编写优化 RAG 应用程序的最佳实践

AWS 规范性指导

AWS 规范性指导: 编写优化 RAG 应用程序的最佳实践

Table of Contents

编写优化 RAG 应用程序的最佳实践

Ivan Cui 和 Samantha Stuart，Amazon Web Services

2025 年 7 月（文档历史记录）

大型语言模型 (LLMs) 凭借其非凡的理解和生成类人文本的能力，彻底改变了人工智能领域。但是，他们面临一个显著的局限性：他们只能使用训练数据中包含的知识。这就是检索增强生成 (RAG) 的用武之地。它提供了一种 LLMs 与外部知识来源（例如您组织的数据和文档）相结合的解决方案。通过涉及信息检索和响应生成的两阶段过程，RAG使人工智能系统能够访问和整合来自不同来源 up-to-date的信息，从而做出更准确、更明智的响应，弥合静态模型知识和动态现实世界信息需求之间的差距。

如何优化内容以便在基于 RAG 的应用程序中进行检索？ 本指南提供了最佳实践，可帮助您优化知识库中基于文本的内容的格式和写作风格。优化内容可以增强上下文，从而帮助 RAG 应用程序更准确地理解特定于任务的信息。当系统检索到高度相关和准确的内容时，法学硕士的响应质量就会提高。在系统层面优化上下文交付过程称为上下文工程，它构成了代理RAG架构的重要组成部分。在 agentic RAG 中，一个或多个额外的 LLMs 理由，然后在 RAG 执行之前对录取请求采取行动。这便于多步骤的信息传送过程。随着RAG架构变得越来越复杂，源内容优化仍然是向其提供清晰上下文的最直接方法。LLMs这些最佳实践旨在帮助您最大限度地提高组织对RAG应用程序的投资。

目标受众

本指南适用于使用一个或多个 RAG 组件构建 LLM 应用程序的 AI 工程师、数据科学家、数据工程师或软件开发人员。要理解本指南中的概念和建议，您应该熟悉矢量数据库和提示 LLMs。

目标

本指南中的建议可以帮助您实现以下目标：

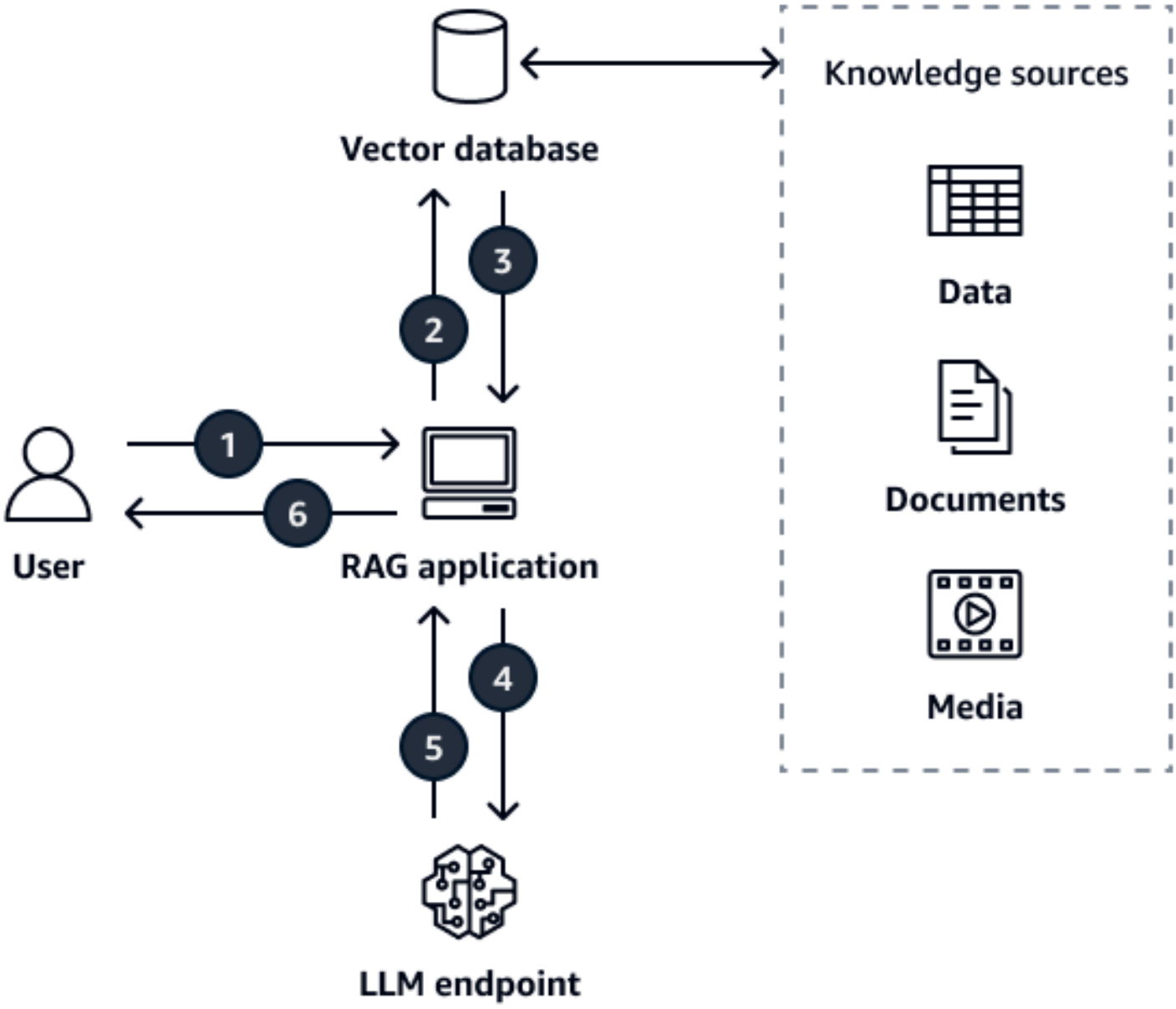
- 通过提供结构良好且语义丰富的源文档，针对令牌使用和冗余进行了优化，提高 RAG 应用程序生成的响应的准确性和相关性。
- 通过在源文档中提供清晰的定义和解释，帮助 RAG 应用程序更好地理解特定领域的知识和上下文。
- 通过遵守源文档中一致的格式和结构指南，便于对 RAG 应用程序进行更轻松的维护和知识库更新。
- 通过将大型单体文档分解为可以高效索引和检索的较小、独立的单元，提高 RAG 解决方案的可扩展性。

了解 LLM 和 RAG

要了解提高源文档质量如何提高RAG响应的质量，您必须了解法学硕士的内部运作方式。法学硕士的真正力量在于他们能够使用自我注意力机制和变压器架构。这些先进的技术使模型能够有效地处理和关联输入序列的不同部分，无论它们在文本中的位置或距离如何。这种能力与传统的语言模型形成鲜明对比，传统语言模型经常在长期依赖关系和上下文理解方面苦苦挣扎。此外，法学硕士的培训规模空前。一些最大的模型由数万亿个参数组成，并从不同的来源摄取了数 TB 的文本数据。这种庞大的规模使法学硕士能够对语言有丰富的理解，捕捉以前对人工智能系统来说具有挑战性的微妙差别、成语和情境线索。结果是一类模型可以生成连贯流畅的文本，并在问答、文本摘要甚至代码生成等任务中展示出非凡的能力。

要使用这些模型，我们可以求助于诸如 Amazon Bedrock之类的服务，该服务允许访问来自亚马逊和第三方提供商的各种基础模型，包括Anthropic、Cohere和Meta。您可以使用 Amazon Bedrock 对最先进的模型进行实验，对其进行自定义和微调，或者通过单个 API 将其整合到您的生成 AI-powered 解决方案中。

尽管法学硕士擅长捕捉模式和生成连贯的文本，但他们通常无法获得最新或专业的信息。RAG 将 LLM 的生成能力与检索组件相结合，该检索组件可以访问和整合来自外部来源的相关信息，作为具体化的 LLM 提示的一部分。外部来源的示例包括 Amazon Bedrock 知识库、Ama z on Kendra 等智能搜索系统或矢量数据库（例如亚马逊 OpenSearch 服务）。



该图描述了以下工作流程：

1. 用户向 RAG 应用程序提交查询。
2. RAG 应用程序查询包含知识来源（例如文档、数据或媒体）的矢量数据库。
3. RAG 应用程序根据查询和存储文档之间的语义相似性从矢量数据库中检索相关信息。
4. RAG 应用程序使用检索到的上下文对原始提示进行扩展，并将其发送到 LLM 端点。
5. LLM 端点生成响应并将其返回给 RAG 应用程序。
6. RAG 应用程序将生成的响应返回给用户。

从本质上讲，RAG采用了两个阶段的流程。在第一阶段，检索模型根据输入查询识别和检索相关文档或段落。这种检索模型可以是传统的信息检索系统，也可以是密集的检索模型，也可以是两者的组合。在第二阶段，检索到的信息和原始查询作为完全实现的提示模板输入到 LLM 中。LLM 在很大程度上依赖于检索器组件提供的源内容的质量。他们应用自我注意力机制对检索到的内容与任务的关系进行数学编码。然后，LLM 会根据查询和检索到的信息生成响应。在RAG中，控制检索到的源文档的质量是改善法学硕士任务内部表现形式的一种直接手段。RAG 使用相关的外部数据有效地增强了法学硕士的训练数据。这种方法使RAG能够利用法学硕士和检索系统的优势，从而能够根据当前和专业知识生成更准确、更明智的回应。

向量和嵌入

向量和嵌入是机器学习和自然语言处理的基本概念。向量是表示同时具有大小和方向的量的数学对象。在自然语言处理 (NLP) 的背景下，单词、句子或文档通常在高维向量空间中表示为向量。另一方面，嵌入是一种在低维向量空间中表示单词或文档等对象的方法，其中向量之间的关系捕捉语义或句法的相似性。例如，单词嵌入允许含义相似的单词具有相似的向量表示形式。这有助于算法更有效地理解和处理语言。

矢量数据库

在生成式 AI 中，矢量数据库是存储和管理文档、查询或其他对象的矢量表示的数据库。它旨在有效地存储和检索向量。这支持快速且可扩展的操作，例如语义搜索和相似度匹配。向量数据库使用专门的数据结构对向量进行索引，例如分层可导航小世界 (HNSW) 图或 K-Nearest 邻居 (KNN) 算法。这些数据结构允许快速进行最近邻搜索，从而可以在数据库中快速找到相似的向量。

语义搜索

语义搜索是一种通过了解查询的意图和上下文（而不仅仅是匹配关键字）来提高搜索结果相关性的技术。在技术术语中，语义搜索包括比较查询的向量表示形式和数据库中的文档，以找到最相关的匹配项。语义搜索可以使用不同的检索策略，包括但不限于：

- HNSW — 一种基于图形的数据结构，它以一种可以高效搜索最近邻的方式组织向量。
- KNN — 一种基于距离度量（例如余弦相似度）查找最接近查询向量的 K 个向量的算法。
- 余弦相似度 — 衡量两个非零向量之间相似度的度量，用于测量它们之间角度的余弦值。它通常用于语义搜索来比较高维空间中向量的方向。
- Locality-sensitive 哈希 (LSH) — 一种以高概率将相似向量与相同或附近的存储桶进行哈希处理的技术。这允许进行近似最近邻搜索，这可能比在高维空间中进行精确搜索更快。

影响RAG应用程序的源数据面临的挑战

开发最佳检索增强生成 (RAG) 应用程序的重大挑战之一在于所使用的原始数据或文档的性质。通常，企业使用为供人参考而创建的现有文档。这些文档通常包含超链接和图像屏幕截图，以增进理解。但是，由于摘录标记限制，这些元素会阻碍语义检索。这会导致检索器性能不佳。

以下是最佳 RAG 应用程序最常见的原始文档挑战：

- 缺少结构化格式和元数据-原始文档可能缺少清晰的章节标题、副标题或元数据。这使得识别和提取相关信息变得困难。例如，没有明确标题的长文档会使确定特定信息的上下文变得困难。
- 非正式且不一致的语言 — 原始文档通常包含非正式语言或不一致的术语。这可能会混淆RAG模型。例如，文档中未定义或法学硕士已知的缩写可能会在整个文档中使用。
- 冗余和冗余 — 原始文档可能很冗长，并且包含不必要或冗余的信息。这可能会使RAG模型不堪重负，从而导致响应不那么简洁和相关。示例包括多次重复相同信息的文档，或者包含相似或矛盾信息的多个文档。
- 模棱两可的术语和短语 — 原始文档可能包含模棱两可的术语或短语，这些术语或短语可能有多种解释。这种模棱两可可能导致RAG模型的误解和不准确的响应。例如，使用具有多种含义的术语的文档可能会得到与预期含义不一致的响应。
- 注入图形和超链接元素 — 包含图形和超链接信息的原始文档非常适合人类使用。但是，这些元素可能会消耗检索令牌限制。结果是摘录可能不完整。例如，图形和超链接 URLs 作为检索的一部分返回，这会耗尽检索标记，并且后续段落中的关键信息丢失。
- 缺乏特定领域的知识或上下文 — 原始文档可能缺乏准确生成所需的特定领域知识或上下文。这可能会限制 RAG 模型生成相关且准确的响应的能力。例如，文档引用了专门的概念，但没有提供上下文。这可能会导致在给定域中没有意义的响应。

尽管这份清单并不全面，但它为企业提供了一个起点，让他们思考哪些不起作用以及为什么。文档可能面临其中一个或多个难题。优化 RAG 应用程序的关键是使用一组符合编写优化检索的最佳实践的文档。

RAG 应用程序的文档最佳实践

开发成功的检索增强生成 (RAG) 应用程序需要仔细考虑各种与文档相关的因素，以优化其性能。本节中的最佳实践是根据与许多组织领导一起构建 RAG 系统的经验精心策划的。以下是一些关键的文档最佳实践，可提高 RAG 应用程序的有效性：

- 正确使用标题和副标题 — 使用清晰的标题和副标题整理内容可以提高可读性，并有助于 RAG 模型了解文档的结构。这种做法使模型能够更好地浏览和从文档中提取信息，从而提高生成的响应的质量。
- 确保编号是连续的-使用编号列表时，务必保持正确的编号，以免造成混淆。确保每个列表项按顺序编号，不要跳过数字。这有助于保持内容的清晰度和连贯性。
- 在@@ 列表项之间添加过渡 — 在项目符号列表或编号列表中的项目之间提供过渡有助于引导 LLM 浏览内容。例如，你可以使用 “完成第 2 步后，做...” 之类的短语来联系想法并改善信息流。
- 替换表格-避免使用表格。以多级项目符号列表或扁平语法格式化此信息。平面语法是在相同的层次结构级别上排列元素或项目，没有嵌套的从属级别。这些结构 LLMs 有助于消化信息。由于大多数索引文档都是从左向右读取的，因此扁平语法允许信息更连贯地跟踪，而无需引用其他维度。这种格式更有利于RAG应用程序，因为它以结构化且易于消化的方式呈现信息。
- 预处理图形信息以提高效率 — 多模态 LLMs 可以同时摄取图像和文本。降低图像的分辨率，删除多余的图像，并以文本格式描述图形元素的内容。这些措施改善了有意义的上下文，避免了不必要地消耗代币，并提高了RAG模型的可访问性。
- 为常见问题添加会话启动器 — 在解决常见问题或任务时，例如 “如何订购软件？”，添加会话启动器，让读者进入流程。例如，您可以添加 “如果您要订购软件，请按照以下步骤操作...”。这有助于创建高语义匹配，从而帮助法学硕士构建有凝聚力的响应。
- 向每个部分添加摘要- 在每个标题或副标题之后，添加该部分内容的简短摘要。这可以增加语义覆盖范围并强化关键点。这提高了嵌入空间内相似度搜索的准确性，从而提高了 RAG 应用程序的性能。如果文档供法学硕士学位和人类使用，或者需要表格和图形元素，则这特别有用。
- 消除歧义 — 文件应简洁明了。 LLMs 根据检索到的摘录生成响应，因此消除歧义有助于模型使用清晰而相关的信息。这样可以得到更准确、内容更丰富的回复。
- 定义缩写和设置上下文 — LLMs 受过大量互联网数据的训练，而且大多数时候，它们没有企业内部文档的上下文。因此，设置上下文、定义缩写以及避免或定义公司特定的术语有助于法学硕士了解您的企业数据。这有助于法学硕士更准确地回答问题，并有助于防止出现幻觉。
- 将@@ 大型文档重组为较小的文档，以实现高效的标记和索引 — 避免为包含多个子主题的大型文档编制索引。可以考虑将大型文档分成标题清晰的较小、独立的文档。这改进了索引和标记。

常见问题解答

为什么 RAG 应用程序优化文档很重要？

原始文档通常是在不考虑高级人工智能系统（例如检索增强生成 (RAG) 应用程序）的要求的情况下编写的。通过遵循最佳实践来优化文档，可以为模型提供结构化、明确和相关的信息，从而显著提高 RAG 应用程序的性能和准确性。

原始文档有哪些常见的难题会阻碍 RAG 的性能？

一些关键挑战包括缺乏结构化格式和元数据、语言非正式或不一致、冗长和冗余、术语和短语含糊不清、包含超链接元素以及缺乏特定领域的上下文。这些问题可能会混淆RAG模型，并导致不准确或不相关的响应。有关更多信息，请参阅本指南中的源数据中影响 RAG 应用程序的难题。

如何使用标题和副标题来提高 RAG 的性能？

清晰的标题和副标题可帮助 RAG 模型了解内容的结构和上下文。这使他们能够更好地浏览和从文档中提取相关信息，并提高生成的回复的质量。有关更多信息，请参阅本指南中的 RAG 应用程序文档最佳实践。

为什么建议用平面语法替换表格信息？

RAG 模型解释表格可能很困难，因为它们需要对二维结构的理解。以扁平语法或项目符号列表呈现表格信息有助于模型更轻松地处理信息，从而提高性能。有关更多信息，请参阅本指南中的 RAG 应用程序文档最佳实践。

添加摘要如何提高 RAG 性能？

在每个部分或小节的开头加入简洁的摘要可以增加语义覆盖面并强化要点。这提高了嵌入空间内相似度搜索的准确性，从而最终提高了 RAG 应用程序的性能。有关更多信息，请参阅本指南中的 RAG 应用程序文档最佳实践。

为什么定义缩写和设置上下文很重要？ LLMs

LLMs 接受了有关广泛数据的培训，但他们缺乏企业特定缩写词或术语的背景信息。定义缩写和提供上下文有助于更准确地 LLMs 理解和回应。这可以帮助防止出现幻觉或误解。有关更多信息，请参阅本指南中的 RAG 应用程序文档最佳实践。

后续步骤和资源

本指南讨论了 RAG 应用程序面临的一系列文档级挑战以及缓解这些挑战的最佳实践。这些学习是通过与行业领导者的访谈和讨论而精心策划的，并得到了企业用例的支持。

要开始针对RAG应用程序优化您的文档，我们建议对您的现有文档进行审计。确定对 RAG 应用程序构成挑战的领域。例子包括结构不足、语言模棱两可或过度使用图形元素。对经常访问或对业务运营至关重要的文档进行优先排序。与主题专家合作，实施本指南中的最佳实践。确保使用清晰的标题、简洁的语言和上下文设置元素对文档进行重组。对于新文档，应制定指导方针和模板，确保一致性并帮助作者遵守最佳实践。此外，可以考虑投资于可以自动执行文档优化过程各个方面的工具或服务，例如使用生成式人工智能来重构文档。通过采取积极主动的方法进行文档优化，您可以充分发挥 RAG 应用程序的潜力，并在整个组织中推动更准确、更有洞察力的结果。

以下资源可以帮助您在组织中了解和构建 RAG 应用程序。

资源

AWS 文档

- 为 RAG 用例选择 AWS 矢量数据库（AWS 规范性指导）
- 使用 Terraform 和 Amazon AWS Bedrock（规范性指南）部署 RAG 用例AWS
- 使用 RAG 和 ReAct 提示（规范性指导）开发基于人工智能聊天的高级生成式助手AWS
- 使用亚马逊 Bedrock 知识库检索数据并生成 AI 响应（亚马逊 Bedrock 文档）
- 检索增强生成（Amazon SageMaker AI 文档）
- 检索增强生成选项和架构 AWS（AWS 规范性指导）

其他 AWS 资源

- 使用@@ 高级 RAG 和 Amazon Bedrock 创建多式联运助手（AWS 博客文章）

文档历史记录

下表介绍了本指南的一些重要更改。如果您希望收到有关未来更新的通知，可以订阅 RSS 源。

变更	说明	日期
初次发布	—	2025 年 7 月 24 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。