段）以及音频分析（例如，会议摘要）。它们需要专门的数据处理，例如光学字符识别（光学字符识别） [36, 76, 81] 或自动语音识别（ASR） [73, 84]。RAG 流水线采用统一的嵌入策略，将各类文件类型映射到共同的向量空间。

这些多样化的RAG应用对底层硬件有截然不同的资源需求，并可能遇到不同的瓶颈，例如内存容量、I/O 带宽和计算能力。

2.3 RAG 系统的基准测试

在 RAG 流水线的各个组件之间分配资源涉及众多配置选项。理解这些决策对特定应用的性能影响十分重要，但也颇具挑战，因为多个组件与参数之间的交互使开发者难以选择最优的系统配置。此外，集成零散软件组件（例如数据分块、索引、向量数据库和大语言模型推理）的复杂性给开发者带来了额外负担。因此，如果没有端到端评估框架，在真实部署场景下研究性能影响将既困难又耗时。

遗憾的是，正如表 2 所示，现有检索增强生成基准测试无法全面呈现系统行为。当前大多数检索增强生成基准测试框架优先关注检索增强生成应用的语义指标，包括检索精确率、幻觉率和生成忠实度。尽管这些指标对于确保输出质量至关重要，但它们几乎无法反映生产部署中的系统效率（例如延迟、吞吐量和硬件利用率）。类似地，AutoRAG 等框架 [46] 仅依据响应质量优化流水线配置。其他基准测试，如 vectordbbench [105] 和 vespa [67]则只关注单个组件，并且仅评估



Fi图 2：RAGPerf 框架的架构 。

向量数据库或大语言模型生成速度。它们无法捕捉完全集成的检索增强生成流水线中的运行时性能干扰和资源争用。此外，现有基准测试通常缺乏可配置性。例如，LegalBench-RAG [68] 和 OmniEval [91] 预先定义了语料库，使用户难以自定义检索增强生成流水线。基于上述原因，我们希望开发一个统一的端到端基准测试框架，以便于检索增强生成流水线及应用的开发与评估。

3 设计与实现

3.1 RAGPerf 的目标

我们开发 RAGPerf，旨在实现以下目标。

- **端到端评估：** RAGPerf 应衡量整个 RAG 流水线的性能表现和任务质量。与逐组件基准测试不同，RAGPerf 应能够识别系统级性能瓶颈。
- **模块化与易用性：** 为适应检索增强生成技术的快速演进，RAGPerf 应提供完全模块化且可配置的架构。它应允许用户以最少的工程投入，无缝集成现有或自定义的检索器、重排序器和生成器。
- **低开销性能分析：** RAGPerf 应提供轻量级的性能分析方法，以收集细粒度的系统指标，例如内存占用、GPU/CPU 利用率、I/O 带宽等。
- **工作负载多样性：** RAGPerf 应支持数据集、查询类型、输入长度和到达模式的真实组合。多样且可配置的工作负载可在不同运行条件下对系统的不同方面施加压力。
- **可扩展性与可移植性：** RAGPerf 应能适配大多数平台，无论部署在单个 GPU 还是多个 GPU 上。该框架还应支持部署实验配置，并能在不同硬件配置下轻松运行实验。

RAGPerf 旨在反映真实世界的部署情形，同时让用户能够轻松配置和使用，如图所示2。下面介绍 RAGPerf 的设计与实现。

3.2 工作负载生成器

真实世界中的检索增强生成系统通常运行于知识库持续更新的环境中。与传统数据库中的索引不同，RAG 系统中的向量索引

更加复杂且成本更高。首先，每次插入都需要使用嵌入模型将原始数据转换为向量，这一过程计算成本高昂。其次，支持更新和删除的成本也很高，因为许多广泛使用的索引结构（例如 HNSW [59] 和 IVF [80]）不支持增量更新

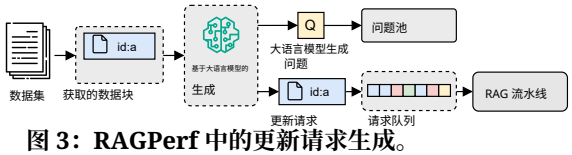


图 3：RAGPerf 中的更新请求生成。

而是需要进行成本高昂的索引重建。为降低重建开销，许多向量数据库会使用一个小型的临时平坦索引来接收新插入的数据和更新，以牺牲更高的查询延迟为代价，减少重建频率。随着这个辅助索引不断增大，索引效率会下降，导致查询延迟升高，即使数据集总规模保持不变也是如此。因此，仅在静态数据集上评估检索性能的基准测试，无法反映现实世界的 RAG 部署中持续更新数据所造成的性能下降。

为模拟真实场景，RAGPerf 会生成并发读取和写入请求，以对目标检索增强生成流水线进行压力测试。RAGPerf 中的工作负载由以下四种基本操作组成：

- 查询：从预定义的查询池中提出一个用户问题，以测试数据检索和生成。
- 插入：导入全新的数据块，要求向量数据库对新的嵌入进行编码和索引。
- 更新：使用新的事实数据修改现有文档，以测试知识库反映变化的能力。• 移除：删除具有特定文件 ID 的文件。

工作负载的整体组成由配置决定，其中为每种操作设定了发生概率。

请求除了操作组合之外，RAGPerf 还确定访问分布，用于控制如何选择要更新和删除的目标文件 ID，以及要查询哪些文档。RAGPerf 支持两种主要的访问模式，以模拟不同的访问行为。均匀分布确保所有文件被访问或更新的概率相等。齐夫分布模拟“热点”现象，即少量文件接收绝大多数更新和查询。

通过调整这些参数，用户可以使工作负载呈现特定部署场景的特征。例如，要构建一个基于维基百科数据集的读取密集型工作负载，我们可以生成一个包含 90% 查询和 10% 更新，并采用齐夫访问分布的工作负载。该工作负载模拟现实世界中的维基行为：少量热门维基页面接收绝大多数事实更新和查询，而大多数文章保持静态。

请求生成。根据分布选定操作和目标文件后，RAGPerf 构造请求负载。对于插入、移除和查询操作，所需数据直接从源数据集或问题池中获取。然而，更新操作需要更复杂的逻辑来确保评估有效性。更新的动态真实值生成。为了支持对更新操作下的检索增强生成流水线进行准确性和性能评估，RAGPerf 需要将带版本的数据块作为更新负载，并准备相应的问题和真实答案，以测试更新后的知识库。由于大多数公开数据集不支持版本控制，RAGPerf 使用基于大语言模型的生成模块来生成合成事实更新及其对应的真实问答对。

给定用户定义的更新分布，RAGPerf 会抽取一部分文档进行更新。对于每份文档，RAGPerf 会随机选择一个文本块，并修改其中的名词短语或数值。这样便可通过一个简单的问题轻松测试更新内容。修改由 DistilBERT [79] 模型完成：模型会遮蔽文本块中的目标词，并在遮蔽位置生成替换词。随后，更新后的文本块会被打包为请求负载，并在基准测试期间发送至 RAG 流水线。接着，生成器使用问题生成模型 T5 [74] 读取更新后的数据块，并提出一个答案为替换词的问题。这些问题会被打乱后加入问题池，以验证检索增强生成系统检索到的是更新后的事实，而非过时数据。

图 3 展示了更新请求的生成过程。工作负载生成器根据用户定义的访问分布，从所有文件 ID 中选取一组待更新的目标文件 ID。随后，RAGPerf 从数据集中获取相应的数据块（图 $id:\ a$ 中的文本块 3）。所有获取的数据都会传递给基于大语言模型的生成模块，以生成更新后的数据块和测试问题。最后，RAGPerf 将修改后的数据块作为负载，构建更新请求，并在基准测试运行期间将其发送至 RAG 流水线。测试问题会存储在问题池中，可在基准测试运行期间选作用户查询。

3.3 可配置的 RAG 流水线

典型的检索增强生成系统可能提供数十到数百种配置选项和参数。例如，用户可以配置不同的向量数据库后端，并通过选择其他索引方法、量化方案和相似度指标，进一步微调其行为。此外，批次大小、检索深度和更新策略等系统级参数会显著影响性能和资源消耗。这些选择共同构成了一个庞大而复杂、难以探索的优化空间。我们不将 RAG 流水线视为一个整体的黑盒，而是将其分解为一组模块，并通过外部 YAML 配置定义这些模块的行为。在运行时，RAGPerf 会收集细粒度的准确率和性能指标，以量化不同配置对性能和资源的影响。

3.3.1 **嵌入**。 嵌入阶段的性能和效率取决于两个关键因素：原始数据的分块方式，以及嵌入模型的配置方式。

分块。嵌入模型通常受严格的序列长度限制，并假定每个基本文本块单元都代表一段语义连贯的信息。因此，分块是嵌入前数据预处理的关键环节。

对于基于文本的文档，定长分块会将文档划分为大小均匀的片段。这种方法简单易行，内存使用量可预测，但可能会打破语义边界并降低检索相关性。基于分隔符的分块利用文本片段（例如句子或段落）来提升语义连贯性，但会导致嵌入批次形状不规则，且访问延迟的波动更大。基于语义的分块则显式建模内容含义，以确定连贯的边界。它通常采用自然语言处理方法，或直接使用小型语言模型来保留语义连贯的上下文并减少语义碎片化，从而进一步提高检索准确率和下游生成质量，但代价是产生更大的计算开销。

文本块之间的重叠也会影响数据检索的准确率。在许多情况下，一段文本在语境上可能同时与前后句子相关，因此，引入重叠有助于保留相邻文本块之间共有的语境。

文档格式。检索增强生成系统使用的许多知识源主要包含文本信息，但以面向演示的格式存储（例如 PDF、Microsoft Word 和 HTML）。这给嵌入流程带来了额外的权衡。一种方案是仅提取文档的文本内容。这种方法灵活性高、计算开销低，并且能够重复使用标准文本嵌入模型，但可能会丢失结构信息或布局信息，例如表格、图表和格式。另一种方案是使用基于 OCR 的解决方案，它直接处理渲染后的文档布局，能够更可靠地恢复丰富的结构信息。然而，这些方法会带来显著更高的计算开销和预处理延迟。RAGPerf 支持这两种数据分块方法。

嵌入模型。数据分块后，我们会将每个文本块编码为一个向量，使其能够在高维空间中捕捉该文本块的语义含义。选择合适的嵌入模型是一项关键的设计决策。每个模型都可通过若干关键属性来描述，包括参数数量、输出嵌入维度和计算成本。尽管较大的模型通常具有更强的表征能力，但这通常会以更高的推理延迟和更多的资源消耗为代价。嵌入效果在很大程度上取决于模型的训练方式（包括训练数据集和训练目标）。主要基于通用对话语料或自然语言语料库训练的模型，在应用于源代码等领域专用输入时，表现可能不佳。因此，为了获得良好的检索质量，通常需要针对特定领域定制专用嵌入模型。

尽管许多嵌入模型的内存占用相对较小，但将它们与生成模型部署在同一 GPU 上，会减少生成模型可用的键值（KV）缓存内存，从而可能降低生成吞吐量并增加延迟。在同一设备上并发执行嵌入和生成工作负载，还会进一步造成资源争用。另一种做法是将嵌入计算卸载到主机 CPU，以减轻 GPU 内存压力，但由于 CPU 的计算能力有限，嵌入延迟会显著增加。尽管资源争用有所减少，这仍可能损害端到端响应时间。这些权衡表明，谨慎进行资源分配十分必要。RAGPerf 允许用户配置硬件资源分配（即 CPU/GPU 和主机/GPU 内存），并考察其对检索增强生成流水线性能的影响。

多模态嵌入。无需明确区分分块和嵌入，专用的多媒体或多模态嵌入模型可以直接处理格式化输入。这些解决方案提供了一种模块化且易于集成的黑箱方法，但限制了分别调整分块或嵌入行为的灵活性。RAGPerf 支持用户采用这种方法，并提供 ColPali [23] 和 Clip [72]等多模态嵌入模型。

嵌入配置与性能分析。我们通过模块化且可扩展的预处理与嵌入栈，支持多种输入模态，并采用 `BaseEmbedder` 抽象（图 4）。对于基于文本的文档，该框架集成了常用的开源光学字符识别流水线，并将其抽象为统一模块，以便于采用。在嵌入阶段，RAGPerf 通过 HuggingFace 开发的 `SentenceTransformer` 接口，支持灵活集成嵌入模型，使用户能够评估不同的模型架构和嵌入维度。RAGPerf 还为音频、图像和视频数据提供预构建流水线，支持模态到文本的转换和直接多模态嵌入，确保对各种输入格式进行一致的评估。

RAGPerf 还会在此阶段收集辅助元数据，以支持性能分析。在文本分块期间，RAGPerf 会记录每个文本块的起始和结束偏移量，以及对原始文档的引用；这只会带来较低的存储开销，同时还能准确追踪分块行为。这有助于考察每个输入批次中文本块长度差异对性能的影响。在嵌入期间，RAGPerf 通过测量摄取吞吐量以及批量编码期间的 GPU 内存利用率，报告数据准备开销。RAGPerf 还会量化采用更高维嵌入模型所产生的存储和内存成本。

3.3.2 **向量数据库与索引。** 作为检索增强生成流水线的骨干，向量数据库的检索性能取决于三个关键因素，这些因素共同影响索引成本、插入延迟、查询延迟和查询质量。

索引方法。基于图的索引（例如 HNSW [59]）搜索速度快，但需要较大的内存容量和较长的构建时间。基于聚类的索引（例如 IVF [80]）可以减少内存占用，但需要谨慎调整分区数量（*nlist*），以平衡搜索速度与训练时间。各项参数（例如最大图度（M）以及构建过程中探索的候选集大小（*ef_construction*））让用户能够在索引构建时间与搜索性能和准确率之间进行权衡。为进一步减少内存消耗，通常会使用向量量化来压缩

嵌入表示。常见技术包括标量量化（SQ），例如将 FP32 向量转换为 INT8，以及乘积量化（PQ），即通过聚类压缩向量。采用激进的量化策略可以大幅降低内存使用量并提高缓存效率，但不可避免地会引入近似误差，从而降低检索准确率。

向量数据库一直在提供 GPU 加速索引，以利用大规模并行处理，实现快速索引构建和高吞吐量检索。然而，由于检索增强生成流水线通常也依赖相同的 GPU 进行 LLM 生成，向量数据库可能会与生成模型争用 GPU 资源。

向量数据库更新。为支持实时更新，同时避免对完整向量索引进行代价高昂的重建，检索增强生成系统通常采用混合索引方法。该方法会维护一个临时平坦索引，用于缓存新收到的更新，并与原有的近似索引并存。虽然这能确保数据立即保持新鲜，但也带来了查询延迟与重建开销之间的权衡。向量数据库配置与性能分析。不同供应商提供的 API 和部署方式各不相同（例如 Milvus、Lance 和 Qdrant），使向量数据库生态高度碎片化。为了在单一统一框架中支持多个向量数据库，

```python
class DBInstance()
    // Declare a DB instance with its type and storage location
    def __init__(db_type, db_path, client)
    // Build indexing of given type on a collection
    def build_index(index_type, metric_type, collection_name)
    // Insert a batch of chunks into a collection
    def insert(vectors, chunks, collection_name)
    // Retrieve similar chunks given a batch of chunks using ANN
    def search(vectors, collection_name)

class BaseEmbedder()
    // Declare an embedding model using model name and resource constraint
    def __init__(model_name, device, mem_util)
    // Embed a set of inputs into vector representations
    def embedding(input_docs)

class BaseReranker()
    // Declare a reranker using model name and resource constraint
    def __init__(model_name)
    // Rerank and get the top-k results given the query and input documents
    def rerank(self, query, candidate_docs, topk)

class BaseLLM()
    // Declare an LLM for generation using model name and resource constraint
    def __init__(model_name, device, mem_util)
    // Generate final answers given a set of prompts and context
    def generate(prompts, candidate_docs)
```

图 4：用于支持不同 RAG 流水线模块的应用程序接口。

RAGPerf 开发了一种`DBInstance`抽象（图4），定义了一组精简的标准操作。该抽象将 RAGPerf 与数据库专用 API 解耦，使新的数据库后端能够集成到 RAG 流水线中，并通过一个轻量级适配器层将标准接口映射到原生数据库操作。

RAGPerf 跟踪向量数据库中核心操作的执行时间，包括插入时间、索引构建时间和查询延迟。如果启用了混合索引，RAGPerf 将报告每个索引的延迟。它还会监测所构建索引结构的存储占用，并与原始向量数据进行比较。RAGPerf 以紧凑的二进制格式记录每个请求的检索结果，用于测量查询准确率。通过仅存储检索到的块标识符，RAGPerf 保留检索上下文以供准确性指标使用，例如召回率（即它成功获取回答查询所需内容的频率）。

3.3.3 **重排序。** 重排序在 RAG 系统中充当中间优化步骤，旨在通过重新排序和选择性过滤来提高检索上下文的质量。

重排序模型。选择合适的重排序模型至关重要。双编码器 [17] 重排序器能够提供低延迟评分和高吞吐量，适用于对延迟敏感的大规模部署。相比之下，交叉编码器会联合处理查询和文档，以实现更高的排序准确率，但会产生显著更高的计算成本。基于大语言模型的重排序器可以通过更丰富的语义推理进一步提升重排序质量，但代价是显著增加延迟和资源消耗。检索深度。重排序器的效果会受到初始检索深度的显著影响。增加检索候选项的数量可以提高纳入相关内容的概率，但也会增加重排序的计算成本。此外，所选的输出深度决定了传递给生成模型的上下文量，直接影响生成延迟和质量。重排序的配置与性能分析。与数据库抽象类似，我们开发了一个`BaseReranker`抽象（图 4），用于标准化各种重排序策略的接口——从经过优化的交叉编码器（例如 ms-marco-MiniLM [92], BGE [96], Cohere [13]）到大语言模型排序器（例如 RankLLaMA [57], LLM4Rerank [27]）。RAGPerf 记录重排序后的上下文，通过将准确性指标与检索结果进行比较来评估其影响。为减少存储开销，系统仅跟踪块 ID，而不是完整的文本块内容。

*3.3.4*生成。 生成步骤通过大语言模型推理，利用用户提示和检索上下文生成最终响应。模型选择。大语言模型的选择会显著影响检索增强生成系统的推理能力和生成行为。较大的大语言模型能够提供更高的响应质量，而较小的模型则可以实现更低的延迟和更高的吞吐量。此外，还可以使用特定任务模型（例如视觉模型）来理解包含图像、表格或结构化文档的输入上下文。RAGPerf 使用 vLLM [49], 这一开源高吞吐量推理引擎作为默认的大语言模型后端。它通过统一接口支持广泛的大语言模型架构，使 RAGPerf 能够灵活更换模型，而无需对流水线的其余部分进行额外修改。

并行策略。 RAGPerf 支持多种并行策略。数据并行会在多个设备上复制模型，并将传入请求分发给各个副本，从而提高吞吐量，同时保持每个请求的延迟不变。张量并行会将模型的各个层拆分到多个设备上，以缩短每层的计算时间，或使大型模型能够装入设备内存。不过，这种方法需要较高的 GPU 间带宽（例如使用 NVLink [63]）。相比之下，流水线并行通过将模型划分为在不同设备上执行的多个阶段，使大型模型能够运行，但会引入流水线气泡，也可能出现负载不均衡。生成配置与性能分析。RAGPerf 使用 vLLM 作为大语言模型的后端服务框架。它通过查询 vLLM 提供的内置指标端点，获取详细的性能指标，包括首 token 时间（TTFT）、每个输出 t oken 的时间（TPOT）和 KV 缓存利用率。用户可以通过 TTFT 分析请求排队和调度延迟，通过 TPOT 分析解码效率，并通过 KV 缓存利用率了解内存压力。

3.4 性能与质量指标

性能指标。 RAGPerf 集成了资源监控器，以帮助分析系统资源使用情况。该监控器通过 Linux procfs 采集主机端中央处理器、内存和输入/输出统计信息，并通过 NVIDIA 管理库（NVML）中的 GPU 性能监控（GPM）接口采集 GPU 指标 [62]。

所有指标均按可配置的间隔进行采样，并记录为时间序列跟踪数据，以捕获所有流水线组件的资源利用率。为尽量减少对检索增强生成工作负载的干扰，我们将监控子系统作为解耦的低优先级后台守护进程运行。性能分析数据在执行期间持久化到单独的存储设备，监控器会跟踪从每个探针采集指标所需的时间，并自动调整采样周期。性能分析器支持在崩溃时优雅关闭，确保所有缓冲数据在终止时被安全刷新并持久化。质量指标。RAGPerf 通过与 *Ragas* [21]框架集成，支持生成质量评估。它是一个开源框架，利用大语言模型作为评判者，依据多项预定义指标对检索增强生成质量进行评分。RAGPerf 提供了一个封装器，可通过 vLLM 引擎在本地运行 Ragas，并传入收集到的流水线跟踪数据（包括检索条目、生成的响应和真实答案）以供评估。为避免干扰检索增强生成工作负载，评估在工作负载执行之后进行。

RAGPerf 采用三个标准指标来评估检索增强生成的生成质量。上下文召回率通过计算所有检索到的文本块中相关文本块所占的比例，评估检索效果

表 3：RAGPerf 中使用的示例数据集。

数据集	Type	Size	条目
维基百科 [24]	Text	19.3 GB	6.41M
arXiv [42]	PDF	48 GB	30K
GitHub 代码 [12]	Code	32 GB	11M
The People's Speech [26]	音频	35.5 GB	0.3M

，这些文本块来自向量数据库或重排序之后。事实一致性评估生成响应中的声明有多少比例得到了检索上下文的明确支持。数值越高，表示大语言模型越依赖检索到的文档来生成响应，反之亦然。查询准确率衡量生成响应与真实答案之间的语义对齐程度。

3.5 RAGPerf 实现

RAGPerf 框架使用 Python 实现，并利用现有的开源框架部署模块，具体如下。RAGPerf 可直接与 HuggingFace 生态系统集成，从而支持按名称部署嵌入模型、语言模型和数据集。对于向量数据库，RAGPerf 目前支持 LanceDB [50], 、Milvus [90], 、Qdrant [69], 、Chroma [11], 和 Elasticsearch [20], ，并使用这些数据库公开提供的开源应用程序接口。RAGPerf 使用 vLLM 作为大语言模型服务后端。

不同的 RAG 流水线组件可能需要不同的 Python 模块依赖要求。为确保基准测试环境可复现且稳定，RAGPerf 使用 CMake 和 pip-tools 处理依赖解析。构建系统会根据流水线配置自动解析依赖项，确保执行环境一致且易于部署。RAGPerf 还支持将需要不兼容依赖项的组件隔离在不同的 Python 虚拟环境或容器中。

RAGPerf 的监控子系统以多线程守护进程的形式实现，定期从 Linux proc 文件系统、cgroups v2 和 NVIDIA GPU 监控库中采集系统及组件级资源统计数据。主机级系统范围指标从 /proc/ 收集，而逐组件统计数据则从 /proc/<pid>/ 获取。进行 cgroup 级探测时，监控器从 /sys/fs/cgroup/<cg>/ 读取数据。GPU 指标通过 NVML GPM 接口收集。我们使用 protobuf 30.2 存储输出统计数据。

4 基准测试工作负载

RAGPerf 默认定义了一组具有代表性的检索增强生成工作负载。我们利用这些标准化工作流对系统进行压力测试，重点考察数据模态、模型规模和检索逻辑的变化。我们将在本节中详细介绍这些工作负载所用的数据集、模型和向量数据库配置。

4.1 数据集

RAGPerf 使用各类真实世界数据集，对实践中 RAG 系统的知识源进行建模。如表3所总结，我们的选择涵盖不同的模态和结构复杂性。对于通用信息检索系统，我们以维基百科作为检索语料库，并搭配 Natural Questions [48] 作为查询。为应对特定领域的挑战，我们分别纳入 arXiv 研究论文和 GitHub 代码仓库，并使用机器学习问答和编程助手查询。此外，我们将评估扩展到多模态场景，通过

表4：用于生成、嵌入和重排序的模型。

生成模型	任务
Qwen2.5-7B-Instruct [71]	通用
gpt-oss-20b [64]	通用
DeepSeek-R1-Distill-Qwen-32B [16]	通用
Llama-3.3-70B-Instruct [86]	通用
Qwen2.5-72B-Instruct	通用
Qwen2.5-VL-3B-Instruct [1]	视觉
Qwen2.5-VL-7B-Instruct	视觉
Qwen2.5-VL-32B-Instruct	视觉
Qwen2.5-Coder-7B-Instruct [35]	编程
Qwen2.5-Math-7B-Instruct [100]	Math
嵌入模型	**维度**
all-MiniLM-L6-v2 [77]	384
all-mpnet-base-v2 [77]	768
gte-large-en-v1.5 [102]	1024
colpali-v1.2 [23]	1030 × 128
clip-ViT-B-32 [72]	512
重排序模型	**最大上下文长度**
ms-marco-MiniLM [92]	512 词元
BGE [96]	8192 词元

纳入 arXiv 页面图像和 The People's Speech，以支持视觉与音频检索增强生成系统。为模拟从资源受限的本地设备到大规模服务器的部署场景，我们确保所有数据集均支持可定制的规模扩展。

4.2 模型

我们的基准测试支持覆盖整个检索增强生成流水线的全面模型集，涵盖嵌入、重排序和生成。在 RAGPerf 中，我们将模型能力与 §4.1 中的特定数据集相匹配，以进行基准测试。对于文本生成，我们采用通义千问-2.5、Llama-3.3 和 DeepSeek-R1 等模型，用于通用知识检索和推理。针对特定领域的挑战，我们集成了专用模型，包括用于编程任务的 Qwen2.5-Coder、用于数学的 Qwen2.5-Math，以及用于视觉文档理解的 Qwen2.5-VL。为支持检索阶段的各类输入，我们使用多种嵌入模型，从 all-MiniLM 和 gte-large 等标准稠密检索器，到用于视觉 PDF 处理的多模态 ColPali 模型。我们还通过引入 BGE 和 ms-marco-MiniLM 等重排序模型来进一步提升检索精确率，以适应不同的上下文需求。在所有阶段，我们都纳入了多种规模的生成模型和嵌入模型，以评估不同部署环境下计算效率与输出质量之间的权衡。表 4 列出了支持的模型及其目标任务。

4.3 向量数据库

RAGPerf 支持 LanceDB、Milvus、Qdrant、Chroma 和 Elastic 搜索作为向量存储后端，并支持多种索引方法，包括 HNSW、IVF 和 DiskANN。具体支持情况汇总于表 5。在这些系统中，LanceDB、Milvus 和 Qdrant 支持基于 GPU 的索引构建，而 Milvus 和 Qdrant 还支持 GPU 加速查询。

4.4 多模态检索

为支持不同模态，我们默认分别为视觉数据和音频数据实现独立的流水线。

视觉文档（PDF/图像）。RAGPerf 提供两种截然不同的视觉文档处理策略，以应对文本提取准确率与视觉语义保留之间的权衡

表 5：RAGPerf 支持的向量数据库。

向量数据库	索引类型	设备支持
LanceDB [65]	IVF、HNSW、IVF-HNSW [2]	CPU/GPU
Milvus [90]	HNSW, IVF、ScaNN [31], DiskANN [37]	CPU/GPU
Qdrant [69]	HNSW	CPU/GPU
Chroma [11]	HNSW	CPU
Elasticsearch [20]	HNSW，平坦索引	CPU

。第一种方法是基于光学字符识别的提取，我们使用 Docling [55] 执行光学字符识别。该光学字符识别模型会提取文本信息，或生成视觉元素的文本描述，并将其输入基于文本的嵌入模型。或者，我们也支持视觉嵌入，直接处理整个文档页面，从而无需进行光学字符识别。我们利用 ColPali [23] 为每个页面生成多向量嵌入，并在检索过程中采用 ColBERT 风格的 [45]重排序器，将查询词直接与视觉页面区块进行匹配。最后，将检索到的视觉页面输入视觉语言模型，以生成响应。

音频。对于基于音频的检索，我们实现了一个 ASR 工作流。我们使用 Whisper [73] 模型将音频文档转录为文本。转录完成后，所得文本会使用标准文本 RAG 流水线进行处理，从而有效检索语音内容中包含的信息。

5 评估

我们的评估表明，RAGPerf 能够：（1）准确分析 RAG应用的端到端性能（§5.2）；（2）通过细粒度的硬件资源分析，识别性能瓶颈（§5.3）；（3）评估不同流水线的 RAG 准确率（§5.4）；（4）揭示更新操作对性能的影响（§5.5）；（5）量化各种系统资源配置对 RAG 性能的影响（§5.6）；以及（6）开展详细的敏感性研究，以隔离不同 RAG 参数变化的影响（§5.7）。此外，我们还表明 RAGPerf 仅产生极低的开销（§5.8）。

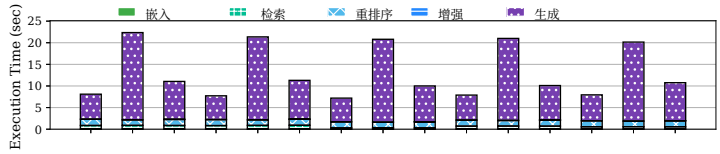
5.1 实验设置

测试平台。我们的测试服务器配备两颗 32 核 AMD EPYC 9334 中央处理器、1.4TB DDR5 主机内存，以及两块配备 94GB 设备内存的 NVIDIA H100NVL GPU。在 §5.2和 §5.3中，实验仅使用单块 GPU。我们使用 Ubuntu 22.04、Linux 内核 5.15.0 和 PyTorch 2.8。0+cu128。所有数据集均存储在数块 2TB Samsung 990 Pro SSD 中。

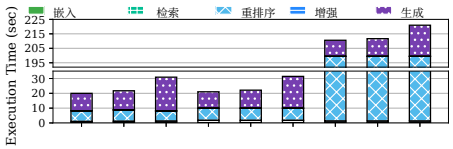
数据集。文本流水线使用 维基百科 数据集。该数据集包含 641 万篇文档/文章，原始文本总计 19.3 GB。对于 PDF/图像流水线，我们选用 48GB 的 *arXiv* 数据集，其中包含 3 万份 PDF 文档。我们使用 *PeoplesSpeech* 数据集评估音频检索，该数据集包含 35.5GB 音频数据。

5.2 端到端性能

图 5 对比了文本检索增强生成流水线和 PDF/图像检索增强生成流水线的预处理与查询阶段延迟。查询。图 5a展示了批次大小为 64 时文本流水线的延迟细分情况。使用通义千问 Qwen7B、GPT20B 和 Qwen72B 时，生成阶段分别占总延迟的 75%、80% 和 91%。因此，向量数据库的选择对端到端延迟的性能影响很小。

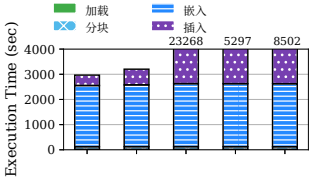


（a）文本流水线。

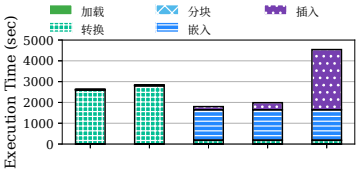


（b）PDF/图像流水线。

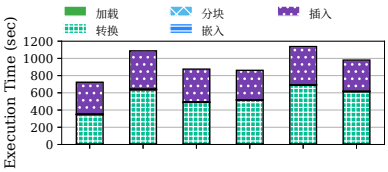
图5：一批 RAG 应用请求的执行时间分解。



（a）文本流水线。



（b）PDF/图像流水线。



(c) 音频流水线。

图6：RAG应用中数据索引阶段的执行时间细分。

图 5b展示了 PDF/图像流水线的结果。该流水线基于 ColPali 的算法构建，代表了多种 PDF/图像检索增强生成流水线。在该流水线中，每次重排序操作都需要检索完整的源文档，这平均会产生 90 次对向量数据库的查找。因此，重排序的成本很高，占总迭代时间的 28% 至 87%。由于 Chroma 对高并发查找的支持欠佳，重排序阶段在 Chroma 中占据了总延迟的主要部分。索引。图 6 详细展示了索引阶段的延迟细分情况。在文本流水线中（图 6a），嵌入阶段的耗时相对稳定，平均需要 2450 秒。相比之下，不同向量数据库的插入阶段延迟差异显著，其中一些数据库（即 Chroma、Qdrant 和 Elasticsearch）的插入和索引构建时间较长。具体而言，Chroma 在插入过程中遭遇了严重的可扩展性瓶颈，使总执行时间增加了 7。8× 与 LanceDB 相比。

对于使用 OCR工具（例如 EasyOCR [36] 和 RapidOCR [76]）的 PDF/图像流水线，格式转换占据索引阶段的大部分执行时间，平均占总时长的 98.2%。尽管 OCR 期间的峰值 GPU利用率可能很高，但平均利用率仍然较低（约为 10%），因此即使同时部署多个实例，吞吐量也不理想。使用 ColPali 模型时，我们完全绕过文本提取，将成本转移到嵌入阶段，由 ColPali 直接对文档页面图像进行编码。在向量数据库方面，Chroma 的插入吞吐量仍然较低，反映出其存在与查询处理期间观察到的相同的可扩展性限制。

最后，音频流水线（图6c）主要受转换和数据库插入成本的影响。转换时间会随着模型复杂度的增加而延长。Whisper-turbo [73]转录约需 612 秒，1.77× Whisper-tiny 所需的时间为 [73]（347 秒）。这一增长是由于计算成本更高。此外，数据库插入仍然是显著的开销，最高可占总索引时间的 51%。

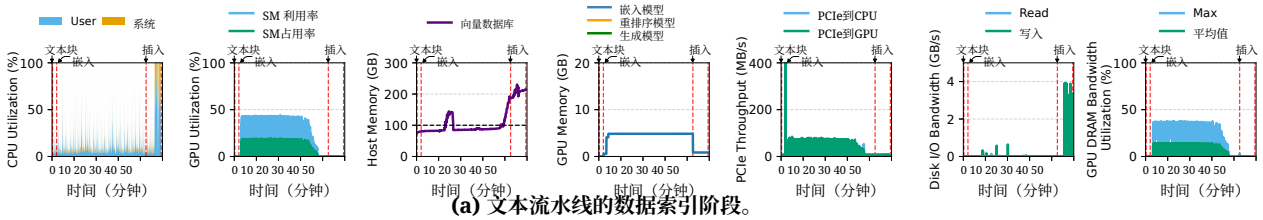
- RAGPerf 对不同模态（文本、PDF、音频）的性能进行了详细分析，揭示了在各种模型和数据库配置下，每条流水线的不同瓶颈。
- 对于文本查询流水线，LLM 生成是主要瓶颈，因此就其对端到端性能的影响而言，向量数据库的选择并不那么重要。
- 对于多模态索引，格式转换对整体延迟有显著影响。转换方法的选择会显著影响索引效率。

5.3 资源利用率细分

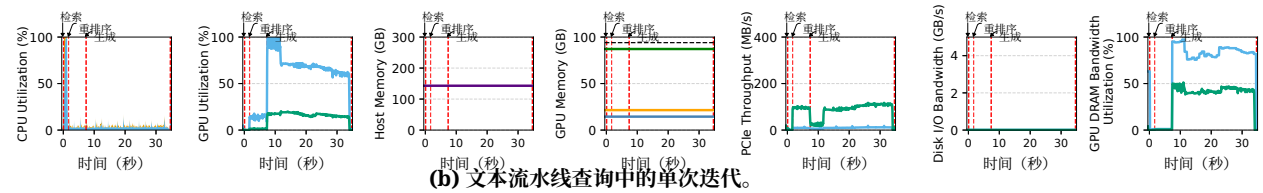
RAGPerf 的一项关键能力是对系统资源利用率进行细粒度性能分析。使用 §5.2 中相同的工作负载，图 7展示了索引和查询阶段的资源利用率细分。

计算利用率。 如图 7 所示，大多数阶段的 CPU 利用率都较低（<5%），而 GPU 的 SM 利用率在嵌入和生成阶段占主导，证实了核心 RAG 逻辑受推理限制。此外，我们观察到 SM 利用率较高（平均为 60%–70%），但 SM 占用率相对较低（< 15%），这表明活跃 SM 的数量较多，尽管相对于硬件上限而言，活跃线程束的数量仍然较少。这是因为生成阶段受内存带宽限制，而非受计算能力限制；在此期间观测到的 GPU DRAM 带宽利用率峰值也印证了这一点。相比之下，CPU 仅在特定的非推理操作中发挥关键作用。由于构建向量索引的开销很大，插入阶段的 CPU 使用率显著上升。检索阶段会使 CPU 利用率达到饱和，约为 100%（图 7b）。此外，在 PDF 重排序阶段（图 7c），由于需要为所有检索到的文本块密集计算重排序分数，CPU 利用率会飙升至 75%。

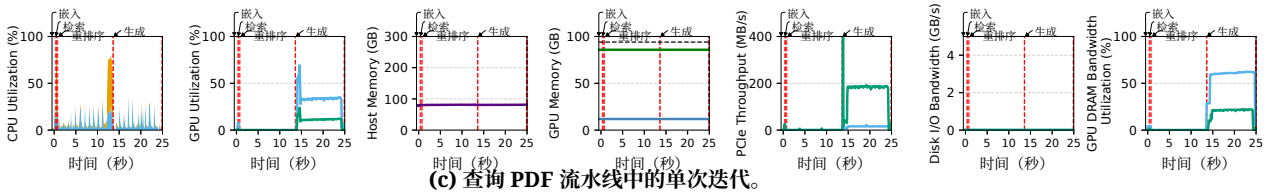
内存利用率。 在整个查询流水线中，GPU 内存使用量始终稳定在约 90GB，主要由嵌入模型、重排序模型、大语言模型权重和键值缓存占用。这种情况源于现代服务后端（如 vLLM）采用的静态内存分配策略，即使模型处于非活动状态，也会将其常驻内存。虽然动态模型卸载可以回收未使用的内存，但会因重新加载模型而引入显著的延迟



(a) 文本流水线的数据索引阶段。



(b) 文本流水线查询中的单次迭代。



(c) 查询 PDF 流水线中的单次迭代。

图 7：不同RAG应用的资源利用率分解。

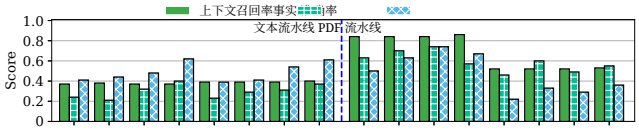


图8：文本与 PDF 流水线的准确率得分。

。此外，主机内存主要被向量数据库索引占用，在向量检索和数据处理期间成为主要瓶颈。在数据索引阶段，插入会触发主机内存消耗大幅增加：使用 20 GB 的维基百科数据集时，内存占用从约 90 GB 上升至超过 220 GB。这表明，更新性能从根本上受到可用主机内存的限制。

输入/输出流量。 在 RAG 流水线的大多数阶段，磁盘输入/输出相对空闲。然而，数据索引阶段构成了主要的输入/输出瓶颈。如图 7a所示，插入阶段需要 4 GB/s 的峰值磁盘写入吞吐量，以将生成的嵌入和向量索引持久化到存储中。

- RAGPerf 集成了一个资源监控器，能够捕获每个单独流水线阶段的细粒度硬件利用率。
- 瓶颈会在不同的 RAG 阶段于不同硬件资源之间转移：嵌入和生成阶段在 GPU 上，检索和索引构建阶段在中央处理器上，数据处理阶段在主机内存上，从而为系统优化创造了机会，例如重叠流水线执行。
- 如果流水线中的所有模型在执行期间都需要将权重驻留在 GPU 内存中，那么当模型处于非活动状态时，静态资源分配会导致内存利用不足。因此，需要在各流水线阶段之间动态调度 GPU 内存。

5.4 准确率评估

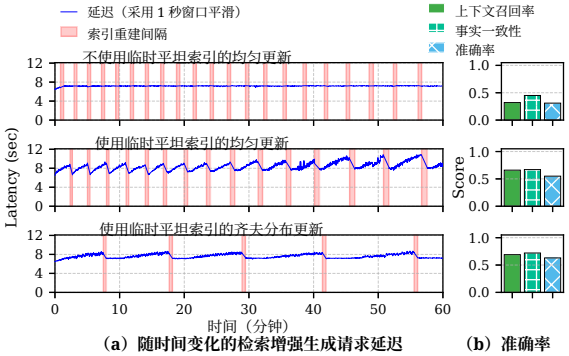
图 8展示了文本和 PDF 检索流水线的准确率。我们使用 §3.4中定义的三个指标来评估检索增强生成的质量：上下文召回率、事实一致性和准确率。整体质量主要取决于生成模型的选择，而非向量数据库。如图 8中的文本流水线所示，在生成模型相同的情况下，将 LanceDB 替换为 Milvus 后，上下文召回率得分几乎没有变化。相比之下，在 LanceDB 配置相同的情况下，将生成模型从 Qwen-7B 升级到 Qwen-72B，可使事实一致性得分提高 1.67×，准确率得分提高 1.51×。

PDF 流水线呈现出不同的准确率趋势，这一趋势同时受检索质量和模型容量的影响。得益于多向量检索和重排序，[50]，LanceDB 配置的平均上下文召回率达到 0.84。使用通义千问 QwenVL-32B 这类能力足够强的模型时，如此高的召回率可转化为 0.77 的最高准确率。然而，对于小模型而言，较高的召回率并不能带来更高的准确率。Milvus+通义千问 QwenVL-3B 的上下文召回率达到 0.52，但准确率仅为 0.35，这表明小型视觉语言模型的模型容量不足，无法有效利用检索到的 PDF 上下文，尽管这些上下文与问题相关。这说明，在 PDF 流水线中，模型容量是决定高检索质量能否转化为准确答案的瓶颈。

- 较高的上下文召回率并不总能保证更高的准确率，因为小模型可能缺乏有效利用检索上下文的能力。

5.5 更新操作

我们使用三种配置评估持续文档更新情况下的文本流水线，并研究更新对请求延迟和响应准确率的影响。我们将 RAGPerf 配置为针对 LanceDB 发起混合工作负载（50% 查询和 50% 更新）



（a）随时间变化的检索增强生成请求延迟　（b）准确率

图9：更新工作负载下文本流水线的延迟。

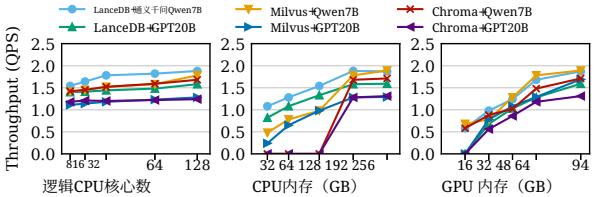


图10：资源受限时的文本流水线性能。

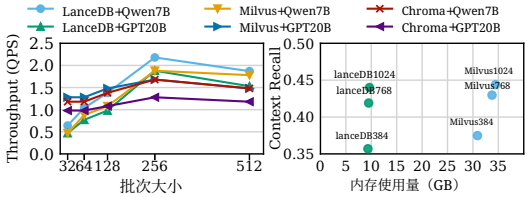


图 11：不同批次大小和嵌入维度下的文本流水线性能。

，使用 IVF-HNSW [2] 索引，运行 60分钟时间窗口，并采用通义千问-7B 作为生成模型。图 9 展示了检索增强生成请求随时间变化的端到端延迟及对应的准确率得分。

我们首先展示在均匀更新分布下未配置临时平坦索引时的端到端性能。没有临时平坦索引时，新插入的条目在下一次完整索引重建之前无法搜索。因此，延迟稳定在约 6.8 秒，而上下文召回率和查询准确率则保持在较低水平。

在 LanceDB 中启用临时平坦索引即可解决这一问题：新条目会缓冲在一个独立结构中，并在每次检索时进行线性扫描。这样可确保新更新的文档立即可搜索，从而恢复上下文召回率和准确性指标。不过，由于平坦索引在重建之间不断增长，延迟会逐渐从 7.01 秒升至 10.7 秒；每次将平坦索引合并到主索引后，延迟都会骤降。

在第三种配置中，我们采用相同的临时平坦索引，但使用齐夫更新分布，此时修改集中在少量文档上。平坦索引累积的唯一条目较少，因此延迟从 6.5 秒升至 8.7 秒的过程更为平缓，索引重建也不那么频繁，而准确性指标则保持不变。

- RAGPerf 中的动态工作负载生成器使用户能够评估不同更新策略和分布下检索质量与查询性能之间的权衡。
- 临时平坦索引使新更新的文档能够立即被检索，从而提升检索质量，但随着平坦索引的增长，延迟也会增加。如果忽略下次重建前的所有更新，搜索延迟将保持稳定，但流水线会检索过时的数据，导致准确率下降。
- 采用临时平坦索引时，与均匀更新相比，偏斜更新分布所维护的平坦结构中的唯一条目更少，因此延迟开销更低。

5.6 多种资源配置的影响

现在，我们展示 RAGPerf 量化不同系统资源配置下 RAG 系统性能影响的能力。我们评估了 RAG 流水线的性能

采用不同的中央处理器、主机内存和 GPU 内存配置。图 10 展示了端到端吞吐量（每秒查询数）。首先，中央处理器核心数对吞吐量影响不大。与使用

128 个核心的系统相比，该流水线在 32 个核心时可达到峰值吞吐量的 90.3%，在 8 个核心时可达到 78.2%。相比之下，主机内存限制会造成严重的性能损失。我们将 Milvus 和 LanceDB 配置为使用基于磁盘的向量索引（DiskANN 和 IVF-HNSW）；在可用内存为 32 GB 时，Milvus 和 LanceDB 的吞吐量分别降至 15.3 % 和 37.6%。这种性能下降主要是由于输入/输出开销，使检索延迟增加了 6.1× 至 12.5×。Chroma 完全依赖内存中的 HNSW 索引，当主机内存低于 128 GB 时无法运行。最后，GPU 内存限制会显著影响可扩展性。将 GPU 内存容量降至 32 GB 会限制推理批次大小，使平均吞吐量降至 47.1%。在 16 GB GPU 内存限制下，所有 GPT-20B 实验均无法运行，因为该容量不足以加载模型权重。

- CPU 核心数对 RAG 吞吐量的影响很小，因为检索和索引算法的计算密集度不高。相比之下，主机内存不足会迫使系统采用基于磁盘的索引，从而显著降低吞吐量。
- GPU 内存是主要的硬件瓶颈，因为它直接限制推理批次大小，并决定能否将更大的大语言模型完整加载到 GPU 中以提供服务。

5.7 敏感性分析

RAGPerf 支持广泛的流水线设置（§3.3）。我们通过分析一组具有代表性的参数上的端到端 RAG 吞吐量来加以展示。

批次大小。 图 11 展示了改变批次大小对性能的影响。将批次大小从 32 增加到 256，在所有配置中都能显著提升吞吐量。例如，LanceDB+Qwen7B 配置的吞吐量提高了 3.6×，这得益于 GPU 并行度的提高。然而，将批次大小进一步增加到 512 会使吞吐量最多降低 21%，因为更大的批次需要成比例更多的 GPU 内存用于 KV 缓存，从而限制了可并行解码的词元数量，并迫使采用顺序解码步骤，进而增加生成延迟。

嵌入维度。 图 11 展示了不同向量维度下上下文召回率与索引内存占用之间的权衡。LanceDB 的内存使用量显著较低

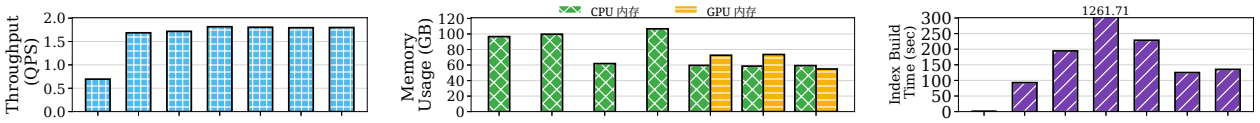


图 12：采用不同索引方案的文本流水线性能。

(≈10 GB)，而 Milvus 为 (>30GB)。这种高效性归功于 LanceDB 的架构：打开表时无需将完整索引加载到主机内存中；相比之下，Milvus 会在集合初始化时将整个索引加载到内存中。值得注意的是，在同一数据库中，使用 IVF_PQ 索引时，不同维度下的内存使用量变化很小。这是因为乘积量化主要使用固定长度的量化码表示向量，因此嵌入维度的增加对索引总大小影响不大。就检索质量而言，维度较高的向量始终能实现更好的上下文召回率。**向量数据库索引**。图 12 展示了以 Milvus 为数据库平台，对向量数据库索引类型进行的全面性能比较，因为它支持大多数向量索引类型。暴力 FLAT 索引作为基线，将端到端检索增强生成吞吐量限制在每秒查询数 0.69。其他近似最近邻（ANN）索引的吞吐量更高，为每秒查询数 1.68 至 1.81。不过，不同索引方法的资源需求各不相同：HNSW 索引的成本最高，需要超过 100 GB 的主机内存，且构建时间最长（>1200秒），因此不适用于更新频繁的知识库。相比之下，量化 IVF_PQ 索引实现了最有效的平衡：吞吐量表现出色，同时构建速度最快（200 秒），且内存效率很高。此外，我们的分析表明，基于 GPU 的索引吸引力较低，因为其观测到的吞吐量提升幅度有限（1.04×），与其资源需求（70GB GPU 内存用于 GPU_CAGRA）相比并不显著；鉴于 GPU 设备对生成阶段也至关重要，这是一个需要重点考虑的因素。

- 增大批次大小可以提高服务吞吐量。然而，额外的请求可能会占用 GPU 内存，而这些内存原本用于 KV 缓存，从而导致整体性能下降。

- GPU 加速索引会消耗大量 GPU 内存，但只能带来有限的性能提升。相反，通过为大语言模型推理预留更多 GPU 内存，我们可以为检索增强生成流水线带来更大的性能收益。

5.8 开销分析

我们通过测量 RAGPerf 对工作负载的干扰并量化其资源消耗，评估其开销。我们比较启用和未启用性能分析时，文本流水线和 PDF 流水线的端到端查询延迟。性能分析带来的开销可以忽略不计，单次迭代时间仅增加 0.11%。即使采样间隔为 100 毫秒，监控器平均每秒也只产生约 48 KB 的磁盘写入。为控制内存开销，RAGPerf 为每项指标分配一个固定大小为 2 MB 的循环缓冲区，避免长时间运行的工作负载导致内存无限增长。同时，在内存中缓冲指标可减少磁盘写入频率，进而降低中央处理器利用率。总体而言，性能分析器本身造成的资源开销极小，中央处理器使用量不到 0.26%，系统 DRAM 使用量峰值为 243 MB。

6 相关工作

检索增强生成的基准测试。先前的研究提出了多种基准测试，用于评估检索增强生成流水线 [25, 33, 46, 47, 53, 61, 67, 85, 88, 95, 104, 105]。其中一类侧重于评估检索上下文和生成答案的语义质量与准确性。例如，BEIR [85] 和 RAGBench [25] 提供了多样化的数据集，用于衡量检索精确率和生成忠实度。同样，TruthfulQA [53]，、RAGTruth [61]，和 FELM [104]等基准测试会评估检索增强生成所生成答案的事实性、幻觉率和语义对齐。另一类则评估检索增强生成各个组件的性能。VectorDBBench [105] 和 VESPA [67] 评估向量数据库，但不考虑生成阶段。AutoRAG [46] 和 Frames [47] 等框架则根据检索增强生成的响应质量优化流水线配置。不同于以往仅局限于准确性指标，或分别评估检索与生成的研究，RAGPerf 提出了一种端到端检索增强生成式人工智能系统基准测试框架。

检索增强生成的系统优化。许多研究致力于优化检索增强生成系统的性能 [4, 39–41, 58]。RAGX [58] 和 IKS [70] 通过将近似最近邻搜索卸载到近存储加速器来提升检索性能。PipeRAG [40] 采用算法与系统协同设计方法来提高系统资源利用率。Berchan-sky 等人 [4] 通过消除检索上下文中的低影响词元来提高生成吞吐量。面向系统的扩展进一步优化了检索增强生成的执行，例如缓存可复用的检索结果，以降低延迟和后端负载 [41]。与这些工作相一致，RAGPerf 提供了一个易于使用的框架，通过详尽的系统性能剖析以及对检索增强生成流水线中不同设计权衡的全面研究，助力系统优化。

检索增强生成扩展。近期研究通过引入新的检索与推理机制扩展了检索增强生成范式 [28, 32, 54, 87, 101]。HyDE [28]和 HyPE [87] 通过生成假设文档或提示词，改善提示词与上下文之间的匹配。ImpRAG [101]将生成过程中产生的中间表示用作检索嵌入。GraphRAG [32]借助知识图谱扩展了标准检索增强生成，而 GGR [54] 则进一步利用图神经网络（GNN）来指导检索。得益于易于配置的模块化设计，RAGPerf 能够支持此类扩展。

7 结论

我们开发了 RAGPerf，这是一个用于刻画 RAG 流水线系统行为的端到端基准测试框架。RAGPerf 涵盖完整的数据嵌入、索引、检索和生成工作流。它采用贴近实际的多模态工作负载生成器和系统性能剖析，帮助开发者识别 RAG 流水线中的性能瓶颈。它还支持用户灵活配置 RAG 流水线并探索不同的设计权衡。实验结果表明，RAGPerf 能够有效量化关键设计权衡，同时带来的性能分析开销可以忽略不计。

参考文献

[1] Shuai Bai、Keqin Chen、Xuejing Liu、Jialin Wang、Wenbin Ge、Sibo Song、Kai Dang、Peng Wang、Shijie Wang、Jun Tang、Humen Zhong、Yuanzhi Zhu、Mingkun Yang、Zhaohai Li、Jianqiang Wan、Pengfei Wang、Wei Ding、Zheren Fu、Yiheng Xu、Jiabo Ye、Xi Zhang、Tianbao Xie、Zesen Cheng、Hang Zhang、Zhibo Yang、Haiyang Xu 和 Junyang Lin。2025年。Qwen2.5-VL 技术报告。arXiv: 2502.13923 [ ] 2502.13923。cs.CV

[2] https://arxiv.org/abs/。Dmitry Baranchuk、Artem Babenko 和 Yury Malkov。2018年。重新审视用于十亿级近似最近邻搜索的倒排索引。载于《欧洲计算机视觉会议（ECCV）论文集》。202–216。[3] Ryan C Barron、Maksim E Eren、Olga M Serafimova、Cynthia Matuszek 和 Boian S Alexandrov。2025年。弥合法律知识与人工智能：结合向量存储、知识图谱和层次非负矩阵分解的检索增强生成。arXiv 预印本 arXiv:2502.20364（2025年）。[4] Moshe Berchansky、Peter Izsak、Avi Caciularu、Ido Dagan 和 Moshe Wasserblat。2023年。通过词元消除优化检索增强阅读器模型。arXiv 预印本 arXiv:2310.13682（2023年）。[5] Sartaj Bhuvaji、Prachitee Chouhan、Madhuroopa Irukulla、Jay Singhvi、Wan D. Bae 和 Shayma Alkobaisi。2025年。用于会议洞察提取的检索增强框架。载于第40届 ACM/SIGAPP应用计算研讨会论文集（意大利卡塔尼亚卡塔尼亚国际机场）（SAC '25）。计算机协会，纽约，美国，899–906 10 1145 3672608 3707915。https://doi.org/。/。[6] Matteo Buffa、Alfio Ferrara、Sergio Picascia、Davide Riva 和 Silvana Castano。2025年。利用检索增强生成改进法律文档构建。《计算机法律与安全评论》59（2025年），106229。[7] Tianchi Cai、Zhiwen Tan、Xierui Song、Tao Sun、Jiyan Jiang、Yunqi Xu、Yinger Zhang 和 Jinjie Gu。2024年。Forag：面向网页增强长篇问答的事实性优化检索增强生成。载于第30届 ACM SIGKDD知识发现与数据挖掘会议论文集。199–210。[8] Zongsheng Cao、Yangfan He、Anran Liu、Jun Xie、Feng Chen 和 Zhepeng Wang。2025年。TV-RAG：用于长视频检索与理解的时间感知与语义熵加权框架。载于第33届 ACM国际多媒体会议论文集（爱尔兰都柏林）（MM '25）。计算机协会，纽约，美国，9071–9079。https://doi.org/10.1145/3746027.3755873[9] Bonggeun Choi、Jeongjae Park、Yoonsung Kim、Jaehyun Park 和 Youngjoong Ko。2024年。RAC：用于对话场景下开放域问答的检索增强对话数据集。载于2024年自然语言处理实证方法会议论文集：工业应用专题。1477–1488。[10] Chanyeol Choi、Jihoon Kwon、Jaeseon Ha、Hojun Choi、Chaewoon Kim、Yongjae Lee、Jy-yong Sohn 和 Alejandro Lopez-Lira。2025年。Finder：用于问答和评估检索增强生成的金融数据集。载于第6届 ACM金融人工智能国际会议论文集。638–646。[11] Chroma团队。2025年。Chroma。https://www.trychroma.com/ 向量数据库。[12] codepattot。2026年。GitHub代码。https://huggingface.co/datasets/codeparrot/github-code [13] Cohere。2023年。Cohere重排序。https://docs.cohere.com/docs/rerank-overview。[14] Brian F. Cooper、Adam Silberstein、Erwin Tam、Raghu Ramakrishnan 和 Russell Sears。2010年。使用 YCSB 对云服务系统进行基准测试。载于第1届 ACM云计算研讨会论文集（美国印第安纳州印第安纳波利斯）（SoCC '10）。计算机协会，纽约，美国，143–154 10 1145 1807128 1807152。https://doi.org/。/。[15] Hudson de Martim。2025。面向法律规范的图检索增强生成：一种层次化与时间方法。arXiv 预印本 arXiv:2505.00039（2025年）。[16] DeepSeek-AI。2025年。DeepSeek-R1：通过强化学习激励大语言模型的推理能力。arXiv:2501.12948 [cs.CL] https://arxiv.org/abs/2501.12948。Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019年。BERT：面向语言理解的深度双向变换器预训练。arXiv: 1810.04805 [ ] 1810.04805。cs.CL

[17] https://arxiv.org/abs/。Djellel Eddine Difallah、Andrew Pavlo、Carlo Curino 和 Philippe Cudre-Mauroux。2013 年。OLTP-Bench：用于关系数据库基准测试的可扩展测试平台。7，4（2013年12月），277–288。https://doi.org/10.14778/2732240.2732246[18] Yiming Du、Hongru Wang、Zhengyi Zhao、Bin Liang、Baojun Wang、Wanjun Zhong、Zezhong Wang 和 Kam-Fai Wong。2024年。Perltqa：用于问答中记忆分类、检索和综合的个人长期记忆数据集。arXiv 预印本 arXiv:2402.16288（2024年）。[19] Elastic。2025年。Elasticsearch。https://www.elastic.co/elasticsearch/ 分布式搜索与分析引擎。[20] Shahul Es、Jithin James、Luis Espinosa-Anke 和 Steven Schockaert。2025年。Ragas：检索增强生成的自动评估。arXiv: 2309.15217 [ ] 2309.15217。cs.CL

[21] https://arxiv.org/abs/。Aneela Farnaz 和 Chris Huyck。2026。TravQuery：一种基于检索增强生成（RAG）的客户支持聊天机器人。载于《人工智能 XLII》，Max Bramer 和 Frederic Stahl (Eds.). Springer 瑞士施普林格·自然，沙姆，130–140。[23] Manuel Faysse、Hugues Sibille、Tony Wu、Bilel Omrani、Gautier Viaud、Céline Hudelot 和 Pierre Colombo。2024 年。ColPali：利用视觉语言模型实现高效文档检索。arXiv: 2407.01449 [ ] 2407.01449。cs.IR

[22] 2025 https://arxiv.org/abs/。维基媒体基金会。。维基媒体下载。https: //dumps.wikimedia.org [25] Robert Friel、Masha Belyi 和 Atindriyo Sanyal。2024 年。Ragbench：面向检索增强生成系统的可解释基准测试。arXiv 预印本 arXiv:2407.11005（2024）。[26] Daniel Galvez、Greg Diamos、Juan Ciro、Juan Felipe Cerón、Keith Achorn、Anjali Gopi、David Kanter、Maximilian Lam、Mark Mazumder 和 Vijay Janapa Reddi。2021 年。The People's Speech：面向商业用途的大规模、多样化英语语音识别数据集。arXiv:2111.09344 [cs.LG] https://arxiv.org/abs/2111.09344[27] Jingtong Gao、Bo Chen、Xiangyu Zhao、Weiwen Liu、Xiangyang Li、Yichao Wang、Wanyu Wang、Huifeng Guo 和 Ruiming Tang。2025 年。LLM4Rerank：基于大语言模型的推荐自动重排序框架。载于 2025年万维网大会。https://openreview.net/forum?id=HEBVEmK22u [28] Luyu Gao、Xueguang Ma、Jimmy Lin 和 Jamie Callan。2023 年。无需相关性标签的精准零样本稠密检索。载于计算语言学协会第61届年会论文集（第1卷：长篇论文）。1762–1777。[29] Tianhong Gao、Jundong Shen、Bei Shi、Jiapeng Wang、Ying Ju、Junfeng Yao、Jiao Ran、Yong Zhang、Lin Dong、Huiyu Yu 等。2025 年。OlaMind：迈向类人且防幻觉的检索增强对话客户服务。arXiv 预印本 arXiv:2510.22143（2025）。[30] Yunfan Gao、Yun Xiong、Xinyu Gao、Kangxiang Jia、Jinliu Pan、Yuxi Bi、Yixin Dai、Jiawei Sun、Haofen Wang 和 Haofen Wang。2023 年。面向大语言模型的检索增强生成：综述。arXiv 预印本 arXiv:2312.10997 2, 1 2023 . ( )。[31] Ruiqi Guo、Philip Sun、Erik Lindgren、Quan Geng、David Simcha、Felix Chern 和 Sanjiv Kumar。2020 年。利用各向异性向量量化加速大规模推理。载于国际机器学习大会。https://arxiv.org/abs/1908.10396[32] Haoyu Han、Yu Wang、Harry Shomer、Kai Guo、Jiayuan Ding、Yongjia Lei、Mahantesh Halappanavar、Ryan A. Rossi、Subhabrata Mukherjee、Xianfeng Tang、Qi He、Zhigang Hua、Bo Long、Tong Zhao、Neil Shah、Amin Javari、Yinglong Xia 和 Jiliang Tang。2025 年。基于图的检索增强生成（GraphRAG）。arXiv: 2501.00309 [ ] 2501.00309。cs.IR

[33] https://arxiv.org/abs/。Cheng-Ping Hsieh、Simeng Sun、Samuel Kriman、Shantanu Acharya、Dima Rekesh、Fei Jia、Yang Zhang 和 Boris Ginsburg。2024 年。RULER：长上下文语言模型的真实上下文长度究竟是多少？arXiv 预印本 arXiv:2404.06654（2024）。[34] Yizheng Huang 和 Jimmy Huang。2024 年。面向大语言模型的检索增强文本生成综述。arXiv 预印本 arXiv:2404.10981（2024）。[35] Binyuan Hui、Jian Yang、Zeyu Cui、Jiaxi Yang、Dayiheng Liu、Lei Zhang、Tianyu Liu、Jiajun Zhang、Bowen Yu、Kai Dang 等。2024 年。通义千问2.5-Coder 技术报告。arXiv 预印本 arXiv:2409.12186（2024）。[36] JaidedAI。2020 年。EasyOCR：开箱即用的光学字符识别，支持 80+ 多种语言。https://github.com/JaidedAI/EasyOCR。[37] Suhas Jayaram Subramanya、Fnu Devvrit、Harsha Vardhan Simhadri、Ravishankar Krishnawamy 和 Rohan Kadekodi。2019 年。DiskANN：在单节点上快速、精准地搜索十亿级点的最近邻。载于神经信息处理系统进展，H. Wallach、H. Larochelle、A. Beygelzimer、F. d'Alché-Buc、E. Fox 和 R. Garnett（编），第32卷。Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/09853c7fb1d3f8ee67a61b6bf4a7f8e6-Paper.pdf [38] Soyeong Jeong、Kangsan Kim、Jinheon Baek 和 Sung Ju Hwang。2025 年。VideoRAG：对视频语料库进行检索增强生成。arXiv:2501.05874 [cs.CV] https://arxiv.org/abs/2501.05874[39] Wenqi Jiang、Suvinay Subramanian、Cat Graves、Gustavo Alonso、Amir Yazdanbakhsh 和 Vidushi Dadu。2025 年。Rago：检索增强生成服务的系统化性能优化。载于第52届年度国际计算机体系结构研讨会论文集。974–989。[40] Wenqi Jiang、Shuai Zhang、Boran Han、Jie Wang、Bernie Wang 和 Tim Kraska。2024 年。PipeRAG：通过算法与系统协同设计实现快速检索增强生成。arXiv 预印本 arXiv:2403.05676（2024）。[41] Chao Jin、Zili Zhang、Xuanlin Jiang、Fangyue Liu、Shufan Liu、Xuanzhe Liu 和 Xin Jin。2025 年。Ragcache：用于检索增强生成的高效知识缓存。ACM计算机系统汇刊 44, 1（2025），1–27。[42] Nikhil Kandpal、Brian Lester、Colin Raffel、Sebastian Majstorovic、Stella Biderman、Baber Abbasi、Luca Soldaini、Enrico Shippole、A. Feder Cooper、Aviya Skowron、John Kirchenbauer、Shayne Longpre、Lintang Sutawika、Alon Albalak、Zhenlin Xu、Guilherme Penedo、Loubna Ben Allal、Elie Bakouch、John David Pressman、Honglu Fan、Dashiell Stander、Guangyu Song、Aaron Gokaslan、Tom Goldstein、Brian R. Bartoldson、Bhavya Kailkhura 和 Tyler Murray。2025 年。The Common Pile v0.1：一个包含 8TB 公有领域文本和开放许可文本的数据集。

arXiv:2506.05209 [cs.CL] https://arxiv.org/abs/2506.05209[43] Jeongwoo Kang、Markarit Vartampetian、Felix Herron、Yongxin Zhou、Diandra Fabre 和 Gabriela Gonzalez-Saez。2025 年。GETALP@AutoMin 2025：利用检索增强生成，根据会议转录文本回答问题。arXiv:2508.00476 [cs.CL] https://arxiv.org/abs/2508.00476[44] Vladimir Karpukhin、Barlas Oğuz、Sewon Min、Patrick Lewis、Ledell Wu、Sergey Edunov、Danqi Chen 和 Wen-tau Yih。2020 年。用于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（EMNLP）论文集。计算语言学协会，6769–6781 – 10 – 18653 1 2020 550 。https://doi.org/ 。/v / . [45] emnlp-main。Omar Khattab 和 Matei Zaharia。。ColBERT：通过 BERT 上下文化晚期交互实现高效且有效的段落搜索。arXiv:2004.12832 [cs.IR] https://arxiv.org/abs/ 2004 12832 。Dongkyu Kim、[46] Byoungwook Kim、Donggeon Han 和 Matouš Eibich。2024 年。AutoRAG：用于优化检索增强生成流程的自动化框架。arXiv 预印本 arXiv:2410.20878（2024）。[47] Satyapriya Krishna、Kalpesh Krishna、Anhad Mohananey、Steven Schwarcz、Adam Stambler、Shyam Upadhyay 和 Manaal Faruqui。2025 年。事实、检索与推理：对检索增强生成进行统一评估。载于2025年计算语言学协会美洲分会会议：人类语言技术论文集（第1卷：长篇论文）。4745–4759。[48] Tom Kwiatkowski、Jennimaria Palomaki、Olivia Redfield、Michael Collins、Ankur Parikh、Chris Alberti、Danielle Epstein、Illia Polosukhin、Jacob Devlin、Kenton Lee、Kristina Toutanova、Llion Jones、Matthew Kelcey、Ming-Wei Chang、Andrew M. Dai、Jakob Uszkoreit、Quoc Le 和 Slav Petrov。2019 年。Natural Questions：问答研究基准测试。计算语言学协会会刊 7（2019年08月），453–466。https://doi.org/10.1162/tacl_a_00276 arXiv:https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00276/1923288/tacl_a_00276.pdf[49] Woosuk Kwon、Zhuohan Li、Siyuan Zhuang、Ying Sheng、Lianmin Zheng、Cody Hao Yu、Joseph E. Gonzalez、Hao Zhang 和 Ion Stoica。2023 年。利用分页注意力实现大语言模型服务的高效内存管理。arXiv: 2309 06180 [ ] 2309 06180 . cs.LG [50] https://arxiv.org/abs/ 2025 。LanceDB Inc.。LanceDB 中的多向量搜索。https://lancedb.com/docs/search/multivector-search/。LanceDB 文档。[51] Van-Duc Le 和 Hai-Thien To。2025 年。RAG-IT：面向自动化金融分析的检索增强指令微调。技术报告。arXiv.org。[52] Patrick Lewis、Ethan Perez、Aleksandra Piktus、Fabio Petroni、Vladimir Karpukhin、Naman Goyal、Heinrich Küttler、Mike Lewis、Wen-tau Yih、Tim Rocktäschel 等。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（NeurIPS），第33卷。9459–9474。[53] Stephanie Lin、Jacob Hilton 和 Owain Evans。2022 年。TruthfulQA：衡量模型模仿人类谬误的程度。载于计算语言学协会第60届年会论文集（第1卷：长篇论文）。3214–3252。[54] Haochen Liu、Song Wang 和 Jundong Li。2025 年。通过图神经网络引导提示实现知识图谱检索增强生成。载于第二届语言建模会议。[55] Nikolaos Livathinos、Christoph Auer、Maksym Lysak、Ahmed Nassar、Michele Dolfi、Panos Vagenas、Cesar Berrospi Ramis、Matteo Omenetti、Kasper Dinkla、Yusik Kim、Shubham Gupta、Rafael Teixeira de Lima、Valery Weber、Lucas Morin、Ingmar Meijer、Viktor Kuropiatnyk 和 Peter W. J. Staar。2025 年。Docling：用于人工智能驱动的文档转换的高效开源工具包。arXiv:2501.17887 [cs.CL] https://arxiv.org/abs/2501.17887[56] Yongdong Luo、Xiawu Zheng、Guilin Li、Shukang Yin、Haojia Lin、Chaoyou Fu、Jinfa Huang、Jiayi Ji、Fei Chao、Jiebo Luo 和 Rongrong Ji。2025 年。VideoRAG：视觉对齐的检索增强长视频理解。arXiv: 2411 13093 [ ] 2411 13093 . cs.CV [57] https://arxiv.org/abs/ 。Xueguang Ma、Liang Wang、Nan Yang、Furu Wei 和 Jimmy Lin。2023 年。微调 LLaMA 以实现多阶段文本检索。arXiv:2310.08319 [cs.IR] https://arxiv.org/abs/2310.08319[58] Rohan Mahapatra、Harsha Santhanam、Christopher Priebe、Hanyang Xu 和 Hadi Esmaeilzadeh。2025 年。检索增强生成即服务的存储内加速。载于第52届年度国际计算机体系结构研讨会论文集。450–466。[59] Yu A Malkov 和 Dmitry A Yashunin。2018 年。使用分层可导航小世界图实现高效且稳健的近似最近邻搜索。IEEE模式分析与机器智能汇刊 42, 4（2018），824–836。[60] Mingyang Mao、Mariela M. Perez-Cabarcas、Utteja Kallakuri、Nicholas R. Waytowich、Xiaomin Lin 和 Tinoosh Mohsenin。2025 年。多模态检索增强生成系统 Multi-RAG：实现自适应视频理解。arXiv:2505.23990 [cs.AI] https://arxiv.org/abs/2505.23990[61] Cheng Niu、Yuanhao Wu、Juno Zhu、Siliang Xu、Kashun Shum、Randy Zhong、Juntong Song 和 Tong Zhang。2024 年。RAGTruth：用于开发可信检索增强语言模型的幻觉语料库。载于论文集

计算语言学协会第62届年会（第1卷：长篇论文）。10862–10878。[62] NVIDIA。2025年。NVIDIA 管理库（NVML）。https: //developer.nvidia.com/management-library-nvml。访问日期：2025年8月1日。[63] NVIDIA。2026年。NVIDIA NVLink 和 NVLink 交换机。= https://www.nvidia.com/en-us/data-center/nvlink/。[64] OpenAI。2025年。gpt-oss-120b 和 gpt-oss-20b 模型卡。arXiv:2508.10925 [cs.CL] https://arxiv.org/abs/2508.10925[65] Weston Pace、Chang She、Lei Xu、Will Jones、Albert Lockett、Jun Wang 和 Raunak Shah。2025年。Lance：通过自适应结构编码实现列式存储中的高效随机访问。arXiv:2504.15247 [cs.DB] https://arxiv.org/abs/ 2504 15247 。Piyushkumar Patel。图增强检索增强问答在电子商务客户支持中的应用。[66] arXiv预印本 arXiv:2509.14267（2025年）。[67] Norman W. Paton、M. Howard Williams、Kosmas Dietrich、Olive Liew、Andrew Dinn 和 Alan Patrick。2000 年。VESPA：向量空间数据库基准测试。载于《数据库进展》，Brian Lings 和 Keith Jeffery（编）。[68] Nicholas Pipitone 和 Ghita Houir Alami。2024年。Legalbench-rag：法律领域检索增强生成基准测试。arXiv预印本 arXiv:2408.10343（2024年）。[69] Qdrant 团队。2025年。Qdrant。https://qdrant.tech/ 向量数据库。[70] Derrick Quinn、Mohammad Nouri、Neel Patel、John Salihu、Alireza Salemi、Sukhan Lee、Hamed Zamani 和 Mohammad Alian。2025年。加速检索增强生成。载于第30届ACM编程语言与操作系统架构支持国际会议论文集，第1卷。15–32。[71] 通义千问、：、An Yang、Baosong Yang、Beichen Zhang、Binyuan Hui、Bo Zheng、Bowen Yu、Chengyuan Li、Dayiheng Liu、Fei Huang、Haoran Wei、Huan Lin、Jian Yang、Jianhong Tu、Jianwei Zhang、Jianxin Yang、Jiaxi Yang、Jingren Zhou、Junyang Lin、Kai Dang、Keming Lu、Keqin Bao、Kexin Yang、Le Yu、Mei Li、Mingfeng Xue、Pei Zhang、Qin Zhu、Rui Men、Runji Lin、Tianhao Li、Tianyi Tang、Tingyu Xia、Xingzhang Ren、Xuancheng Ren、Yang Fan、Yang Su、Yichang Zhang、Yu Wan、Yuqiong Liu、Zeyu Cui、Zhenru Zhang 和 Zihan Qiu。2025年。Qwen2.5技术报告。arXiv: 2412 15115 [ ] 2412 15115 。cs.CL [72] https://arxiv.org/abs/。Alec Radford、Jong Wook Kim、Chris Hallacy、Aditya Ramesh、Gabriel Goh、Sand- hini Agarwal、Girish Sastry、Amanda Askell、Pamela Mishkin、Jack Clark 等。2021年。通过自然语言监督学习可迁移视觉模型。载于《国际机器学习大会》。PMLR，8748–8763。[73] Alec Radford、Jong Wook Kim、Tao Xu、Greg Brockman、Christine McLeavey 和 Ilya Sutskever。2022年。通过大规模弱监督实现稳健语音识别。arXiv:2212.04356 [eess.AS] https://arxiv.org/abs/2212.04356[74] Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J. Liu。2023年。探索统一文本到文本转换器的迁移学习极限。arXiv: 1910 10683 [ ] 1910 10683 。cs.LG https://arxiv.org/abs/。[75] Mercy Ranjit、Gopinath Ganapathy、Ranjit Manuel 和 Tanuja Ganu。2023年。使用 OpenAI GPT 模型进行检索增强的胸部X光报告生成。arXiv: 2305 03660 [ ] 2305 03660 。cs.CL [76] https://arxiv.org/abs/。RapidAI 团队。RapidOCR：OCR 工具箱。https://github.com/RapidAI/RapidOCR。[77] Nils Reimers 和 Iryna Gurevych。2019年。Sentence-BERT：使用孪生 BERT 网络生成句子嵌入。载于《2019年自然语言处理实证方法会议论文集》。计算语言学协会。https://arxiv.org/abs/ 1908 10084 。Xubin Ren、Lingrui Xu、Long Xia、Shuaiqiang Wang、Dawei Yin 和 Chao Huang。2025年。VideoRAG：借助超长上下文视频实现检索增强生成。arXiv: 2502 01549 [ ] 2502 01549 。[78] cs.IR https://arxiv.org/abs/。Victor Sanh、Lysandre Debut、Julien Chaumond 和 Thomas Wolf。2020年。DistilBERT，BERT的蒸馏版本：更小、更快、更便宜、更轻量。arXiv: 1910 01108 [ ] 1910 01108 。cs.CL https://arxiv.org/abs/。[79] Sivic 和 Zisserman。Video Google：一种用于视频目标匹配的文本检索方法。载于第九届IEEE国际计算机视觉会议论文集。IEEE，1470–1477。[80] R. Smith。2007年。Tesseract OCR 引擎概述。载于第九届国际文档分析与识别会议（ICDAR 2003 2007 2 629 633 10 1109 2007 4376991 ），第 2007 2 629 633 10 1109 2007 4376991 卷。–。https://doi.org/。/ [82] ICDAR。。Steven Song、Anirudh Subramanyam、Irene Madejski 和 Robert L. Grossman。2025年。LaB-RAG：用于放射学报告生成的标签增强检索增强生成。arXiv: 2411 16523 [ ] 2411 16523 。cs.CV [83] https://arxiv.org/abs/。Liwen Sun、James Zhao、Megan Han 和 Chenyan Xiong。事实感知多模态检索增强，用于准确生成医学放射学报告。arXiv: 2407 15268 [ ] 2407 15268 。cs.CL [84] https://arxiv.org/abs/。SYSTRAN。faster-whisper。https://github.com/SYSTRAN/faster-whisper。使用 CTranslate2 对 OpenAI 的 Whisper 进行重新实现。[85] Nandan Thakur、Nils Reimers、Andreas Rücklé、Abhishek Srivastava 和 Iryna Gurevych。2021年。BEIR：用于信息检索模型零样本评估的异质性基准测试。arXiv:2104.08663 [cs.IR]

https://arxiv.org/abs/ 2104 08663 。Hugo Touvron、[86] Thibaut Lavril、Gautier Izacard、Xavier Martinet、Marie-Anne Lachaux、Timothée Lacroix、Baptiste Rozière、Naman Goyal、Eric Hambro、Faisal Azhar、Aurelien Rodriguez、Armand Joulin、Edouard Grave 和 Guillaume Lample。2023年。LLaMA：开放且高效的基础语言模型。arXiv: 2302 13971 [ ] 2302 13971 。[87] cs.CLhttps://arxiv.org/abs/ 。Domen Vake、Jernej Vičič 和 Aleksandar Tošić。2025年。弥合检索增强生成中的问答差距：假设提示嵌入。IEEE Access（2025年）。[88] Hengyi Wang、Haizhou Shi、Shiwei Tan、Weiyi Qin、Wenyuan Wang、Tunyu Zhang、Akshay Nambi、Tanuja Ganu 和 Hao Wang。2025年。多模态大海捞针：多模态大语言模型长上下文能力基准测试。载于《2025年计算语言学协会美洲分会会议：人类语言技术》（第1卷：长篇论文）论文集。3221–3241。[89] Jinhong Wang、Tajamul Ashraf、Zongyan Han、Jorma Laaksonen 和 Rao Muhammad Anwer。2025年。MIRA：一种融合医疗检索增强生成中多种模态的新型框架。载于第33届ACM国际多媒体会议论文集（爱尔兰都柏林）（MM '25）。计算机协会，纽约州纽约市，美国，6307–6315。https://doi.org/10.1145/3746027.3755760[90] Jianguo Wang、Xiaomeng Yi、Rentong Guo、Hai Jin、Peng Xu、Shengjun Li、Xiangyu Wang、Xiangzhou Guo、Chengming Li、Xiaohai Xu、Kun Yu、Yuxing Yuan、Yinghao Zou、Jiquan Long、Yudong Cai、Zhenxiang Li、Zhifeng Zhang、Yihua Mo、Jun Gu、Ruiyi Jiang、Yi Wei 和 Charles Xie。2021年。Milvus：专为向量数据管理打造的系统。载于《2021年国际数据管理会议》（线上活动，中国）（SIGMOD '21）论文集。计算机协会，纽约州纽约市，美国，2614 2627 10 1145 3448016 3457550 – 。https://doi.org/ 。/ 。[91] Shuting Wang、Jiejun Tan、Zhicheng Dou 和 Ji-Rong Wen。2025年。Omnieval：金融领域全方位自动化检索增强生成评测基准。载于《2025年自然语言处理实证方法会议论文集》。5737–5762。[92] Wenhui Wang、Furu Wei、Li Dong、Hangbo Bao、Nan Yang 和 Ming Zhou。2020年。MiniLM：用于任务无关压缩预训练Transformer的深度自注意力蒸馏。arXiv:2002.10957 [cs.CL] https://arxiv.org/abs/2002.10957[93] Xinyu Wang、Jijun Chi、Zhenghan Tai、Tung Sum Thomas Kwok、Hailin He、Zhuhong Li、Yuchen Hua、Muzhi Li、Peng Lu、Suyucheng Wang 等。2025年。Finsage：用于金融文件问答的多方面检索增强生成系统。载于第34届ACM国际信息与知识管理会议论文集。6144–6152。

[94] Zheng Wang、Zhongyang Li、Zeren Jiang、Dandan Tu 和 Wei Shi。2024年。通过可编辑记忆图上的检索增强生成构建个性化智能体。arXiv 预印本 arXiv:2409.19401（2024年）。[95] Navve Wasserman、Roi Pony、Oshri Naparstek、Adi Raz Goldfarb、Eli Schwartz、Udi Barzelay 和 Leonid Karlinsky。2025年。REAL-MM-RAG：真实世界多模态检索基准。arXiv 预印本 arXiv:2502.12342（2025年）。[96] Shitao Xiao、Zheng Liu、Peitian Zhang、Niklas Muennighoff、Defu Lian 和 Jian-Yun Nie。2024年。C-Pack：通用中文嵌入打包资源。arXiv: 2309 07597 [ ] 2309 07597 。[97] cs.CLhttps://arxiv.org/abs/ 。Shan Xu、Zhaokun Yan、Chengxiao Dai 和 Fan Wu。2025年。MEGA-RAG：一种检索增强生成框架，通过多证据引导的答案优化，缓解大语言模型在公共卫生领域中的幻觉。《公共卫生前沿》13（2025年），1635381。[98] Zhentao Xu、Mark Jerome Cruz、Matthew Guevara、Tie Wang、Manasi Desh- pande、Xiaofeng Wang 和 Zheng Li。2024年。利用知识图谱进行检索增强生成，实现客户服务问答。载于第47届国际ACM SIGIR会议论文集：信息检索研究与开发。2905–2909。[99] Shi-Qi Yan、Jia-Chen Gu、Yun Zhu 和 Zhen-Hua Ling。2024年。纠正式检索增强生成。arXiv：2401。15884 [cs.CL][100] An Yang、Beichen Zhang、Binyuan Hui、Bofei Gao、Bowen Yu、Chengpeng Li、Dayiheng Liu、Jianhong Tu、Jingren Zhou、Junyang Lin、Keming Lu、Mingfeng Xue、Runji Lin、Tianyu Liu、Xingzhang Ren 和 Zhenru Zhang。2024年。Qwen2.5-Math 技术报告：通过自我改进迈向数学专家模型。arXiv 预印本 arXiv:2409.12122（2024年）。[101] Wenzheng Zhang、Xi Victoria Lin、Karl Stratos、Wen-tau Yih 和 Mingda Chen。2025年。ImpRAG：基于隐式查询的检索增强生成。arXiv 预印本 arXiv:2506.02279（2025年）。[102] Xin Zhang、Yanzhao Zhang、Dingkun Long、Wen Xie、Ziqi Dai、Jialong Tang、Huan Lin、Baosong Yang、Pengjun Xie、Fei Huang 等。2024年。mGTE：用于多语言文本检索的通用长上下文文本表示与重排序模型。arXiv 预印本 arXiv:2407.19669（2024年）。[103] Xuejiao Zhao、Siyan Liu、Su-Yin Yang 和 Chunyan Miao。2025年。Medrag：借助知识图谱启发的推理增强检索增强生成，打造医疗保健副驾驶。载于2025年ACM网络会议论文集。4442–4457。[104] Yiran Zhao、Jinghan Zhang、I Chern、Siyang Gao、Pengfei Liu、Junxian He 等。2023年。FELM：大语言模型事实性评估基准测试。《神经信息处理系统进展》36（2023年），44502–44523。[105] Zilliz。2024年。zilliztech/VectorDBBench：向量数据库基准测试工具。https://github.com/zilliztech/VectorDBBench。