0001年1月1日 2 分钟阅读

当然，以下是修改后的技术中文内容，准备发布：

大模型 RAG（检索增强生成）应用演进之路与技术解析：

大型语言模型（LLM）的检索增强生成（RAG）应用，其演进历程是一部不断突破局限、追求卓越性能与灵活性的奋斗史。从最初的朴素检索到如今的模块化智能系统，RAG 应用在每个阶段都取得了显著进展。本文将深入剖析这一演进过程，揭示其技术细节与未来趋势。[1]

核心论断：

RAG 应用的演进，可概括为从 Naive RAG 的基础检索生成，到 Advanced RAG 的精细优化，再到 Modular RAG 的灵活组合这三大阶段。每个阶段均致力于提升检索精度、生成内容相关性及系统适应性，以应对日益复杂的应用场景。[2]

1. 朴素 RAG（Naive RAG）：奠基石

核心框架：索引、检索、生成三大基石，构建了从外部知识库获取信息并生成答案的基本流程。

技术实现细节：

索引过程：离线执行，原始文档经过清洗与分块处理，随后通过 Embedding 模型将文本块转化为向量，并构建向量索引。

检索过程：用户 Query 的 Embedding 向量与文档块的 Embedding 向量进行相似度计算，遴选出相似度最高的若干文档块作为增强上下文。

生成过程：原始 Query 与检索到的相关文档块合并为 Prompt，输入至大型语言模型，生成最终答案。[3]

挑战：

索引质量：关键词易被大量无效信息淹没，导致检索效果欠佳；内容块的语义信息易受分割方式影响，关键信息可能遗失。

检索精度：用户 Query 的表达方式与知识库索引内容可能存在偏差，导致检索结果不准确；单纯依赖向量相似度计算，缺乏对 Query 与文档间深层关系的理解。

生成幻觉：当检索不到相关知识或检索知识质量较差时，大型语言模型可能自主生成不准确或不相关的内容，即产生“幻觉”。[4]

目录

文章信息

字数

阅读时间

发布时间

实际应用案例：

智能客服：在简单的客户咨询场景中，通过检索预设知识库，为用户提供常见问题解答。

局限性：无法处理复杂的语义关系和多轮对话，易受噪声信息干扰，生成内容缺乏深度与创新性。

2. 高级 RAG（Advanced RAG）：精雕细琢

核心目标：通过在检索前、检索中、检索后增加优化策略，提升检索精度与生成内容的相关性。

技术实现细节：

检索前优化（Pre-Retrieval）：

索引优化：

索引降噪：依据业务特点，剔除索引数据中的无效成分，凸显核心知识。例如，从“如何在 github.com 下载源代码”中提取“下载源代码”、“github”等核心信息。

知识切分：训练专门的语义理解小模型，将较长文本按照语义内聚性切分为更小的块。例如，使用滑动窗口技术，确保上下文信息的完整性。

元数据增强：增加内容摘要、时间戳、关键词等附加信息，丰富知识库，提高检索准确性。例如，添加章节引用、关键信息、小节标题等作为元数据。

查询改写（Query Rewriting）：

Query 分解：将复杂问题分解为多个子问题，分别进行检索。例如，将“请详细介绍 RAG”分解为“RAG 的概念是什么？”、“RAG 的原理是怎样的？“等。

Query 扩展：将问题转化为多种不同的问法，提高检索覆盖率。例如，使用 LLM 模型对用户初始查询进行改写生成多个查询，再进行多路搜索召回。

检索时优化（Retrieval）：

嵌入模型微调（Embedding Fine-tuning）：将 Embedding 模型定制为特定领域的上下文，提高语义相似度计算的准确性。例如，在专业术语浓度高的场景中，对嵌入模型进行微调。

检索后优化（Post-Retrieval）：

提示压缩（Prompt Compression）：删除检索到的内容块中的无关信息，凸显重要上下文，减少噪声干扰。

重新排序（Re-ranking）：使用专门的排序模型，重新计算上下文的相关性得分，提高生成内容的质量。例如，使用 Cohere 模型考虑查询意图、词汇多重语义、用户历史行为等特征。

挑战：

数据质量：依赖于高质量的原始数据和文档，数据质量欠佳将严重影响优化效果，即“Garbage in, garbage out”。

结构化数据处理：难以有效处理表格、数据库等结构化数据，无法捕捉实体间的复杂关系与层次结构。[5]

实际应用案例：

金融智能投研：通过优化检索策略，从海量研报和新闻资讯中提取关键信息，为投资者提供更精准的投资建议。

关键技术：

Query 改写：利用大型语言模型对用户 Query 进行语义分析和转换，使其更符合知识库的检索要求。例如，将“如何评价某公司的新产品”改写为“某公司新产品特点分析”、“某公司新产品市场前景”。

嵌入模型微调：使用特定领域的数据对 Embedding 模型进行微调，使其更好地捕捉领域内的语义信息。例如，在医疗领域，使用医学文献对 Embedding 模型进行训练，提高医学术语的识别能力。[6]

3. 模块化 RAG（Modular RAG）：运筹帷幄

核心目标：将 RAG 系统拆分为多个独立模块，实现更高的灵活性和可维护性，并根据不同应用场景和需求，灵活组合和配置这些模块。

框架：在高级 RAG 的基础上，增加编排（Orchestration）环节，实现对整个 RAG 流程的智能控制。

技术实现细节：

索引模块：

块优化（Chunk Optimization）：

滑动窗口（Sliding Window）：将文档分割为多个滑动窗口，每个窗口包含多个连续的句子，保证上下文的完整性。

增加元数据（Add Metadata）：为每个内容块增加页码、文件名、作者、时间戳等元数据，提高检索的准确性。

结构组织（Structural Organization）：

多级索引（Multi-Level Indexing）：创建多个索引，例如文档摘要索引和文档块索引，分步进行检索，提高效率。例如，先通过摘要过滤相关文档，再在相关组内搜索。

知识图谱（Knowledge Graph）：利用知识图谱对数据集建立索引，提取实体和实体之间的关系，构建全局性的知识网络。例如，提取实体（Entity）及实体之间的关系（Relationship），构建全局性优势。

预检索模块：

查询扩展（Query Expansion）：

多查询（Multi-Query）：将原始 Query 扩展成多个相似的 Query，并行执行检索。例如，借助提示工程通过大型语言模型来扩展查询。

子查询（Sub-Query）：将原始 Query 分解为多个子问题，分别进行检索。例如，将"请详细且全面地介绍 RAG"分解为"RAG 的概念是什么？"、"为什么会产生 RAG？"等。

查询转换（Query Transformation）：

查询重写（Query Rewriting）：利用大型语言模型重新表述问题，使其更符合知识库的检索要求。例如，在多轮对话中，将历史信息和用户提问一并交给 LLM 大模型进行重新表述。

假设文档嵌入（Hypothetical Document Embeddings, HYDE）：让大型语言模型生成一个"假设"答案，将其和问题一起进行检索。

查询构建（Query Construction）：

Text-to-SQL：将自然语言 Query 转化为 SQL 语句，进行数据库查询。例如，在 ChatBI 场景下，将用户 Query 转化为 SQL 语句。

检索模块：

检索器选择（Retriever Selection）：根据不同的 Query 选择不同的检索器，例如稀疏检索器和密集检索器。

检索器微调（Retriever Fine-tuning）：使用特定领域的数据对检索器进行微调，提高检索的准确性。

检索后模块：

重排序（Re-ranking）：使用专门的排序模型，重新计算上下文的相关性得分。[7]

压缩（Compression）：删除无关内容，突出重要上下文。

选择（Selection）：移除无关的文档块。例如，使用 LLM-Critique 通过 LLM 批评机制，过滤掉相关性不高的文档。

生成模块：

生成器微调（Generator Fine-tuning）：使用指令微调和强化学习等技术，优化生成器的性能。

验证（Verification）：使用知识库验证和基于模型的验证等技术，验证生成内容的准确性。

编排模块：

路由（Routing）：为每个 Query 选择最合适的处理管道。例如，在索引环节引入多重索引技术后，使用多级路由机制，根据 Query 引导至最合适的父级索引。

调度（Scheduling）：控制 RAG 流程的执行顺序和条件。例如，通过规则判断、知识图谱判断、Agentic-Rag 等方式进行调

度。

融合（Fusion）：合并多个检索结果，生成最终的答案。例如，使用 LLM 融合和互反排名融合等技术。

实际应用案例：

智能投顾：根据客户的风险偏好和投资目标，从多个知识库中检索相关信息，并生成个性化的投资建议。

关键技术：

多级索引：通过构建多层索引结构，实现对大规模数据的快速检索。例如，先通过文档类型索引过滤出相关文档，再通过关键词索引定位到具体的段落。

知识图谱构建：利用知识图谱技术，将非结构化的文本数据转化为结构化的知识表示，提高检索的准确性和效率。例如，利用知识图谱对数据集建立索引，提取实体（Entity）以及实体之间的关系（Relationship），构建全局性的优势。[8]

4. 未来展望

伴随技术的持续演进，RAG 应用的未来发展趋势将聚焦于以下几个维度：

多模态 RAG：将 RAG 的能力拓展至文本之外的广阔领域，如图像、音频、视频等。通过集成多模态数据，RAG 系统可访问更广泛的源材料，实现文本与视觉信息的深度融合。[9]

GraphRAG：运用图数据库存储与检索知识，更有效地处理实体间的复杂关系，实现更深层次的推理与问答。

Agentic RAG：将 RAG 与智能代理相结合，实现自主学习、自主决策与自主执行，构建更智能化的 RAG 系统。Agentic RAG 系统能够依据用户需求，自动选择合适的知识库、检索策略与生成模型，并进行迭代优化，以提供最佳答案。[10]

通过对 RAG 应用演进之路的系统梳理，我们清晰地看到，RAG 技术正朝着智能化、个性化与多模态方向加速发展，为各行各业的应用带来前所未有的可能性。

多模态RAG 多模态RAG技术将RAG的能力拓展到了文本之外的更广阔领域 通过集成图像 视频等其他模态数据 RAG系统可以访问更加广泛的源材料 实现文本和视 2024年RAG：回顾与展望 - 知乎专栏 ↩

自然语言处理 大型语言模型的 检索 增强 生成 技术 综述 从 Naive RAG 到 Modular RAG 的发展与应用 最新发布 深度解读 RAG 技术发展历程：从基础 Naive RAG 到高级 Advanced，再到模块化Modular RAG的全面升级 ↩

在 RAG 的演进过程中 出现了多种形态 从最初的朴素 RAG 到高级 RAG 再到模块化 RAG 其核心链路始终围绕三大步骤 索引 检索和内容生成 高级 RAG 在检索环节前后增加了更多处理 而模块化 RAG 则将这三大步骤进一步细化为五个不同的阶段 使每个部分都能更专注地进行理论和技术突破 例如 在索引部分 针对分块 chunk 的优化成为许多论文和框架的研究重点 基于模块化 RAG 可以衍生出适用于不同场景的多种 RAG 形态 此外 还有与组织编排相关的 graph RAG 利用知识图谱 knowledge graph 进行知识提取 实体与关系连接等操作 这些内容也都在相关图中得到了概括
复杂场景下的RAG架构演进：跨模态知识联邦与统一语义推理实践-36氪

检索环节 在使用用户原始query做检索时 由于语义的复杂性以及需要用到专业词汇时 无法准确地理解用户的真实需求 导致用户query和知识索引不能很好匹配 同时 查询Query和文档块之间的语义相似度并非总是高度一致 仅依靠相似度计算进行检索 缺乏对查询Query与文档间关系的深度探究 导致检索质量不高 另外 朴素rag会将所有检索到的块直接输入到大模型 但是过多的冗余和噪声信息会干扰大模型对关键信息的识别 增加生成错误
Rag系统的发展历程，从朴素、高级到模块化 - 腾讯新闻

同时 对每个环节的能力做了模块化分割 比如索引环节包括了块优化 chunk optimization 结构组织 structural organization 比如预检索环节包括了查询转化 Query Transformation 查询扩展 Query Expansion 查询构建 Query Construction 等等
Rag系统的发展历程，从朴素、高级到模块化 - 腾讯新闻

未来的趋势可能是将RAG 和agent 结合起来 形成更加综合的系统 RAG 通过将外部知识库的检索与生成模型相结合 显著提升了生成内容的时效性 准确性 显著
RAG 的未来，走向繁荣、重塑还是消亡？ - 36氪

Query改写 Query改写就是把用户原始的问题转换成适合知识库检索的问题 从而提高检索的精准程度 Query改写的方法主要有Query分解和Query扩展 Query分解是将一个问题分解为多个子问题 Query扩展则是将一个问题转化成多种不同的问法
Rag系统的发展历程，从朴素、高级到模块化 - 腾讯新闻

2 检索 对用户输入的Query问题 使用相同的embedding模型 计算Query嵌入和文档块嵌入之间的向量相似度 然后选择相似度最高的前N个文档块作为当前问题的增强上下文信息
Rag系统的发展历程，从朴素、高级到模块化 - 腾讯新闻

尽管高级RAG有所改进 但是仍然与实际应用与需求之间存在差距 比如数据源的质量问题 Garbage in Garbage out 如果本身原始数据或文档的质量就很差 最终生成的结果也不会好
Rag系统的发展历程，从朴素、高级到模块化 - 腾讯新闻

知识图谱判断 从知识图谱中提取与问题相关的信息 构建一个由逻辑上相互关联的节点组成的推理链 每个节点都承载着解决问题的关键信息 可以分别进行信息检索和内容生成

Rag系统的发展历程，从朴素、高级到模块化 - 腾讯新闻 ↩

分享这篇文章