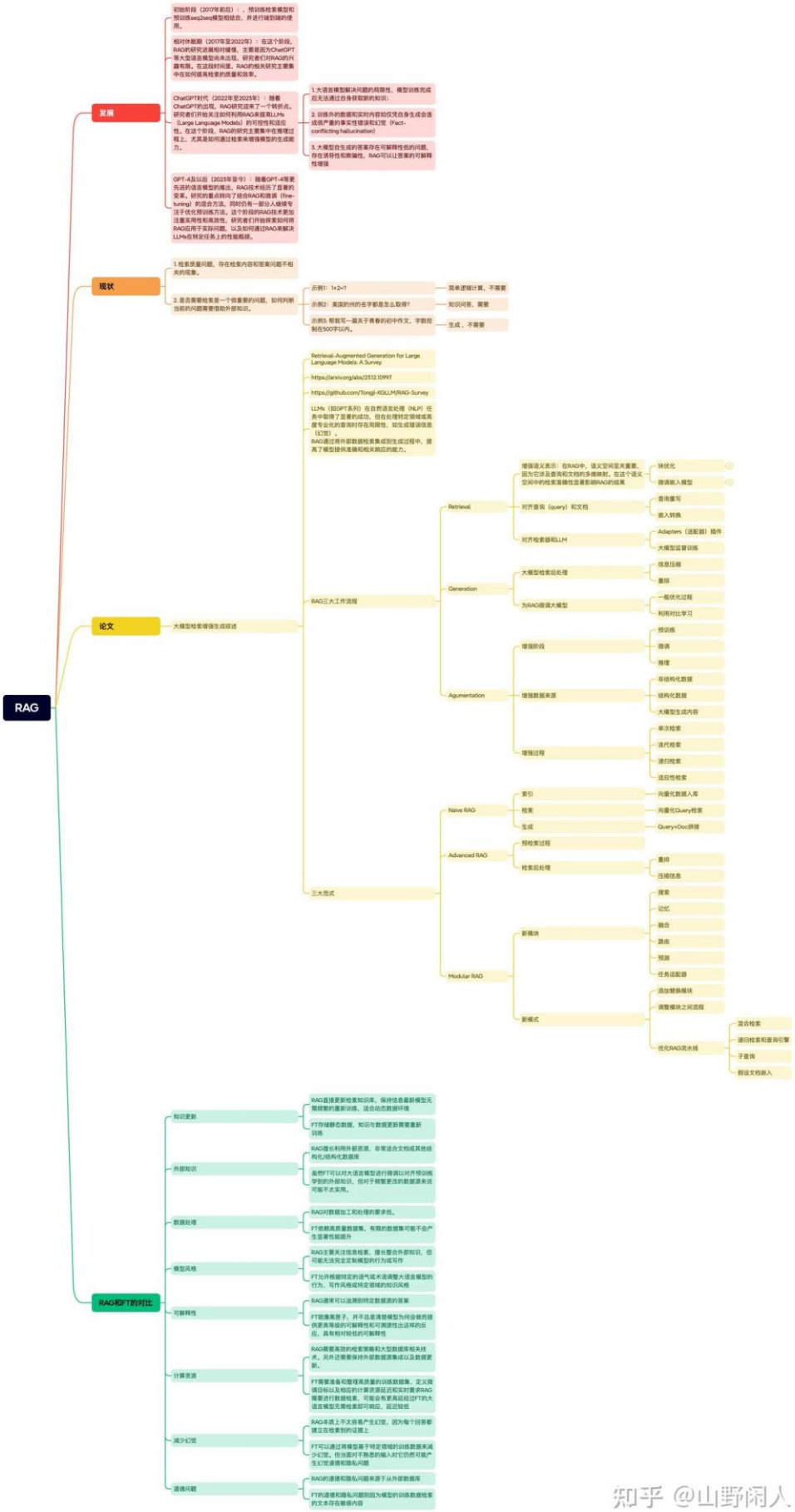
RAG
发展
初始阶段（2017年前后）：，预训练检索模型和预训练seq2seq模型相结合，并进行微到端的使用。
相对休眠期（2017年至2022年）：在这个阶段，RAG的研究进展相对缓慢，主要是因为ChatGPT等大型语言模型尚未出现，研究者们对RAG的兴趣有限。在这段时间里，RAG的相关研究主要集中在如何提高检索的质量和效率。
ChatGPT时代（2022年至2023年）：随着ChatGPT的出现，RAG研究迎来了一个转折点。研究者们开始关注如何利用RAG来提高LLMs（Large Language Models）的可控性和适应性。在这个阶段，RAG的研究主要集中在推理过程上，尤其是如何通过检索来增强模型的生成能力。
GPT-4及以后（2023年至今）：随着GPT-4等更先进的语言模型的推出，RAG技术经历了显著的变革。研究的重点转向了结合RAG和微调（fine-tuning）的混合方法。同时仍有一部分人继续专注于优化预训练方法。这个阶段的RAG技术更加注重实用性和高效性。研究者们开始探索如何将RAG应用于实际问题。以及如何通过RAG来解决LLMs在特定任务上的性能瓶颈。
1.大语言模型解决问题的局限性。模型训练完成后无法通过自身获取新的知识；
2. 训练外的数据和实时内容如仅凭自身生成会造成很严重的事实性错误和幻觉（Fact-conflicting hallucination）
3. 大模型自生成的答案存在可解释性低的问题，存在诱导性和欺骗性，RAG可以让答案的可解释性增强
现状
1.检索质量问题，存在检索内容和答案问题不相关的现象。
2. 是否需要检索是一个很重要的问题，如何判断当前的问题需要借助外部知识。
示例1：1+2=?
简单逻辑计算，不需要
示例2：美国的州的名字都是怎么取得?
知识问答，需要
示例3. 帮我写一篇关于青春的初中作文，字数控制在500字以内。
生成，不需要
论文
大模型检索增强生成综述
Retrieval-Augmented Generation for Large Language Models: A Survey
https://arxiv.org/abs/2312.10997
https://github.com/Tongji-KGLLM/RAG-Survey
LLMs（如GPT系列）在自然语言处理（NLP）任务中取得了显著的成功。但在处理特定领域或高度专业化的查询时存在局限性。如生成错误信息（幻觉）。
RAG通过将外部数据检索集成到生成过程中，提高了模型提供准确和相关响应的能力。
RAG三大工作流程
Retrieval
增强语义表示：在RAG中，语义空间至关重要。因为它涉及查询和文档的多维映射。在这个语义空间中的检索准确性显著影响RAG的结果
块优化
微调嵌入模型
对齐查询（query）和文档
查询重写
嵌入转换
对齐检索器和LLM
Adapters（适配器）插件
大模型监督训练
Generation
大模型检索后处理
信息压缩
重排
为RAG微调大模型
一般优化过程
利用对比学习
Agumentation
增强阶段
预训练
微调
推理
增强数据来源
非结构化数据
结构化数据
大模型生成内容
增强过程
单次检索
迭代检索
递归检索
适应性检索
三大范式
Naive RAG
索引
向量化数据入库
检索
向量化Query检索
生成
Query+Doc拼接
Advanced RAG
预检索过程
检索后处理
重排
压缩信息
Modular RAG
新模块
搜索
记忆
融合
路由
预测
任务适配器
新模式
添加替换模块
调整模块之间流程
优化RAG流水线
混合检索
递归检索和查询引擎
子查询
假设文档嵌入
RAG和FT的对比
知识更新
RAG直接更新检索知识库，保持信息最新模型无需频繁的重新训练，适合动态数据环境
FT存储静态数据，知识与数据更新需要重新训练
外部知识
RAG擅长利用外部资源，非常适合文档或其他结构化/结构化数据库
虽然FT可以对大语言模型进行微调以对齐预训练学到的外部知识，但对于频繁更改的数据源来说可能不太实用。
数据处理
RAG对数据加工和处理的要求低。
FT依赖高质量数据集，有限的数据集可能不会产生显著性能提升
模型风格
RAG主要关注信息检索，擅长整合外部知识，但可能无法完全定制模型的行为或写作
FT允许根据特定的语气或术语调整大语言模型的行为，写作风格或特定领域的知识风格
可解释性
RAG通常可以追溯到特定数据源的答案
FT就像黑匣子，并不总是清楚模型为何会做出提供更高等级的可解释性和可溯源性出这样的反应，具有相对较低的可解释性
计算资源
RAG需要高效的检索策略和大型数据库相关技术。另外还需要保持外部数据源集成以及数据更新。
FT需要准备和整理高质量的训练数据集，定义微调目标以及相应的计算资源延迟和实时要求RAG需要进行数据检索，可能会有更高延迟的FT的大语言模型无需检索即可响应，延迟较低
减少幻觉
RAG本质上不太容易产生幻觉，因为每个回答都建立在检索到的证据上
FT可以通过将模型基于特定领域的训练数据来减少幻觉，但当面对不熟悉的输入时它仍然可能产生幻觉道德和隐私问题
道德问题
RAG的道德和隐私问题来源于从外部数据库
FT的道德和隐私问题则因为模型的训练数据检索的文本存在敏感内容

TLTR

本文综述了检索增强生成（RAG）技术在提升大型语言模型（LLMs）性能方面的进展。RAG 通过结合 LLMs 的内在知识和外部数据库的非参数化数据，提高了模型在知识密集型任务中的准确性和可信度。文章详细讨论了 RAG 的三个发展阶段：初级、高级和模块化 RAG，并分析了其在检索、生成和增强技术方面的关键组件。此外，文章还探讨了 RAG 在多模态领域的扩展，以及如何通过改进评估方法来适应其不断发展的应用范围。最后，文章强调了 RAG 生态系统的增长，以及为了充分利用 RAG 技术，需要进一步的研究和开发。

摘要

大型语言模型（LLMs）如 GPT 系列和 LLama 系列在自然语言处理领域取得了显著的成功，但它们面临着幻觉、过时知识和不透明、不可追溯的推理过程等挑战。检索增强生成（RAG）通过整合外部数据库的知识，作为一种有前景的解决方案，增强了模型的准确性和可信度，特别是对于知识密集型任务。RAG 将 LLMs 的内在知识与外部数据库的庞大、动态的知识库相结合。这篇综述论文详细考察了 RAG 范式的进展，包括 Naive RAG、Advanced RAG 和 Modular RAG。它仔细审查了 RAG 框架的三个组成部分：检索、生成和增强技术。论文强调了这些关键组件中嵌入的最新技术，提供了对 RAG 系统进展的深刻理解。此外，本文介绍了评估 RAG 模型的指标和基准，以及最新的评估框架。最后，论文描绘了未来的研究方向，包括识别挑战、扩展多模态性和 RAG 基础设施及其生态系统的发展。

引言

LLMs 如 GPT 系列和 LLama 系列在各种基准测试中表现出色，但它们在处理特定领域或高度专业化的查询时存在显著局限性。一个常见问题是生成不正确的信息，或称为“幻觉”。这些缺陷强调了在没有额外信息的情况下，将 LLMs 作为黑盒解决方案部署在现实生产环境中的不切实际。RAG 通过将外部数据检索整合到生成过程中，从而提高了模型提供准确和相关响应的能力。RAG 涉及一个初始检索步骤，其中查询外部数据源以获取相关信息，然后再 LLM 进行回答问题或生成文本。这个过程不仅为后续的生成阶段提供信息，而且确保响应基于检索到的可解释性，从而显著提高了输出的准确性和相关性。

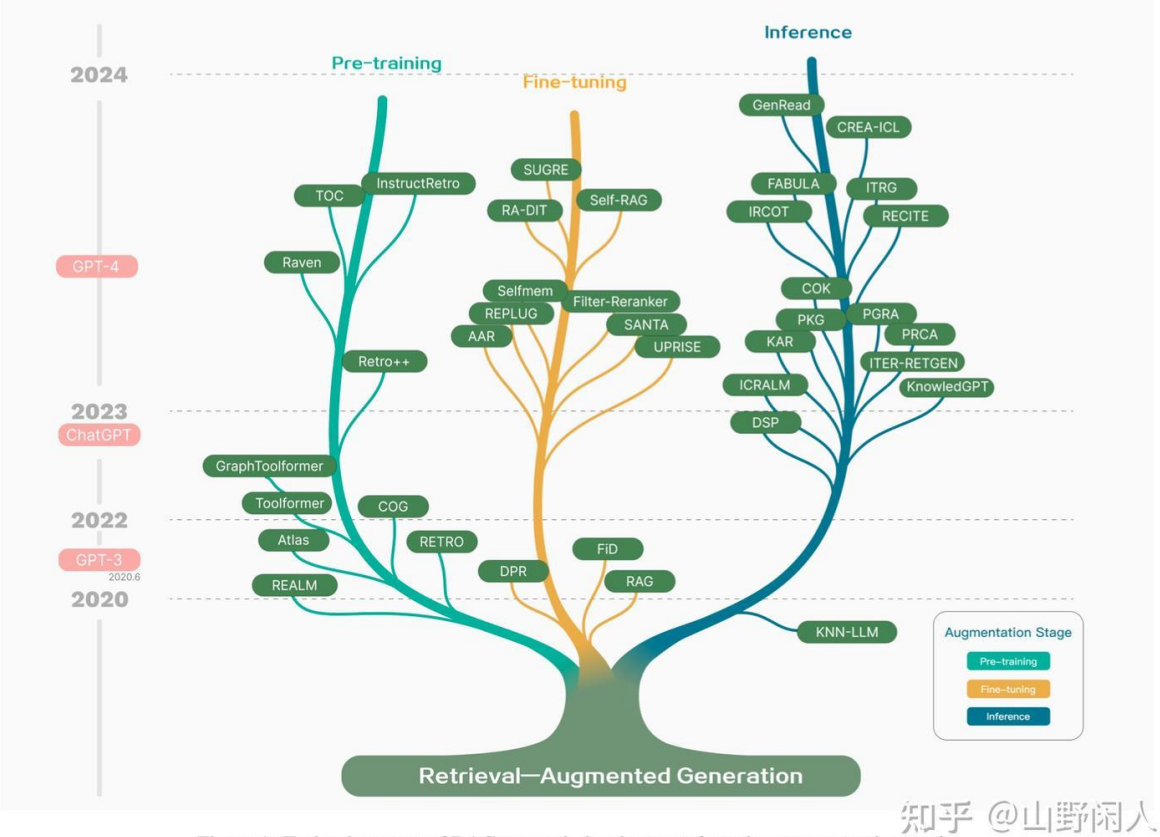

Figure 1: Technology tree of RAG research development featuring representative works

图 1 RAG 的四个发展阶段

1. **初始阶段（2017 年前后）**：
   RAG 的研究主要集中在如何通过预训练模型（Pre-Training Models, PTM）来增强语言模型的知识。这一时期的工作主要是探索如何将额外的知识融入到语言模型中，以提高其性能。这个阶段的研究为 RAG 奠定了基础，主要集中在优化预训练方法上。

2. **相对休眠期（2017 年至 2022 年）**：
   在这个阶段，RAG 的研究进展相对缓慢，主要是因为 ChatGPT 等大型语言模型尚未出现，研究者们对 RAG 的兴趣有限。在这段时间里，RAG 的相关研究主要集中在如何提高检索的质量和效率。

3. **ChatGPT 时代（2022 年至 2023 年）**：
   随着 ChatGPT 的出现，RAG 研究迎来了一个转折点。研究者们开始关注如何利用 RAG 来提高 LLMs（Large Language Models）的可控性和适应性。在这个阶段，RAG 的研究主要集中在推理过程上，尤其是如何通过检索来增强模型的生成能力。

4. **GPT-4 及以后（2023 年至今）**：
   随着 GPT-4 等更先进的语言模型的推出，RAG 技术经历了显著的变革。研究的重点转向了结合 RAG 和微调（fine-tuning）的混合方法，同时仍有一部分人继续专注于优化预训练方法。这个阶段的 RAG 技术更加注重实用性和高效性，研究者们开始探索如何将 RAG 应用于实际问题，以及如何通过 RAG 来解决 LLMs 在特定任务上的性能瓶颈。

RAG 的定义

RAG 的定义可以从其工作流程中总结出来。

如图二中的工作流示例：用户询问 ChatGPT 关于一个最近的高调事件（例如，OpenAI 首席执行官的突然解雇和复职），这引起了公众的广泛讨论。ChatGPT 作为最著名和广泛使用的 LLM，受到其预训练数据的限制，缺乏对最近事件的了解。RAG 通过从外部知识库检索最新的文档摘要来解决这一差距。这些文章与初始问题一起，被合并成一个丰富的提示，使 ChatGPT 能够综合出一个知情的回应。这个例子说明了 RAG 过程，展示了它通过实时信息检索增强模型响应的能力。

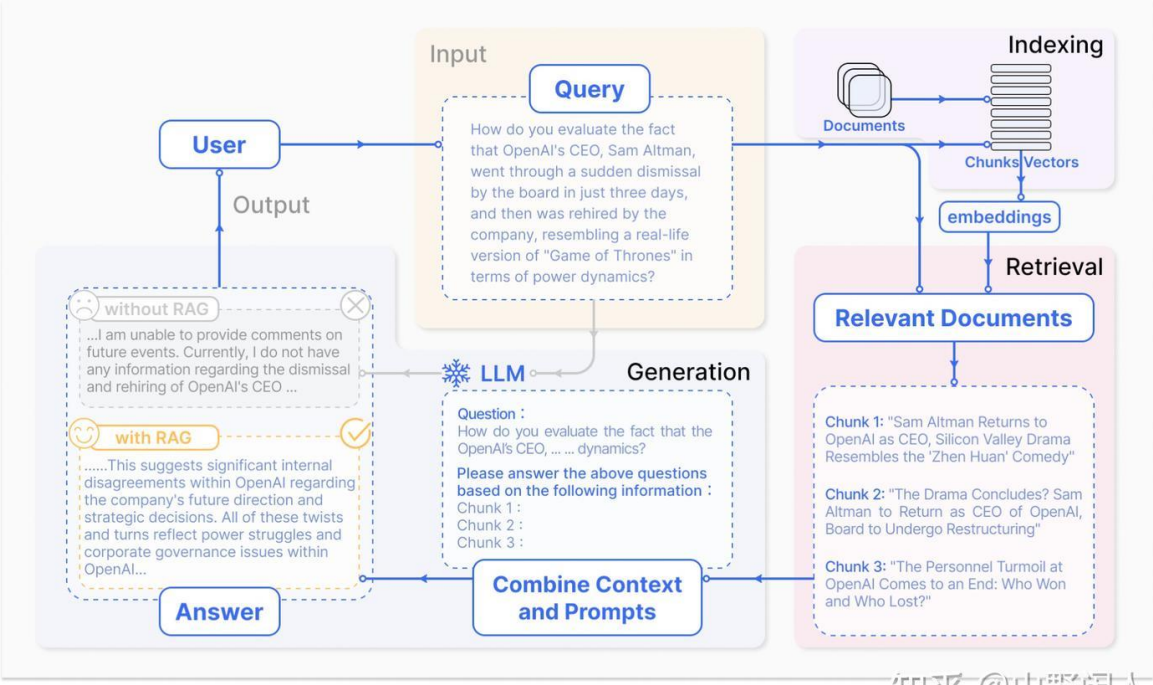

Figure 2: A representative instance of the RAG process applied to question answering

图 2 RAG 工作流

RAG 通过各种创新方法得到了丰富，这些方法解决了“检索什么”、“何时检索”和“如何使用检索到的信息”等关键问题。

RAG 是一种通过整合外部知识库来增强 LLMs 的范式。它采用协同方法，结合信息检索机制和上下文学习（ICL）来提升 LLM 的性能。在这个框架中，用户发起的查询通过搜索算法触发相关信息的检索。然后将这些信息编织到 LLM 的提示中，为生成过程提供额外的上下文。RAG 的关键优势在于它消除了对 LLM 进行任务特定应用的重新训练的需要。开发者可以添加一个外部知识库，丰富输入，从而提高模型输出的精确度。由于其高度实用性和低门槛，RAG 已成为 LLMs 系统中最受欢迎的架构之一，许多对话产品几乎完全建立在 RAG 之上。

RAG 工作流程包括三个关键步骤。首先，语料库被分割成离散的块，然后利用编码器模型构建这些块的向量索引。其次，RAG 根据查询和索引块的向量相似性识别和检索块。最后，模型根

据从检索到的块中获得的上下文信息合成响应。这些步骤构成了 RAG 过程的基本框架，支撑其信息检索和上下文感知生成能力。

索引-> 检索-> 生成， 三个工作流程。

RAG 框架

RAG 的研究范式一直在进化，可以总结为三大类：Naive RAG, Advanced RAG 和 Modular(模块化) RAG。RAG 不论是在效果上还是性价比上都比单独使用 LLM 要好得多，当然也有部分问题。Advanced RAG 和 Modular RAG 的发展的出现就是为了解决 Naive RAG 在使用上出现的缺点。

Naive RAG

朴素 RAG（naive RAG）研究范式代表了最早的方法论，它在 ChatGPT 广泛采用后不久就获得了显著地位。朴素 RAG 遵循一个传统的过程，包括索引、检索和生成。它也被描述为一个“检索-阅读”框架。

索引 索引过程是数据准备中的一个关键初始步骤，它在离线状态下进行，并涉及多个阶段。它从数据索引开始，原始数据被清洗和提取，各种文件格式如 PDF、HTML、Word 和 Markdown 被转换成标准化的纯文本。为了适应语言模型的上下文限制，这些文本随后被分割成更小、更易于管理的块，这个过程被称为分块。**这些块随后通过一个嵌入模型转换为向量表示，模型因其在推理效率和模型大小之间的平衡而被选择**。这有助于在检索阶段进行相似性比较。最后，创建一个索引来存储这些文本块及其向量嵌入作为键值对，这允许高效和可扩展的搜索能力。

检索 在收到用户查询后，系统使用与索引阶段相同的编码模型将输入转换为向量表示。然后计算查询向量与索引语料库中向量化块之间的相似度分数。系统优先检索与查询最相似的前 K 个块。这些块随后被用作扩展的上下文基础，以解决用户的请求。

生成 提出的查询和选定的文档被渲染成一个提示（Prompt），大型语言模型生成回答。模型的回答方法可能因任务特定标准而异，允许它要么利用其内在的参数知识，要么将其回应限制在提供的文档内包含的信息。在持续对话的情况下，任何现有的对话历史可以被整合到提示中，使模型能够有效地参与多轮对话互动。

朴素 RAG 的缺陷 朴素 RAG 在三个关键领域面临重大挑战：“检索”、“生成”和“增强”。 **检索质量提出了多种挑战，包括低精度，导致检索到的块与查询不一致，可能出现幻觉或空中掉物等问题。低召回率也会出现，导致无法检索到所有相关块，从而阻碍了 LLMs 构建全面回应的能力。过时的信息进一步加剧了问题，可能导致检索结果不准确。**

回答生成质量呈现幻觉挑战，模型生成的答案没有基于提供的上下文，以及模型输出中可能出现的无关上下文和潜在的有害或偏见问题。

增强过程在有效地将检索到的段落的上下文与当前生成任务整合方面也面临挑战，可能导致输出不连贯或不连贯。冗余和重复也是问题，特别是当多个检索到的段落包含相似信息时，会导

致生成的回应内容重复。

辨别多个检索到的段落对生成任务的重要性和相关性是另一个挑战，需要适当平衡每个段落的价值。此外，调和写作风格和语调的差异以确保输出的一致性至关重要。

最后，生成模型过度依赖增强信息的风险，可能导致输出仅重复检索到的内容，而没有提供新的价值或综合信息。

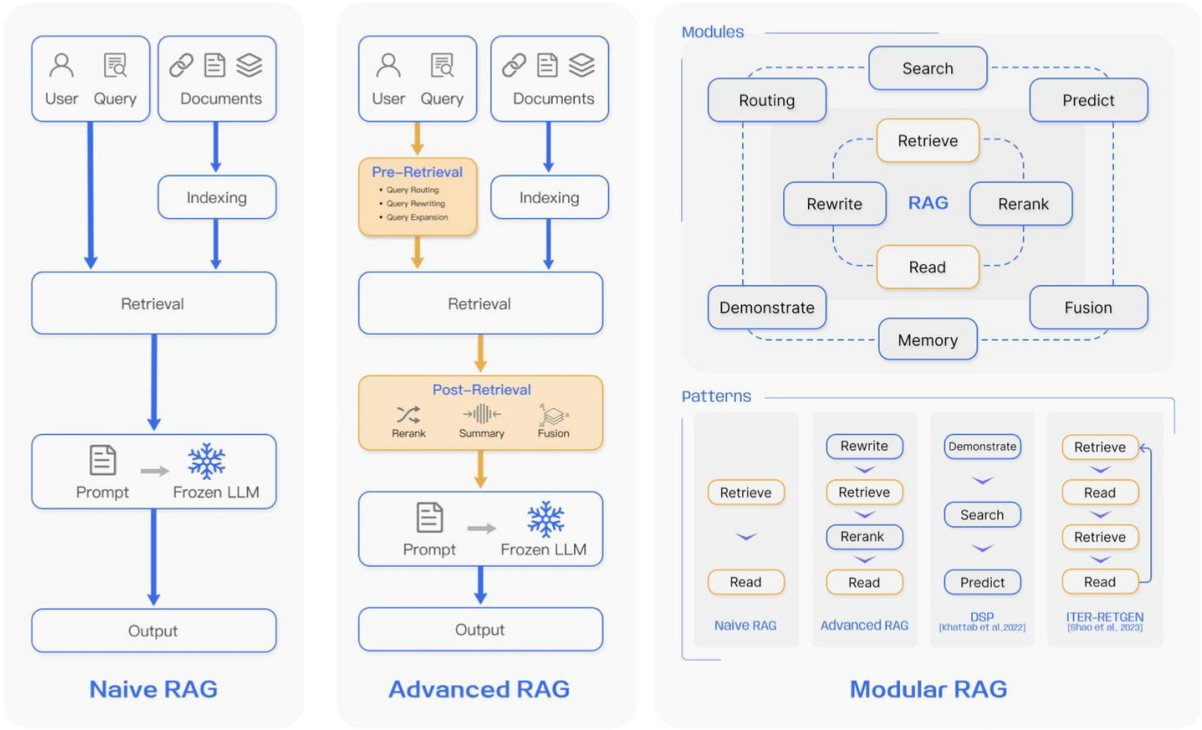

Figure 3: Comparison between the three paradigms of RAG

图 3 RAG 三种范式的对比

Advanced RAG

Advanced RAG 是为了解决 Naive RAG 的不足而开发的，它实现了预检索和后检索策略。为了解决 Naive RAG 在索引过程中遇到的挑战，Advanced RAG 改进了其索引方法，使用了滑动窗口、细粒度分割和元数据等技术。它还引入了各种方法来优化检索过程。

预检索过程（Pre-Retrieval Process）

优化数据索引（Optimizing Data Indexing）： 优化数据索引的目标是提高被索引内容的质量。这涉及五种主要策略：增强数据粒度、优化索引结构、添加元数据、对齐优化和混合检索。

- 增强数据粒度旨在提高文本的标准化、一致性、事实准确性和丰富上下文，以改善 RAG 系统的性能。这包括移除无关信息、消除实体和术语的歧义、确认事实准确性、保持上下文和更新过时文档。
- 优化索引结构涉及调整块的大小以捕获相关上下文，跨多个索引路径查询，并利用图结

构中的节点关系来捕获相关上下文。

- 添加元数据信息涉及将引用的元数据（如日期和目的）整合到块中以用于过滤，并整合章节和子部分的参考元数据以提高检索效率。
- 对齐优化通过在文档中引入“假设性问题”来解决文档之间的对齐问题和差异。
- 混合检索指通过多路召回的方式进行检索融合，例如 BM25 的 sparse retrieval 和深度学习模型的 dense retrieval 方式。

检索（Retrieval）： 在检索阶段，主要关注点是通过计算查询和块之间的相似度来识别适当的上下文。嵌入模型是这个过程的核心。在高级 RAG 中，嵌入模型有优化的潜力。

- 微调嵌入模型。微调嵌入模型显著影响 RAG 系统中检索内容的相关性。**这个过程涉及定制嵌入模型以增强特定领域背景下的检索相关性，特别是对于处理不断演变或罕见术语的专业领域。**例如，BGE 嵌入模型（BAAI，2023）是一个可以微调以优化检索相关性的高性能嵌入模型。微调训练数据可以使用像 GPT-3.5-turbo 这样的语言模型生成，以形成基于文档块的问题，然后用作微调对。
- 动态嵌入适应于单词使用的上下文，与静态嵌入不同，后者为每个单词使用单一向量。例如，**在 BERT 等 Transformers 模型中，同一个词可以根据周围的词有不同的嵌入。**OpenAI 的 embeddings-ada-02 模型是一个复杂的动态嵌入模型，能够捕捉上下文理解。

检索后处理（Post-Retrieval Process）： 从数据库检索到有价值的上下文后，关键是将其与查询合并作为 LLMs 的输入，同时解决上下文窗口限制带来的挑战。简单地一次性向 LLM 展示所有相关文档可能会超出上下文窗口限制，引入噪声，并妨碍对关键信息的关注。需要对检索到的内容进行额外处理以解决这些问题。

- Rerank。重新排名检索到的信息，将最相关内容重新定位到提示的边缘，是一个关键策略。这个概念已经在 LlamaIndex、LangChain 和 HayStack 等框架中实现。例如，多样性 rerank **优先考虑基于文档多样性的重新排序**，而 LostInTheMiddleRanker 则**交替将最佳文档放在上下文窗口的开头和结尾**。此外，cohereAI rerank、bge-rerank 和 LongLLMLingua 等方法**重新计算相关文本和查询之间的语义相似度，解决了解释基于向量的模拟搜索的语义相似度的挑战**。
- 提示压缩。研究表明，检索文档中的噪声对 RAG 性能有不利影响。在后处理中，重点是压缩无关上下文，突出关键段落，并减少整体上下文长度。例如，Selective Context 和 LLMLingua 等方法**利用小型语言模型计算提示的互信息或困惑度，估计元素的重要性**。Recomp 通过在不同粒度上训练压缩器来解决这个问题，而 Long Context 和“Walking in the Memory Maze”**设计了摘要技术，以增强 LLM 对关键信息的感知，特别是在处理广泛上下文时**。

模块化 RAG（Modular RAG）

模块化 RAG 结构与传统的朴素 RAG 框架不同，提供了更大的灵活性和适应性。它整合了各种方法来增强功能模块，例如加入**搜索模块进行相似性检索**，并在检索器中应用微调方法。重构的 RAG 模块和迭代方法已被开发出来以解决特定问题。**模块化 RAG 范式在 RAG 领域越来越成为常态，允许通过多个模块进行序列化流水线或端到端训练。三种 RAG 范式的比较在图 3 中展示**。然而，模块化 RAG 并非独立存在。高级 RAG 是模块化 RAG 的一种特殊形式，而朴素 RAG 本身是高级 RAG 的一个特例。这三种范式之间的关系是继承和发展。

新模块

- **搜索模块**：与朴素/高级 RAG 中的相似性检索不同，搜索模块针对特定场景进行定制，并结合了对额外语料库的直接搜索。这种整合是通过 LLM 生成的代码、SQL 或 Cypher 等查询语言以及其他自定义工具实现的。这些搜索的数据源可以包括搜索引擎、文本数据、表格数据和知识图谱。

- **记忆模块**：该模块利用 LLM 的记忆能力来指导检索。方法涉及识别与当前输入最相似的记忆。Selfmem 利用检索增强的生成器迭代地创建一个无界记忆池，结合“原始问题”和“双重问题”。通过使用一个利用自身输出来改进自身的检索增强生成模型，文本在推理过程中与数据分布更加对齐。因此，模型自己的输出被用来代替训练数据。

- **融合**：**RAG-Fusion 通过使用大模型技术对用户的原始 Query 和历史对话信息进行扩写（改写），生成 multi-queries，从而增强了传统搜索系统，解决了它们的局限性。**这种方法不仅捕捉用户寻求的显式信息，还揭示了更深层次的、变革性的知识。融合过程涉及原始查询和扩展查询的并行向量搜索，智能重排以优化结果，并将最佳结果与新查询配对。这种复杂方法确保搜索结果与用户的显式和隐式意图紧密对齐，从而发现更有洞察力和相关性的信息。

- **路由**：RAG 系统的检索过程利用了不同领域、语言和格式的多样化来源，这些来源可以根据情况交替或合并。查询路由决定了用户查询的后续动作，选项包括摘要、搜索特定数据库或将不同路径合并为单一响应。查询路由器还为查询选择合适的数据存储，可能包括向量存储、图数据库或关系数据库，或层次化的索引——例如，用于多文档存储的摘要索引和文档块向量索引。查询路由器的决策是通过 LLMs 调用预定义并执行的，它将查询定向到所选的索引。

- **预测**：它解决了检索内容中常见的冗余和噪声问题。该模块不直接从数据源检索，而是利用 LLM 生成必要的上下文。LLM 生成的内容比通过直接检索获得的内容更可能包含相关信息。

- **任务适配器**：该模块专注于将 RAG 适配到各种下游任务。UPRISE 自动化了从预构建的数据池中检索零样本任务输入的提示，从而提高了跨任务和模型的通用性。同时，PROMPTAGATOR 利用 LLM 作为少量样本查询生成器，并基于生成的数据创建特定任务的检索器。通过利用 LLM 的泛化能力，它使得能够用最少的示例开发特定任务的端到端检索器。

模块化 RAG 通过引入新的模块和方法，提供了一种灵活的方式来增强和定制 RAG 系统。这些模块包括搜索模块、记忆模块、融合、路由和预测，它们共同工作以提高检索的效率和质量。此外，任务适配器模块使得 RAG 能够适应不同的下游任务，提高了系统的通用性和实用性。这种模块化的方法允许研究者和开发者根据特定需求定制 RAG 系统，从而在各种应用场景中实现更好的性能。

新模式（New Pattern）

模块化 RAG 的组织结构高度适应性强，允许在 RAG 过程中替换或重新排列模块，以适应特定的问题背景。朴素 RAG 和高级 RAG 都可以被视为由一些固定模块组成。如图 3 所示，朴素 RAG 主要由“检索”和“阅读”模块组成。典型的高级 RAG 模式在朴素 RAG 的基础上增加了“重写”和“重排”模块。然而，总的来说，模块化 RAG 享有更大的多样性和灵活性。

当前研究主要探索两种组织范式。第一种涉及添加或替换模块，而第二种专注于调整模块之间的组织流程。这种灵活性使得 RAG 过程能够有效地解决广泛的任务。

添加或替换模块。引入或替换模块的策略涉及在保持检索-阅读过程的核心结构的同时，整合额外的模块以增强特定功能。RRR 模型引入了重写-检索-阅读过程，利用 LLM 的性能作为重写模块的强化学习激励。这使得重写器能够微调检索查询，从而提高阅读器的下游任务性能。

同样，可以在 Generate-Read 等方法中选择性地交换模块，其中 LLM 的生成模块取代了检索模块。Recite-Read 方法将外部检索转化为从模型权重中检索，要求 LLM 最初记忆特定任务的信息，然后产生能够处理知识密集型自然语言处理任务的输出。

调整模块之间的流程。在模块流程调整领域，重点是增强语言模型和检索模型之间的互动。DSP 框架将上下文学习系统视为一个明确的程序，而不是最终的任务提示，从而更有效地处理知识密集型任务。ITER-RETGEN 方法利用生成的内容指导检索，在检索-阅读-检索-阅读流程中迭代实施“检索增强生成”和“生成增强检索”。这种方法展示了一种创新的方式，即使用一个模块的输出来改善另一个模块的功能。

优化 RAG 流水线

检索过程的优化旨在提高 RAG 系统中信息的效率和质量。当前研究集中在整合多样化的搜索技术、精炼检索步骤、融入认知回溯、实施多功能查询策略和利用嵌入相似性。这些努力共同旨在在检索效率和 RAG 系统中上下文信息深度之间实现平衡。

混合搜索探索。RAG 系统通过智能整合各种技术（包括基于关键词的搜索、语义搜索和向量搜索）来优化其性能。这种方法利用每种方法的独特优势，以适应多样化的查询类型和信息需求，确保持续检索到高度相关和内容丰富的信息。混合搜索的使用作为检索策略的有力补充，从而增强了 RAG 流水线的整体效能。

递归检索和查询引擎。递归检索涉及在初始检索阶段获取较小的块以捕获关键的语义含义。随后，在过程的后期阶段向 LLM 提供包含更多上下文信息的较大块。这种两步检索方法有助于在效率和提供丰富上下文的响应之间取得平衡。

子查询。根据场景，可以采用各种查询策略，例如使用 LlamaIndex 等框架提供的查询引擎，利用树查询，使用向量查询，或执行块的简单顺序查询。

假设文档嵌入。HyDE 基于这样一个信念：生成的答案可能在嵌入空间中比直接查询更接近。使用 LLM，HyDE 为查询创建一个假设性文档（答案），嵌入这个文档，并使用结果嵌入来检索与假设文档相似的真实文档。这种方法不是基于查询寻求嵌入相似性，而是关注从一个答案到另一个答案的嵌入相似性。然而，它可能不会始终产生理想的结果，特别是当语言模型对主题不熟悉时，可能导致更多错误实例。

检索

在 RAG 的背景下，从数据源高效地检索相关文档至关重要。然而，创建一个熟练的检索器面临着重大挑战。本节将探讨三个基本问题：1）如何实现准确的语义表示？2）什么方法可以对齐查询和文档的语义空间？3）如何使检索器的输出与大型语言模型的偏好对齐？

增强语义表示

在 RAG 中，语义空间至关重要，因为它涉及查询和文档的多维映射。在这个语义空间中的检索准确性显著影响 RAG 的结果。本节将介绍两种构建准确语义空间的方法。

块优化 管理外部文档的初始步骤涉及将它们分解成较小的块以提取细粒度特征，然后嵌入这些特征以表示它们的语义。然而，嵌入过大或过小的文本块可能导致次优结果。因此，确定语料库中文档的最优块大小对于确保检索结果的准确性和相关性至关重要。

选择合适的分块策略需要仔细考虑几个关键因素，如索引内容的性质、嵌入模型及其最佳块大小、用户查询的预期长度和复杂性，以及特定应用对检索结果的利用。例如，分块模型的选择应基于内容的长度——无论它是长还是短。此外，不同的嵌入模型在不同块大小下表现出不同的性能特征。例如，sentence-transformer 在处理单个句子时表现更好，而 text-embedding-ada-002 在处理包含 256 或 512 个标记的块时表现出色。

此外，用户输入问题的长短和复杂性以及应用的具体需求（例如，语义搜索或问答）也会影响分块策略的选择。这个选择可以直接受到所选 LLMs 的标记限制的影响，需要调整块大小。实际上，获得精确的查询结果涉及灵活应用不同的分块策略。没有一种“通用最佳”策略，只有最适合特定上下文的策略。

当前 RAG 领域的研究探索了各种块优化技术，旨在提高检索效率和准确性。其中一种方法涉及使用滑动窗口技术，通过在多个检索过程中合并全局相关信息来实现分层检索。另一种策略，称为“small2big”方法，在初始搜索阶段使用小文本块，随后为语言模型提供更大的相关文本块进行处理。

抽象嵌入技术根据文档摘要（或摘要）优先进行前 K 检索，提供对整个文档上下文的全面理解。此外，元数据过滤技术利用文档元数据来增强过滤过程。一种创新的方法，图索引技术，将实体和关系转化为节点和连接，显著提高了相关性，特别是在多跳问题的背景下。

这些多样化方法的结合带来了显著的进步，提高了检索结果并改善了 RAG 的性能。

微调嵌入模型 一旦确定了合适的块大小，下一步关键步骤是使用嵌入模型将这些块和查询嵌入到语义空间中。嵌入的有效性至关重要，因为它影响模型表示语料库的能力。最近的研究引入了诸如 AnglE、Voyage、BGE 等著名的嵌入模型。这些模型已经在广泛的语料库上进行了预训练。然而，当应用于专业领域时，它们准确捕获特定领域信息的能力可能受到限制。

此外，针对下游任务的嵌入模型微调对于确保模型理解用户查询的内容相关性至关重要。没有微调的模型可能无法充分满足特定任务的要求。因此，微调嵌入模型对于下游应用至关重要。嵌入微调方法有两个主要范式。

领域知识微调。为了确保嵌入模型准确捕获特定领域的信息，利用特定领域的数据集进行微调至关重要。这个过程与标准的语言模型微调有所不同，主要在于涉及的数据集的性质。通常，嵌入模型微调的数据集包括三个主要元素：查询、语料库和相关文档。模型使用这些查询在语料库中识别相关文档。模型的有效性基于其对查询检索这些相关文档的能力来衡量。数据集构建、模型微调和评估阶段各自呈现不同的挑战。LlamaIndex 引入了一系列关键的类和功能，旨在增强嵌入模型微调工作流程，从而简化这些复杂的过程。通过策划一个充满领域知识的语料库，并利用提供的方法，可以巧妙地微调嵌入模型，使其与目标领域的特定要求紧密对齐。

针对下游任务的微调。针对下游任务的嵌入模型微调是提高模型性能的关键步骤。在利用 RAG 进行这些任务的领域中，出现了创新的方法，通过利用 LLMs 的能力来微调嵌入模型。例如，PROMPTTAGATOR 利用 LLM 作为少量样本查询生成器来创建特定任务的检索器，解决了在数据稀缺领域中的监督微调挑战。另一种方法，LLM-Embedder，利用 LLMs 为多个下游任务的数据生成奖励信号。检索器通过两种类型的监督信号进行微调：数据集的硬标签和 LLMs 的软奖励。这种双信号方法促进了更有效的微调过程，将嵌入模型定制到多样化的下游应用中。

虽然这些方法通过结合领域知识和特定任务的微调提高了语义表示，但检索器可能并不总是与某些大型语言模型（LLMs）完全兼容。为了解决这个问题，一些研究人员探索了使用来自 LLMs 的反馈直接监督微调过程。这种直接监督旨在使检索器更紧密地与 LLMs 对齐，从而提高下游任务的性能。更全面的讨论将在第 4.3 节中呈现。

对齐查询和文档

在 RAG 检索式增强生成应用的背景下，检索器可能使用单一的嵌入模型来编码查询和文档，或者为每个任务分别使用不同的模型。此外，用户的原始查询可能存在表述不精确和缺乏语义信息的问题。因此，将用户查询的语义空间与文档的语义空间对齐至关重要。本节介绍了两种旨在实现这种对齐的基本技术。

查询重写 查询重写是调整查询和文档语义的基本方法。Query2Doc 和 ITER-RETGEN 等方法利用 LLMs 通过结合原始查询和额外的指导来创建一个伪文档[Wangetal.,2023c, Shao et al., 2023]。HyDE 通过使用文本线索构建查询向量来生成一个“假设性”文档，捕捉关键模式[Gao et al., 2022]。RRR 引入了一个框架，它颠倒了传统的检索和阅读顺序，专注于查询重写[Ma et al., 2023a]。STEP-BACKPROMPTING 使 LLMs 能够基于高层次概念执行抽象推理和检索[Zheng et al.,

2023]。此外，多查询检索方法利用 LLMs 同时生成和执行多个搜索查询，这对于解决具有多个子问题的复杂问题特别有利。

嵌入转换 除了查询重写这样的广泛策略外，还存在更精细的技术，专门用于嵌入转换。 LlamaIndex [Liu, 2023]通过引入一个适配器模块来实现这一点，该模块可以在查询编码器之后集成。这个适配器促进了微调，从而优化了查询嵌入的表示，将它们映射到更接近预期任务的潜在空间。

SANTA [Li et al., 2023d]解决了将查询与结构化外部文档对齐的挑战，特别是处理结构化和非结构化数据之间的不一致性。它通过两种预训练策略增强了检索器对结构化信息的敏感性：首先，通过利用结构化和非结构化数据之间的内在对齐来指导结构感知预训练方案中的对比学习；其次，通过实施实体掩蔽预测。后者使用以实体为中心的掩蔽策略，鼓励语言模型预测并填充掩蔽实体，从而促进对结构化数据的更深入理解。

对齐检索器和 LLM

在 RAG 流程中，通过各种技术提高检索命中率并不一定能改善最终结果，因为检索到的文档可能不符合 LLMs 的具体要求。因此，本节介绍了两种旨在将检索器输出与 LLMs 偏好对齐的方法。

微调检索器 一些研究利用来自 LLMs 的反馈信号来完善检索模型。例如，AAR [Yu et al., 2023b]通过使用编码器-解码器架构为预训练的检索器引入了监督信号。这是通过识别 LM 的首选文档通过 FiD 交叉注意力分数实现的。随后，检索器通过硬负采样和标准交叉熵损失进行微调。最终，经过改进的检索器可以直接应用于增强未见目标 LMs，从而在目标任务中提高性能。此外，建议 LLMs 可能更倾向于关注可读性而非信息丰富的文档。

REPLUG [Shi et al., 2023]利用检索器和 LLM 计算检索文档的概率分布，然后通过计算 KL 散度进行监督训练。这种简单有效的训练方法通过使用 LM 作为监督信号，提高了检索模型的性能，消除了对特定交叉注意力机制的需求。

UPRISE [Cheng et al., 2023a]也采用冻结的 LLMs 来微调提示检索器。LLM 和检索器都接受提示输入对作为输入，并利用 LLM 提供的分数来监督检索器的训练，有效地将 LLM 视为数据集标签器。此外，Atlas [Izacard et al., 2022]提出了四种监督微调嵌入模型的方法：

- 注意力蒸馏（*Attention Distillation*）这种方法利用 LLM 在输出期间生成的交叉注意力分数来提炼模型的知识。
- EMDR2。通过使用期望最大化算法，这种方法将检索到的文档作为潜在变量来训练模型。
- 困惑度蒸馏（*Perplexity Distillation*）直接使用生成的标记的困惑度作为指标来训练模型。
- LOOP。这种方法提出了一种基于文档删除对 LLM 预测影响的新损失函数，提供了一种有效的训练策略，使模型更好地适应特定任务。

这些方法旨在改善检索器和 LLM 之间的协同作用，从而提高检索性能并更准确地响应用户查询。

Adapters（适配器）

适配器是一种在模型微调过程中帮助对齐的技术，它可以解决通过 API 集成功能或处理本地计算资源有限所带来的约束等挑战。一些方法选择加入外部适配器来协助这一过程。

PRCA（一种适配器训练方法）通过上下文提取阶段和奖励驱动阶段来训练适配器。然后使用基于标记的自回归策略来优化检索器的输出[Yang et al., 2023b]。标记过滤方法利用交叉注意力分数高效地过滤标记，只选择得分最高的输入标记[Berchansky et al., 2023]。RECOMP 引入了提取式和生成式压缩器用于摘要生成。这些压缩器要么选择相关的句子，要么合成文档信息，创建针对多文档查询的定制摘要[Xu et al., 2023a]。

此外，PKG 提出了一种创新的方法，通过指令性微调将知识整合到白盒模型中[Luo et al., 2023]。在这种方法中，检索器模块被直接替换，根据查询生成相关文档。这种方法有助于解决微调过程中遇到的困难，并提高模型性能。

生成（Generation）

在 RAG（检索式增强生成）中，生成器是一个关键组成部分，它负责将检索到的信息转换为连贯流畅的文本。与传统的语言模型不同，RAG 的生成器通过整合检索到的数据来提高准确性和相关性。**在 RAG 中，生成器的输入不仅包括典型的上下文信息，还包括通过检索器获取的相关文本片段。这种全面的输入使生成器能够深入理解问题的上下文，从而产生更具信息量和上下文相关性的回应。**

此外，生成器受到检索到的文本的指导，以确保生成的内容与获取的信息之间保持一致性。多样化的输入数据导致了在生成阶段的针对性努力，所有这些努力都旨在改进大型模型对来自查询和文档的输入数据的适应性。在接下来的小节中，我们将通过深入探讨检索后处理和微调的方面来介绍生成器。

使用 LLM 进行检索后处理

在不可调的 LLM 领域，许多研究依赖于像 GPT-4 [OpenAI, 2023]这样的成熟模型，利用它们全面的内部知识系统地综合来自各种文档的检索信息。然而，这些大型模型仍然存在挑战，包括上下文长度的限制和对冗余信息的敏感性。为了解决这些问题，某些研究工作转向了检索后处理。

检索后处理涉及对检索器从大型文档数据库中检索到的相关信息进行处理、过滤或优化。其主要目标是提高检索结果的质量，使其更紧密地与用户需求或后续任务对齐。它可以被视为对检索阶段获得的文档进行的再处理。检索后处理中的常见操作通常包括信息压缩和结果重排。

信息压缩

检索器擅长从庞大的知识库中检索相关信息，但在检索文档中管理大量信息是一个挑战。正在进行的研究旨在扩展大型语言模型的上下文长度以解决这个问题。然而，当前的大型模型仍然在上下文限制方面存在困难。因此，有时需要对信息进行压缩。信息压缩对于减少噪声、解决

上下文长度限制和增强生成效果至关重要。

PRCA 通过训练一个信息提取器来解决这个问题[Yang et al., 2023b]。在上下文提取阶段，当提供输入文本 Sinput 时，它能够产生一个输出序列 Cextracted，代表输入文档的压缩上下文。训练过程旨在最小化 Cextracted 和实际上下文 Ctruth 之间的差异。

同样，RECOMP 采用类似的方法，通过对比学习训练一个信息压缩器[Xu et al., 2023a]。每个训练数据点包括一个正面样本和五个负面样本，编码器在整个过程中使用对比损失进行训练[Karpukhin et al., 2020]。

另一项研究采取了不同的方法，旨在通过减少文档数量来提高模型答案的准确性。在[Ma et al., 2023b]的研究中，他们提出了“Filter-Reranker”范式，结合了 LLMs 和小型语言模型（SLMs）的优势。在这个范式中，SLMs 充当过滤器，而 LLMs 作为重新排序代理。研究表明，指导 LLMs 重新排列 SLMs 识别的具有挑战性的样本，可以在各种信息提取（IE）任务中显著提高性能。

重排

重排模型在优化检索器检索到的文档集方面起着关键作用。语言模型在引入额外上下文时经常面临性能下降，而重排有效地解决了这个问题。核心概念涉及重新排列文档记录，优先考虑最相关的项目，从而限制文档的总数。这不仅解决了检索过程中上下文窗口扩展的挑战，还提高了检索效率和响应速度。

重排模型在整个信息检索过程中扮演双重角色，既作为优化器，也作为精炼器。它为后续的语言模型处理提供更有效和准确的输入[Zhuang et al., 2023]。

上下文压缩被纳入重排过程中，以提供更精确的检索信息。这种方法涉及减少单个文档的内容，并过滤整个文档，最终目标是在搜索结果中呈现最相关的信息，以便更专注和准确地展示相关内容。

为 RAG 微调 LLM

优化 RAG 模型中的生成器是其架构的关键方面。生成器的角色是接收检索到的信息并产生相关文本，形成模型的最终输出。生成器的优化旨在确保生成的文本既自然又有效地利用检索到的文档，以更好地满足用户的查询需求。

在标准 LLM 生成任务中，输入通常包括一个查询。RAG 通过将查询和检索器检索到的各种文档（结构化/非结构化）纳入输入。这些额外的信息可以显著影响模型的理解，特别是对于较小的模型。在这种情况下，对模型进行微调以适应查询和检索文档的输入变得至关重要。在将输入呈现给微调模型之前，通常会对检索器检索到的文档进行检索后处理。需要注意的是，RAG 中生成器的微调方法与 LLMs 的一般微调方法一致。接下来，我们将简要描述一些涉及数据（格式化/非格式化）和优化函数的代表性工作。

一般优化过程

作为一般优化过程的一部分，训练数据通常由输入输出对组成，旨在训练模型在给定输入 x 的

情况下产生输出 y。在 Self-Mem [Chengetal.,2023b]的工作中，采用了传统的训练过程，其中给定输入 x，检索相关文档 z（论文中选择 Top-1），并在整合（x，z）后，模型生成输出 y。该论文利用了两种常见的微调范式，即 Joint-Encoder 和 Dual-Encoder [Arora et al., 2023, Wang et al., 2022b, Lewis et al., 2020, Xia et al., 2019, Cai et al., 2021, Cheng et al., 2022]。

在 Joint-Encoder 范式中，使用基于编码器-解码器的标准模型。这里，编码器最初对输入进行编码，解码器通过注意力机制结合编码结果，以自回归方式生成标记。另一方面，在 Dual-Encoder 范式中，系统设置两个独立的编码器，每个编码器分别对输入（查询，上下文）和文档进行编码。产生的输出按顺序经过解码器的双向交叉注意力处理。这两种架构都使用 Transformer [Vaswani et al., 2017]作为基础块，并使用负对数似然损失进行优化。

利用对比学习

在为语言模型准备训练数据的阶段，通常会创建输入和输出的交互对。这种传统方法可能导致“曝光偏差”，即模型只针对单个正确的输出示例进行训练，从而限制了其对可能输出范围的曝光。这种限制可能会通过使模型过度拟合训练集中的特定示例，从而降低其在各种上下文中泛化的能力，影响模型在现实世界中的性能。

为了减轻曝光偏差，SURGE [Kang et al., 2023]提出了使用图-文本对比学习。这种方法包括一个对比学习目标，促使模型产生一系列合理且连贯的响应，超越了训练数据中遇到的实例。这种方法对于减少过拟合并加强模型的泛化能力至关重要。

对于涉及结构化数据的检索任务，SANTA 框架 [Li et al., 2023d]实施了三部分训练计划，有效地封装了结构和语义细节。初始阶段专注于检索器，利用对比学习来细化查询和文档嵌入。随后，生成器的初步训练阶段采用对比学习，将结构化数据与其非结构化文档描述对齐。在生成器训练的进一步阶段，模型认识到实体语义在文本数据表示学习中的关键作用，如[Sciavolino et al., 2021, Zhang et al., 2019]所强调。这个过程从识别结构化数据中的实体开始，然后在生成器的输入数据中对这些实体应用掩码，为模型预测和预测这些掩蔽元素做好准备。

训练计划随着模型学习利用上下文信息重建掩蔽实体而进展。这种练习培养了模型对文本数据结构语义的理解，并促进了结构化数据中相关实体的对齐。总体优化目标是训练语言模型准确地恢复被遮蔽的跨度，从而丰富其对实体语义的理解[Ye et al., 2020]。

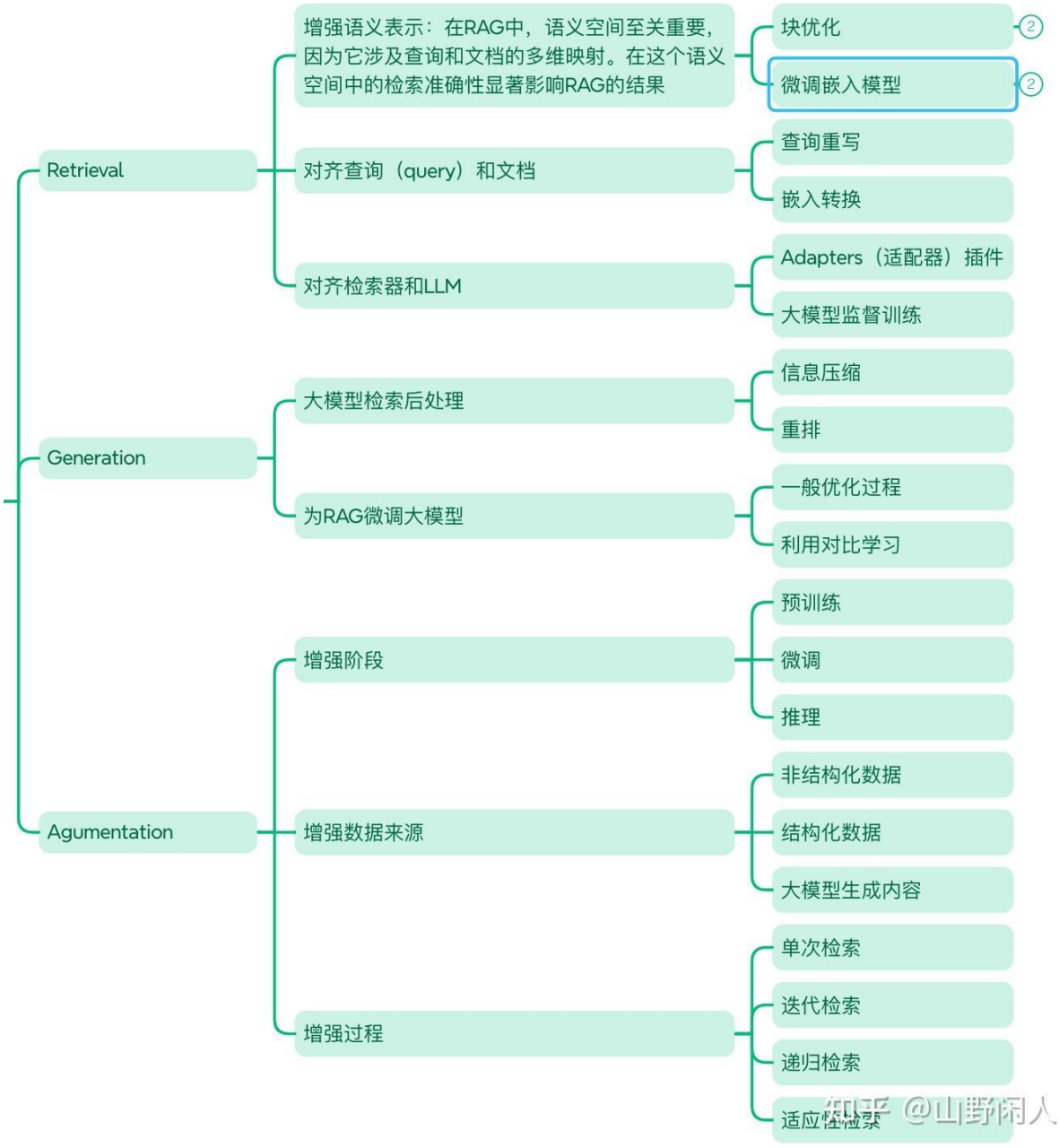

图 4 RAG 中的核心组件

RAG 中的增强

本节围绕三个关键方面展开：增强阶段、增强数据来源和增强过程。这些方面阐明了对 RAG 发展至关重要的关键技术。RAG 核心组件的分类如图 4 所示。

RAG 增强阶段

RAG 是一个知识密集型的努力，涵盖了语言模型训练的预训练、微调和推理阶段的各种技术方法。

预训练阶段 在预训练阶段，研究人员已经研究了通过基于检索的策略来增强开放领域问答

（QA）的预训练转换模型（PTMs）的方法。REALM 模型采用了一种结构化、可解释的知识嵌入方法，将预训练和微调框架为掩码语言模型（MLM）框架内的检索然后预测的工作流[Arora et al., 2023]。

实证证据强调了文本生成质量、事实准确性、减少毒性和下游任务熟练度的显著提高，特别是在开放领域问答等知识密集型应用中。这些结果表明，将检索机制整合到自回归语言模型的预训练中是一个有前景的途径，将复杂的检索技术与广泛的语言模型相结合，以产生更精确和高效的语言生成。

增强预训练的好处包括一个强大的基础模型，其在困惑度、文本生成质量和特定任务性能方面优于标准 GPT 模型，同时使用更少的参数。这种方法特别擅长处理知识密集型任务，并通过在专业语料库上进行训练来促进领域特定模型的发展。

然而，这种方法面临着挑战，如需要大量的预训练数据集和资源，以及随着模型大小的增加，更新频率的降低。尽管存在这些障碍，这种方法在模型弹性方面提供了显著的优势。一旦训练完成，增强检索的模型可以独立于外部库运行，提高生成速度和操作效率。这些潜在的收益使这种方法成为人工智能和机器学习领域持续研究和创新的引人注目的主题。

微调阶段 RAG 和微调是增强 LLMs 的强大工具，两者结合可以满足更具体场景的需求。一方面，微调允许检索具有独特风格的文档，实现更好的语义表达，并调整查询和文档之间的差异。这确保了检索器的输出更适合当前场景。另一方面，微调可以实现风格化和针对性的调整需求。此外，微调还可以用于调整检索器和生成器，以提高模型协同作用。

推理阶段 RAG 模型中的推理阶段至关重要，因为它涉及与 LLMs 的广泛集成。传统的 RAG 方法，也称为 Naive RAG，涉及在这个阶段加入检索内容以指导生成过程。

本质上，这些推理阶段的增强提供了轻量级、成本效益高的替代方案，利用预训练模型的能力，而无需进一步训练。主要优势是在保持静态 LLM 参数的同时提供上下文相关信息以满足特定任务需求。然而，这种方法并非没有局限性，因为它需要精心的数据处理和优化，并受到基础模型固有能力的约束。为了有效地满足多样化的任务需求，这种方法通常与程序优化技术（如逐步推理、迭代检索和自适应检索策略）结合使用。

增强来源

RAG 模型的有效性在很大程度上受到增强数据来源选择的影响。不同层次的知识和维度需要不同的处理技术。它们被归类为非结构化数据、结构化数据和由 LLMs 生成的内容。图 5 展示了具有不同增强方面的代表性 RAG 研究的技术树。叶子以三种不同的阴影着色，代表使用各种类型数据的增强：非结构化数据、结构化数据和由 LLMs 生成的内容。图表清楚地表明，最初，增强主要是通过非结构化数据（如纯文本）实现的。这种方法后来扩展到包括使用结构化数据（例如知识图）以进一步改进。最近，研究中出现了一个趋势，利用 LLMs 自身生成的内容进行检索和增强。

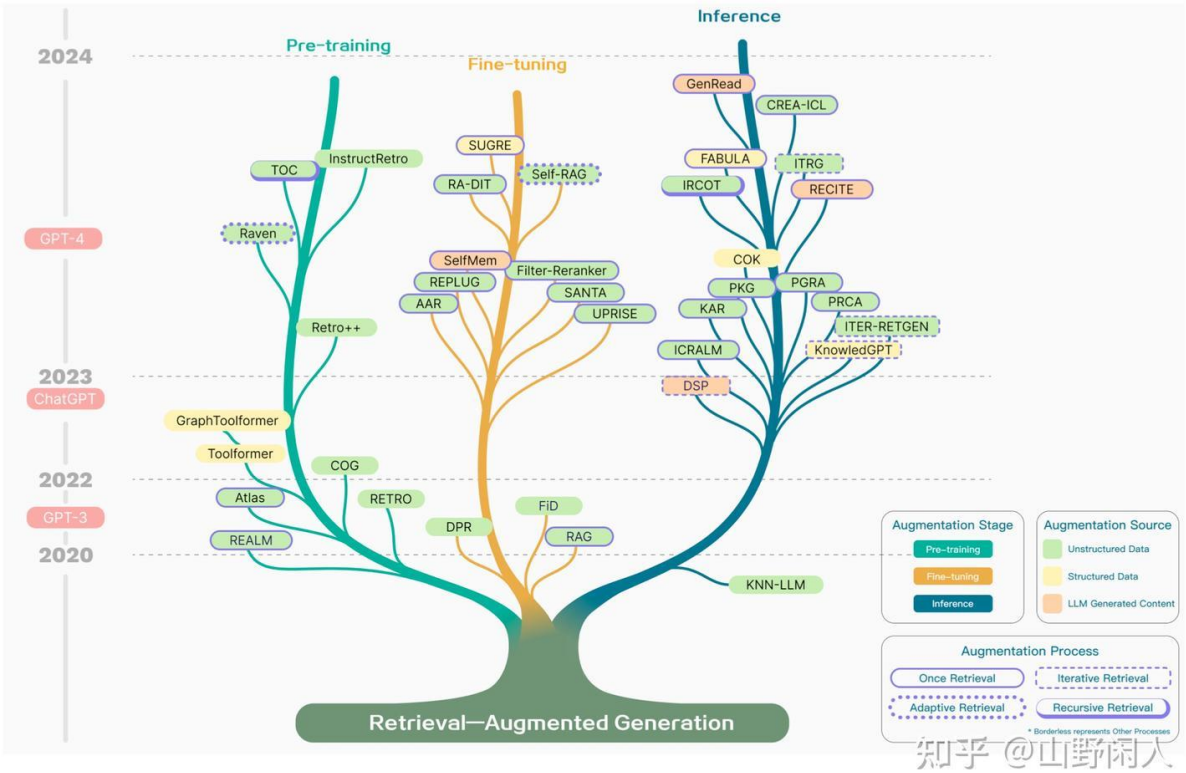

Figure 5: Technology tree of representative RAG research with different augmentation aspects

图 5 RAG 研究科技树 不同的增强方式

使用非结构化数据增强 非结构化文本是从语料库中收集的，例如用于微调大型模型的提示数据[Cheng 等人，2023a]和跨语言数据[Li 等人，2023b]。检索单元从标记（例如，kNN-LM [Khandelwal 等人，2019]）到短语（例如，NPM，COG [Lee 等人，2020，Lan 等人，2022]）和文档段落不等，更细的粒度提供了精确度，但代价是增加了检索复杂性。

FLARE [Jiang 等人，2023b]引入了一种由 LM（语言模型）生成低概率词触发的主动检索方法。它创建一个临时句子用于文档检索，然后使用检索到的上下文重新生成句子以预测后续句子。RETRO 使用前一块内容检索最近邻，结合前一块内容的上下文，指导下一块内容的生成。为了保持因果关系，下一块 Ci 的生成只利用前一块 N(Ci−1)的最近邻，而不是 N(Ci)。

使用结构化数据增强 结构化数据，如知识图谱（KGs），提供了高质量的上下文并减轻了模型幻觉。RET-LLMs [Modarressi 等人，2023]从过去的对话中构建知识图谱记忆以供将来参考。SUGRE [Kang 等人，2023]采用图神经网络（GNNs）编码相关的 KG 子图，通过多模态对比学习确保检索到的事实与生成文本之间的一致性。Knowl-edGPT [Wang 等人，2023d]生成 KB（知识库）搜索查询并在个性化基础上存储知识，增强了 RAG 模型的知识丰富性和上下文性。

LLM 在 RAG 中生成的内容 为了解决 RAG 中外部辅助信息的局限性，一些研究专注于利用 LLM 的内部知识。SKR [Wang 等人，2023e]将问题分类为已知或未知，有选择地应用检索增强。GenRead [Yu 等人，2022]用 LLM 生成器替换检索器，发现 LLM 生成的上下文通常包含更准确的答案，因为它们更好地与因果语言建模的预训练目标对齐。Selfmem [Cheng 等人，2023b]通过检索增强的生成器迭代创建一个无界的记忆池，使用记忆选择器选择作为原始问题的双重

问题的输出，从而自我增强生成模型。

这些方法论强调了在 RAG 中创新数据源利用的广度，努力提高模型性能和任务效果。

增强过程

在 RAG 领域，标准做法通常涉及单一的检索步骤，然后进行生成，这可能导致效率不高。一个值得注意的问题，被称为“中间迷失”现象，**当单一检索产生冗余内容时会出现，这可能会稀释或与关键信息相矛盾，从而降低生成质量[Liu 等人，2023a]**。此外，这种单一检索通常不足以应对需要多步推理的复杂问题，因为它提供的信息范围有限[Yoran 等人，2023]。

正如图 5 所示，为了克服这些挑战，当代研究提出了完善检索过程的方法：**迭代检索、递归检索和自适应检索。**迭代检索允许模型进行多个检索周期，增强所获得信息的深度和相关性。递归检索过程中，一个检索操作的结果被用作随后检索的输入。它有助于深入挖掘相关信息，特别是处理复杂或多步查询时。递归检索通常用于需要逐步方法才能收敛到最终答案的场景，例如学术研究、法律案例分析或某些类型的数据挖掘任务。另一方面，自适应检索提供了动态调整机制，根据不同任务和上下文的具体需求定制检索过程。

迭代检索 在 RAG 模型中的迭代检索是一个过程，其中文档基于初始查询和迄今为止生成的文本被重复收集，为 LLM 提供了更全面的知识库[Borgeaud 等人，2022，Arora 等人，2023]。这种方法已被证明可以通过多次检索迭代提供额外的上下文参考，从而增强后续答案生成的鲁棒性。然而，它可能会受到语义不连续性和无关信息积累的影响，因为它通常依赖于一系列 n 个标记来界定生成文本和检索文档之间的边界。

为了解决特定的数据场景，递归检索和多跳检索技术被利用。递归检索涉及一个结构化的索引，以层次化的方式处理和检索数据，这可能包括在基于此摘要执行检索之前总结文档的部分或长 PDF。随后，在文档内进行的二次检索细化了搜索，体现了过程的递归性质。相比之下，多跳检索旨在深入挖掘图结构化数据源，提取相互关联的信息[Li 等人，2023c]。

此外，一些方法将检索和生成步骤整合在一起。ITER-RETGEN [Shao 等人，2023]采用了一种协同方法，利用“检索增强生成”和“生成增强检索”来执行需要复制特定信息的任务。该模型利用所需的内容作为检索相关知识的上下文基础，这反过来又促进了后续迭代中改进响应的生成。

递归检索 递归检索通常用于信息检索和自然语言处理，以提高搜索结果的深度和相关性。 该过程涉及根据先前搜索结果迭代地完善搜索查询。**递归检索旨在通过反馈循环逐步收敛到最相关的信息，从而增强搜索体验。**IRCoT [Trivedi 等人，2022]使用思维链来指导检索过程，并根据获得的检索结果细化思维链。ToC [Kim 等人，2023]创建了一个澄清树，系统地优化查询中的模糊部分。它在复杂的搜索场景中特别有用，其中用户的需求从一开始就不完全清楚，或者所寻求的信息非常专业化或微妙。过程的递归性质允许持续学习和适应用户的需求，通常导致对搜索结果的满意度提高。

自适应检索 自适应检索方法，如 Flare 和 Self-RAG [Jiang 等人，2023b, Asai 等人，2023]，**通过使 LLM 能够主动确定检索的最佳时机和内容，从而提高 RAG 框架的效率和信息的相关性。**这

些方法是一个更广泛趋势的一部分，其中 LLM 在其操作中采用主动判断，如 AutoGPT、Toolformer 和 Graph-Toolformer 等模型代理[Yang 等人，2023c，Schick 等人，2023，Zhang，2023]。例如，Graph-Toolformer 将其检索过程分为不同的步骤，其中 LLM 主动使用检索器，应用自我提问技术，并使用少量提示来启动搜索查询。这种主动立场允许 LLM 决定何时搜索所需信息，类似于代理如何使用工具。 WebGPT [Nakano 等人，2021]集成了一个强化学习框架，在文本生成过程中自主地使用搜索引擎训练 GPT-3 模型。它使用特殊标记来导航这个过程，这些标记促进了如搜索引擎查询、浏览结果和引用参考等操作，从而通过使用外部搜索引擎扩展了 GPT-3 的能力。 Flare 通过监控生成过程中的置信度（由生成术语的概率表示）来自动化定时检索[Jiang 等人，2023b]。当概率低于某个阈值时，会激活检索系统以收集相关信息，从而优化检索周期。 Self-RAG [Asai 等人，2023]引入了“反思标记”，允许模型反思其输出。这些标记有两种类型：“检索”和“批评”。模型自主决定何时激活检索，或者，预定义的阈值可能触发该过程。在检索期间，生成器在多个段落之间进行片段级别的束搜索，以得出最连贯的序列。批评分数用于更新细分分数，具有在推理期间调整这些权重的灵活性，以定制模型的行为。Self-RAG 的设计消除了对额外分类器或依赖于自然语言推理（NLI）模型的需求，从而简化了何时参与检索机制的决策过程，并提高了模型在生成准确响应方面的自主判断能力。 由于 LLM 的日益普及，LLM 优化受到了显著关注。提示工程、微调（FT）和 RAG 等技术各有特点，如图 6 所示。虽然提示工程利用了模型的固有能力，但优化 LLM 通常需要应用 RAG 和 FT 方法。RAG 和 FT 之间的选择应基于场景的具体要求和每种方法的固有属性。RAG 和 FT 的详细比较见表 1。

RAG 与微调

RAG 就像给模型一本针对特定查询定制的信息检索教科书。另一方面，FT 就像学生随着时间的推移内化知识，更适合复制特定的结构、风格或格式。FT 可以通过加强基础模型知识、调整输出和教授复杂指令来提高模型的性能和效率。然而，它在整合新知识或快速迭代新用例方面并不如 RAG。 这两种方法，RAG 和 FT，并不是相互排斥的，而是可以互补的，在不同层面增强模型的能力。在某些情况下，它们的结合使用可能会产生最佳性能。涉及 RAG 和 FT 的优化过程可能需要多次迭代才能达到满意的结果。

未来展望

本节探讨了 RAG 的三个未来前景：未来的挑战、模态扩展和 RAG 生态系统。

RAG 的未来挑战

尽管 RAG 技术取得了显著进展，但仍存在一些挑战，需要深入研究：

上下文长度。RAG 的有效性受到大型语言模型（LLMs）上下文窗口大小的限制。平衡窗口过短可能导致信息不足，过长可能导致信息稀释的权衡至关重要。随着持续努力扩大 LLM 上下文窗口到几乎无限大，RAG 适应这些变化的研究提出了重要的问题[Xu 等人，2023c，Packer 等人，2023，Xiao 等人，2023]。

鲁棒性。检索过程中噪声或矛盾信息的存在可能对 RAG 的输出质量产生不利影响。这种情况被形象地称为“错误信息可能比没有信息更糟糕”。提高 RAG 对这种对抗性或虚假输入的抵抗力正

在获得研究动力，并已成为关键的性能指标[Yu 等人，2023a, Glass 等人，2021, Baek 等人，2023]。

混合方法（RAG+FT）。将 RAG 与微调结合正成为一种领先的策略。确定 RAG 和微调的最佳整合方式，无论是顺序、交替还是通过端到端的联合训练，以及如何利用参数化和非参数化的优势，都是值得探索的领域[Lin 等人，2023]。

扩展 LLM 角色。除了生成最终答案，LLMs 还在 RAG 框架中用于检索和评估。寻找进一步解锁 RAG 系统中 LLM 潜力的方法是一个不断增长的研究方向。

缩放定律。虽然 LLMs 已经建立了缩放定律[Kaplan 等人，2020]，但它们对 RAG 的适用性仍然不确定。初步研究[Wang 等人，2023b]已经开始解决这个问题，但 RAG 模型的参数数量仍然落后于 LLMs。较小模型胜过较大模型的逆缩放定律的可能性特别引人入胜，值得进一步研究。

RAG 实际使用。RAG 的实用性和与工程要求的一致性促进了它的采用。然而，提高检索效率、改善大型知识库中的文档召回率以及确保数据安全——例如防止 LLMs 无意中披露文档来源或元数据——是需要解决的关键工程挑战[Alon 等人，2022]。

RAG 的模态扩展

RAG（Retrieval-Augmented Generation）已经超越了最初的基于文本的问答限制，拥抱了各种模态数据的多样化。这种扩展催生了创新的多模态模型，这些模型在各个领域整合了 RAG 概念：

图像。RA-CM3 [Yasunaga 等人，2022] 是一个开创性的多模态模型，既能检索又能生成文本和图像。BLIP-2 [Li 等人，2023a] 利用冻结的图像编码器与 LLM（大型语言模型）结合，进行高效的视觉语言预训练，实现零样本图像到文本的转换。"Visualize Before You Write" 方法 [Zhu 等人，2022] 通过图像生成来引导 LM（语言模型）的文本生成，在开放式文本生成任务中显示出前景。

音频和视频。GSS 方法检索并拼接音频片段，将机器翻译数据转换为语音翻译数据 [Zhaoeetal.,2022]。UEOP 在端到端自动语音识别方面取得了重大进展，通过整合外部、离线的语音转文本转换策略 [Chan 等人，2023]。此外，基于 KNN 的注意力融合利用音频嵌入和语义相关的文本嵌入来细化 ASR（自动语音识别），从而加速领域适应。Vid2Seq 通过在语言模型中加入专门的时间标记，帮助预测事件边界和文本描述，统一输出序列 [Yang 等人，2023a]。

代码。RBPS [Nashid 等人，2023] 在小规模学习任务中表现出色，通过编码和频率分析检索与开发者目标一致的代码示例。这种方法在测试断言生成和程序修复等任务中显示出有效性。对于结构化知识，CoK 方法 [Li 等人，2023c] 首先从知识图中提取与输入查询相关的事实，然后将这些事实作为提示集成到输入中，提高了知识图谱问答任务的性能。

RAG 生态系统

下游任务和评估

RAG 在通过利用广泛的知识库处理复杂查询和生成详细回答方面显示出相当的潜力。实证证据表明，RAG 在多种下游任务中表现出色，包括开放式问答和事实验证。RAG 的整合不仅提高了

回答的精确性和相关性，还提高了其多样性和深度。

RAG 在多个领域的可扩展性和多功能性需要进一步研究，特别是在医学、法律和教育等专业领域。在这些领域，RAG 可能相比传统的微调方法在专业领域知识问答中降低训练成本并提高性能。

同时，完善 RAG 的评估框架对于最大化其在不同任务中的有效性和实用性至关重要。这需要开发细致的指标和评估工具，以衡量上下文相关性、内容创造性和非恶意性等方面。

此外，提高 RAG 驱动模型的可解释性仍然是一个关键目标。这样做可以让用户理解模型生成回答背后的推理过程，从而在使用 RAG 应用时促进信任和透明度。

技术栈

RAG 生态系统的发展受到其技术栈进展的极大影响。随着 ChatGPT 的出现，像 LangChain 和 LLamaIndex 这样的关键工具迅速流行起来，提供了广泛的 RAG 相关 API，并在 LLM 领域变得必不可少。

新兴的技术栈，虽然功能不如 LangChain 和 LLamaIndex 丰富，但以其专业服务而脱颖而出。例如，Flowise AI10 优先考虑低代码方法，使用户能够通过用户友好的拖放界面部署包括 RAG 在内的 AI 应用。其他技术如 HayStack、Meltano11 和 Co-here Coral12 也因其对该领域的特殊贡献而受到关注。

除了 AI 专业提供商外，传统的软件和云服务提供商也在扩展其产品，以包括以 RAG 为中心的服务。Weaviate 的 Verba13 专为个人助理应用设计，而亚马逊的 Kendra14 提供了一个智能企业搜索服务，允许用户使用内置连接器浏览各种内容库。在 RAG 技术景观的演变过程中，出现了明显的分化，如：1）定制化。根据特定要求定制 RAG。2）简化。使 RAG 更易于使用，从而降低初始学习曲线。3）专业化。改进 RAG 以更有效地服务于生产环境。

RAG 模型与其技术栈的相互增长是显而易见的；技术进步不断为现有基础设施设定新标准。反过来，技术栈的增强推动了 RAG 能力的进化。RAG 工具包正在汇聚成一个基础技术栈，为高级企业应用奠定基础。然而，一个完全集成、全面的平台的概念仍然在地平线上，有待进一步的创新和发展。

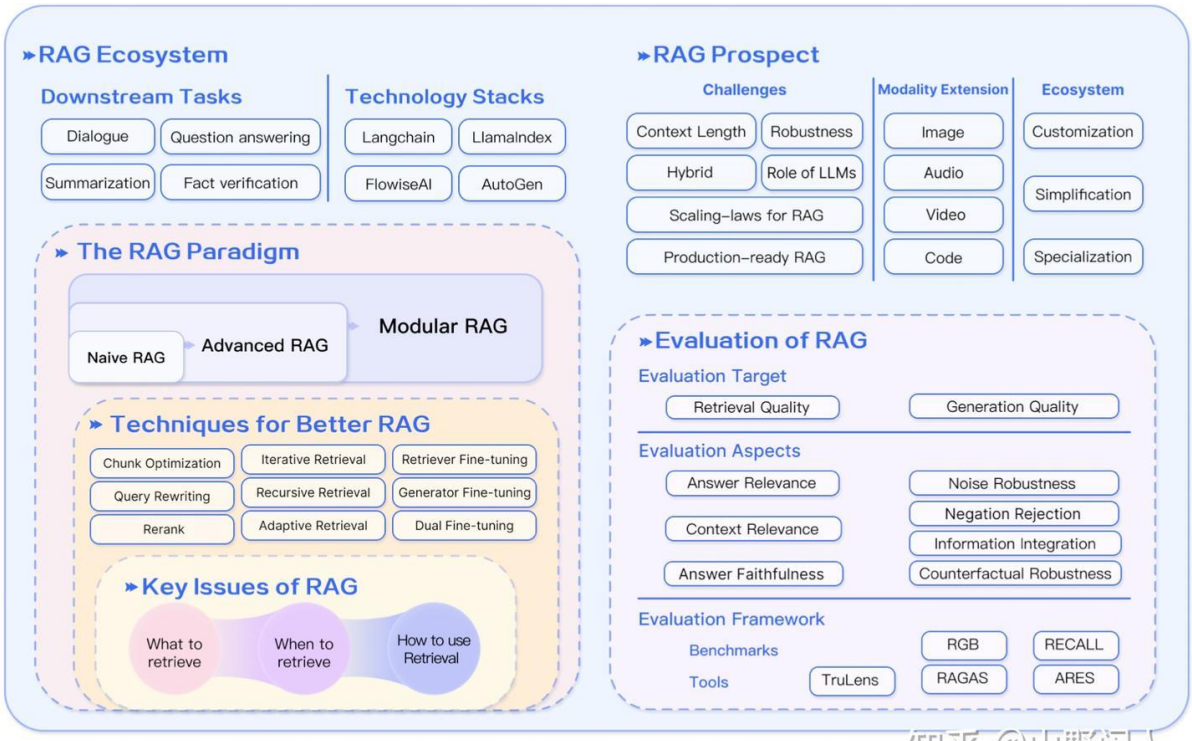

Figure 7: Summary of RAG ecosystem

图 7RAG 生态系统总结

结论

本文的总结如图 7 所示，突出了 RAG 在通过整合来自语言模型的参数化知识与来自外部知识库的广泛非参数化数据，显著提升了 LLM（大型语言模型）能力的重要进步。我们的调查展示了 RAG 技术的演变及其对知识密集型任务的影响。我们的分析在 RAG 框架内划分了三个发展范式：初级、高级和模块化 RAG，每个范式都标志着对其前身的逐步增强。**高级 RAG 范式超越了初级方法，通过整合复杂的架构元素，包括查询重写、块重排和提示总结。这些创新导致了一种更细致和模块化的架构，既提高了 LLM 的性能，也提高了其可解释性。RAG 与其他 AI 方法（如微调和强化学习）的技术整合进一步扩展了其能力。**在内容检索方面，一种利用结构化和非结构化数据源的混合方法正在成为一种趋势，提供了更丰富的检索过程。RAG 框架内的前沿研究正在探索新概念，如 LLM 的自我检索和信息检索的动态时机。

尽管 RAG 技术取得了长足进步，但在提高其鲁棒性及其管理扩展上下文的能力方面仍有许多研究机会。RAG 的应用范围也在扩展到多模态领域，适应其原则以解释和处理各种数据形式，如图像、视频和代码。这种扩展强调了 RAG 对 AI 部署的重要实际意义，吸引了学术界和工业界的关注。RAG 生态系统的增长得益于 RAG 中心 AI 应用的增加和支持工具的持续开发。然而，随着 RAG 应用领域的扩大，迫切需要完善评估方法以跟上其发展的步伐。确保性能评估保持准确和代表性对于捕捉 RAG 对 AI 研究和开发社区的全部贡献至关重要。

编辑于 2024-11-12 10:04・IP 属地广东