**一、RAG技术与应用**

**RAG 本质上就是重构了一个新的Prompt**

**RAG的分类：NativeRAG、AgentRAG、GraphRAG**

**1.什么是RAG ？**

RAG（Retrieval-Augmented Generation）检索增强生成，是一种结合信息检索（Retrieval）和文本生成（Generation）的技术

RAG技术通过实时检索相关文档或信息，并将其作为上下文输入到生成模型中，从而提高生成结果的时效性和准确性。

**2.RAG的优势是什么？**

**（1）解决知识时效性问题：**大模型的训练数据通常是静态的，无法涵盖最新信息，而RAG可以检索外部知识库实时更新信息。

**（2）减少模型幻觉：**通过引入外部知识，RAG能够减少模型生成虚假或不准确内容的可能性。

**（3）提升专业领域回答质量：**RAG能够结合垂直领域的专业知识库，生成更具专业深度的回答。

**（4）生成内容的溯源（可解释性）**

**3.RAG 的核心原理与流程**

**（1）数据预处理，构建索引库**

**知识库构建：**收集并整理文档、网页、数据库等多源数据，构建外部知识库

**文档分块：**将文档切分为适当大小的片段（chunks），以便后续检索。分块策略需要在语义完整性与检索效率之间取得平衡

**向量化处理：**使用嵌入模型（如BGE、M3E、Chinese-Alpaca-2等）将文本块转换为向量，并存储在向量数据库中

**（2）检索阶段**

**查询处理：**将用户输入的问题转换为向量，并在向量数据库中进行相似度检索，找到最相关的文本片段

**重排序：**对检索结果进行相关性排序，选择最相关的片段作为生成阶段的输入

**（3）生成阶段**

**上下文组装：**将检索到的文本片段与用户问题结合，形成增强的上下文输入

**生成回答：**大语言模型基于增强的上下文生成最终回答

**4.LangChain快速搭建本地知识库检索**

1. **PDF文本提取与处理**

使用PyPDF2库的PdfReader从PDF文件中提取文本在提取过程中记录每行文本对应的页码，便于后续溯源

使用RecursiveCharacterTextSplitter将长文本分割成小块，便于向量化处理

**（2）向量数据库构建**

使用OpenAIEmbeddings / DashScopeEmbeddings将文本块转换为向量表示

使用FAISS向量数据库存储文本向量，支持高效的相似度搜索为每个文本块保存对应的页码信息，实现查询结果溯源

1. **语义搜索与问答链**

基于用户查询，使用similarity\_search在向量数据库中检索相关文本块

使用文本语言模型和load\_qa\_chain构建问答链将检索到的文档和用户问题作为输入，生成回答

1. **成本跟踪与结果展示**

使用get\_openai\_callback跟踪API调用成本

展示问答结果和来源页码，方便用户验证信息

**5.三大阶段有效提升RAG质量方法**

**5.1数据准备阶段**

**5.1.1常见问题**

**（1）数据质量差：**企业大部分数据（尤其是非结构化数据）缺乏良好的数据治理，未经标记/评估的非结构化数据可能包含敏感、过时、矛盾或不正确的信息。

**（2）多模态信息：**提取、定义和理解文档中的不同内容元素，如标题、配色方案、图像和标签等存在挑战。

**（3）复杂的PDF提取：**PDF是为人类阅读而设计的，机器解析起来非常复杂。

**5.1.2如何提升数据准备阶段的质量？**

**（1）构建完整的数据准备流程**

**数据评估与分类**

数据审计：全面审查现有数据，识别敏感、过时、矛盾或不准确的信息。

数据分类：按类型、来源、敏感性和重要性对数据进行分类，便于后续处理。

**数据清洗**

去重：删除重复数据

纠错：修正格式错误、拼写错误等

更新：替换过时信息，确保数据时效性

一致性检查：解决数据矛盾，确保逻辑一致

**敏感信息处理**

识别敏感数据：使用工具或正则表达式识别敏感信息，如个人身份信息

脱敏或加密：对敏感数据进行脱敏处理，确保合规。

**数据标记与标注**

元数据标记：为数据添加元数据，如来源、创建时间等

内容标注：对非结构化数据进行标注，便于后续检索和分析

**数据治理框架**

制定政策：明确数据管理、访问控制和更新流程

责任分配：指定数据治理负责人，确保政策执行

监控与审计：定期监控数据质量，进行审计

**（2）智能文档技术**

**阿里文档智能：**

https://www.aliyun.com/product/ai/docmind?spm=a2c4g.11174283.0.0.bfe667a8tIVMdG

**微软：**

LayoutLMv3：<https://www.microsoft.com/en-us/research/articles/layoutlmv3/>

**5.2.知识检索阶段**

**5.2.1常见问题**

**（1）内容缺失：**当检索过程缺少关键内容时，系统会提供不完整、碎片化的答案 => 降低RAG的质量

**（2）错过排名靠前的文档：**用户查询相关的文档时被检索到，但相关性极低，导致答案不能满足用户需求，这是因为在检索过程中，用户通过主观判断决定检索“文档数量”。理论上所有文档都要被排序并考虑进一步处理，但在实践中，通常只有排名top k的文档才会被召回，而k值需要根据经验确定。

**（3）不在上下文中：**从数据库中检索出包含答案的文档，但未能包含在生成答案的上下文中。这种情况通常发生在返回大量文件时，需要进行整合以选择最相关的信息。

**5.2.2如何提升知识检索阶段的质量？**

**（1）通过查询转换澄清用户意图**:明确用户意图，提高检索准确性。

**场景：**用户询问 “如何申请信用卡？”

**问题：**用户意图可能模糊，例如不清楚是申请流程、所需材料还是资格条件。

**解决方法：**通过查询转换明确用户意图。

**实现步骤：**

意图识别：使用自然语言处理技术识别用户意图。例如识别用户是想了解流程、材料还是资格。

查询扩展：根据识别结果扩展查询。例如：

如果用户想了解流程，查询扩展为“信用卡申请的具体步骤”

如果用户想了解材料，查询扩展为“申请信用卡需要哪些材料”

如果用户想了解资格，查询扩展为“申请信用卡的资格条件”

检索：使用扩展后的查询检索相关文档

**示例：**

1. 用户输入：“如何申请信用卡？”

2. 系统识别意图为 `流程`，扩展查询为 `信用卡申请的具体步骤`

3. 检索结果包含详细的申请步骤文档，系统生成准确答案

**（2）采用混合检索和重排策略**:确保最相关的文档被优先处理，生成更准确的答案。

**场景：**用户询问“信用卡年费是多少？”

**问题：**直接检索可能返回大量文档，部分相关但排名低，导致答案不准确。

**解决方法：**采用混合检索和重排策略。

**步骤：**

1. 混合检索：结合关键词检索和语义检索。比如：关键词检索：“信用卡年费”。

2. 语义检索：使用嵌入模型检索与“信用卡年费”语义相近的文档。

3. 重排：对检索结果进行重排。

4. 生成答案：从重排后的文档中生成答案。

**示例：**

1. 用户输入：“信用卡年费是多少？”

2. 系统进行混合检索，结合关键词和语义检索。

3. 重排后，最相关的文档（如“信用卡年费政策”）排名靠前。

4. 系统生成准确答案：“信用卡年费根据卡类型不同，普通卡年费为100元，金卡为300元，白金卡为1000元。”

**5.3.答案生成阶段**

**5.3.1常见问题**

**（1）未提取：**答案与所提供的上下文相符，但大语言模型却无法准确提取。这种情况通常发生在上下文中存在过多噪音或相互冲突的信息时。

**（2）不完整：**尽管能够利用上下文生成答案，但信息缺失会导致对用户查询的答复不完整。格式错误：当prompt中的附加指令格式不正确时，大语言模型可能误解或曲解这些指令，从而导致错误的答案。

**（3）幻觉：**大模型可能会产生误导性或虚假性信息。

**5.3.2如何提升答案生成阶段的质量？**

**（1）改进提示词模板**

|  |  |  |
| --- | --- | --- |
| **场景** | **原始提示词** | **改进后的提示词** |
| 用户询问“如何申请信用卡？” | “根据以下上下文回答问题：如何申请信用卡？” | “根据以下上下文，提取与申请信用卡相关的具体步骤和所需材料：如何申请信用卡？” |
| 用户询问“信用卡的年费是多少？” | “根据以下上下文回答问题：信用卡的年费是多少？” | “根据以下上下文，详细列出不同信用卡的年费信息，并说明是否有减免政策：信用卡的年费是多少？” |
| 用户询问“什么是零存整取？” | “根据以下上下文回答问题：什么是零存整取？” | “根据以下上下文，准确解释零存整取的定义、特点和适用人群，确保信息真实可靠：什么是零存整取？” |

**如何对原有的提示词进行优化？**

可以通过 `DeepSeek-R1` 或 `QWQ` 的推理链，对提示词进行优化：

信息提取：从原始提示词中提取关键信息。

需求分析：分析用户的需求，明确用户希望获取的具体信息。

提示词优化：根据需求分析的结果，优化提示词，使其更具体、更符合用户的需求。

**（2）实施动态防护栏**

动态防护栏（Dynamic Guardrails）是一种在生成式AI系统中用于实时监控和调整模型输出的机制，旨在确保生成的内容符合预期、准确且安全。它通过设置规则、约束和反馈机制，动态地干预模型的生成过程，避免生成错误、不完整、不符合格式要求或含有虚假信息（幻觉）的内容。

在RAG系统中，动态防护栏的作用尤为重要，因为它可以帮助解决以下问题：

未提取：确保模型从上下文中提取了正确的信息。

不完整：确保生成的答案覆盖了所有必要的信息。

格式错误：确保生成的答案符合指定的格式要求。

幻觉：防止模型生成与上下文无关或虚假的信息。

**场景1：防止未提取**

用户问题：“如何申请信用卡？”

上下文：包含申请信用卡的步骤和所需材料。

动态防护栏规则：检查生成的答案是否包含“步骤”和“材料”。如果缺失，提示模型重新生成。

示例：

错误输出：“申请信用卡需要提供一些材料。”

防护栏触发：检测到未提取具体步骤，提示模型补充。

**场景2：防止不完整**

用户问题：“信用卡的年费是多少？”

上下文：包含不同信用卡的年费信息。

动态防护栏规则：检查生成的答案是否列出所有信用卡的年费。如果缺失，提示模型补充。

示例：

错误输出：“信用卡A的年费是100元。”

防护栏触发：检测到未列出所有信用卡的年费，提示模型补充。

**场景3：防止幻觉**

用户问题：“什么是零存整取？”

上下文：包含零存整取的定义和特点。

动态防护栏规则：检查生成的答案是否与上下文一致。如果不一致，提示模型重新生成。

示例：

错误输出：“零存整取是一种贷款产品。

防护栏触发：检测到与上下文不一致，提示模型重新生成。

**如何实现动态防护栏技术？**

事实性校验规则，在生成阶段，设置规则验证生成内容是否与检索到的知识片段一致。例如，可以使用参考文献验证机制，确保生成内容有可靠来源支持，避免输出矛盾或不合理的回答。

**如何制定事实性校验规则？**

当业务逻辑明确且规则较为固定时，可以人为定义一组规则，比如：

规则1：生成的答案必须包含检索到的知识片段中的关键实体（如“年费”、“利率”）。

规则2：生成的答案必须符合指定的格式（如步骤列表、表格等）。

实施方法：

使用正则表达式或关键词匹配来检查生成内容是否符合规则。

例如，检查生成内容是否包含“年费”这一关键词，或者是否符合步骤格式（如“1. 登录；2. 设置”）。

**6.RAG在不同阶段提升质量的实践**

**（1）数据准备环节**，阿里云考虑到文档具有多层标题属性且不同标题之间存在关联性，提出多粒度知识提取方案，按照不同标题级别对文档进行拆分，然后基于Qwen14b模型和RefGPT训练了一个面向知识提取任务的专属模型，对各个粒度的chunk进行知识提取和组合，并通过去重和降噪的过程保证知识不丢失、不冗余。最终将文档知识提取成多个事实型对话，提升检索效果；

**（2）知识检索环节**，哈啰出行采用多路召回的方式，主要是向量召回和搜索召回。其中，向量召回使用了两类，一类是大模型的向量、另一类是传统深度模型向量；搜索召回也是多链路的，包括关键词、ngram等。通过多路召回的方式，可以达到较高的召回查全率。

**（3）答案生成环节**，中国移动为了解决事实性不足或逻辑缺失，采用FoRAG两阶段生成策略，首先生成大纲，然后基于大纲扩展生成最终答案。

**7.QA**

**如果LLM可以处理无限上下文了，RAG还有意义吗？**

**效率与成本：**LLM处理长上下文时计算资源消耗大，响应时间增加。RAG通过检索相关片段，减少输入长度。

**知识更新：**LLM的知识截止于训练数据，无法实时更新。RAG可以连接外部知识库，增强时效性。

**可解释性：**RAG的检索过程透明，用户可查看来源，增强信任。LLM的生成过程则较难追溯。

**定制化：**RAG可针对特定领域定制检索系统，提供更精准的结果，而LLM的通用性可能无法满足特定需求。

**数据隐私：**RAG允许在本地或私有数据源上检索，避免敏感数据上传云端，适合隐私要求高的场景。

结合LLM的生成能力和RAG的检索能力，可以提升整体性能，提供更全面、准确的回答。

**二、RAG高级技术与实践**

**1.RAG技术树**

RAG研究的技术树主要涉及预训练（Pre-training）、微调（Fine-tuning）和推理（Inference）等阶段。

随着LLM的出现，RAG的研究最初侧重于利用LLMs强大的上下文学习能力，主要集中在推理阶段。

随后的研究进一步深入，逐渐与LLMs的微调阶段更加融合。研究人员也在探索通过检索增强技术来提升预训练阶段的语言模型性能。

参考文章：https://www.promptingguide.ai/research/rag

**2.RAFT方法**

RAFT方法（Retrieval Augmented Fine Tuning）

RAFT: Adapting Language Model to Domain Specific RAG, 2024 https://arxiv.org/pdf/2403.10131

如何最好地准备考试？

基于微调的方法通过“学习”来实现“记忆”输入文档或回答练习题而不参考文档。

或者，基于上下文检索的方法未能利用固定领域所提供的学习机会，相当于参加开卷考试但没有事先复习。

相比之下，我们的方法RAFT利用了微调与问答对，并在一个模拟的不完美检索环境中参考文档——从而有效地为开卷考试环境做准备。

让LLMs从一组正面和干扰文档中读取解决方案，这与标准的RAG设置形成对比，因为在标准的RAG设置中，模型是基于检索器输出进行训练的，这包含了记忆和阅读的混合体。在测试时，所有方法都遵循标准的RAG设置，即提供上下文中排名前k的检索文档。

微调数据集准备样例：

RAFT在所有专业领域的RAG性能上有所提升（在PubMed、HotPot、HuggingFace、Torch Hub和TensorflowHub等多个领域），领域特定的微调提高了基础模型的性能，RAFT无论是在有RAG的情况下还是没有RAG的情况下，都持续优于现有的领域特定微调方法。这表明了需要在上下文中训练模型。

**总结：**

RAFT方法（Retrieval Augmented Fine Tuning）：

适应特定领域的LLMs对于许多新兴应用至关重要，但如何有效融入信息仍是一个开放问题。

RAFT结合了检索增强生成（RAG）和监督微调（SFT），从而提高模型在特定领域内回答问题的能力。

训练模型识别并忽略那些不能帮助回答问题的干扰文档，只关注和引用相关的文档。

通过在训练中引入干扰文档，提高模型对干扰信息的鲁棒性，使其在测试时能更好地处理检索到的文档。

训练示例：https://github.com/lumpenspace/raft

**3.RAG高效召回方法**

**3.1.合理设置TOP\_K**

**3.2.改进索引算法**

**知识图谱：**利用知识图谱中的语义信息和实体关系，增强对查询和文档的理解，提升召回的相关性

**3.3.引入重排序（Reranking）**

**（1）重排序模型：**对召回结果进行重排，提升问题和文档的相关性。常见的重排序模型有 `BGE-Rerank` 和 `Cohere Rerank`。

**场景：**用户查询“如何提高深度学习模型的训练效率？”

**召回结果：**初步召回10篇文档，其中包含与“深度学习”、“训练效率”相关的文章。

**重排序：**BGE-Rerank对召回的10篇文档进行重新排序，将与“训练效率”最相关的文档（如“优化深度学习训练的技巧”）排在最前面，而将相关性较低的文档（如“深度学习基础理论”）排在后面。

**（2）混合检索：**结合向量检索和关键词检索的优势，通过重排序模型对结果进行归一化处理，提升召回质量。

**3.4.优化查询扩展**

**相似语义改写：** 使用大模型将用户查询改写成多个语义相近的查询，提升召回多样性。

例如，LangChain的 `MultiQueryRetriever` 支持多查询召回，再进行回答问题。

**3.5.双向改写**

将查询改写成文档（Query2Doc）或为文档生成查询（Doc2Query），缓解短文本向量化效果差的问题

**（1）Query2Doc：将查询改写成文档**

**用户查询：**“如何提高深度学习模型的训练效率？”

Query2Doc 改写：

原始查询较短，可能无法充分表达用户意图。

通过 Query2Doc 生成一段扩展文档：

提高深度学习模型的训练效率可以从以下几个方面入手：

1. 使用更高效的优化算法，如AdamW或LAMB。

2. 采用混合精度训练（Mixed Precision Training），减少显存占用并加速计算。

3. 使用分布式训练技术，如数据并行或模型并行。

4. 对数据进行预处理和增强，减少训练时的冗余计算。

5. 调整学习率调度策略，避免训练过程中的震荡。

**（2）Doc2Query：为文档生成关联查询**

文档内容：

本文介绍了深度学习模型训练中的优化技巧，包括：

1. 使用AdamW优化器替代传统的SGD。

2. 采用混合精度训练，减少显存占用。

3. 使用分布式训练技术加速大规模模型的训练……

通过 Doc2Query 生成一组可能的查询：

1. 如何选择深度学习模型的优化器？

2. 混合精度训练有哪些优势？

3. 分布式训练技术如何加速深度学习？

4. 如何减少深度学习训练中的显存占用？

5. 深度学习模型训练的最佳实践是什么？

**3.6.索引扩展**

**3.6.1.离散索引扩展：**

使用关键词抽取、实体识别等技术生成离散索引，与向量检索互补，提升召回准确性。

**（1）关键词抽取：**从文档中提取出重要的关键词，作为离散索引的一部分，用于补充向量检索的不足。

文档内容：

本文介绍了深度学习模型训练中的优化技巧，包括：

1. 使用AdamW优化器替代传统的SGD。

2. 采用混合精度训练，减少显存占用。

3. 使用分布式训练技术加速大规模模型的训练。

通过关键词抽取技术（如TF-IDF、TextRank）提取出以下关键词：

["深度学习", "模型训练", "优化技巧", "AdamW", "混合精度训练", "分布式训练"]

当用户查询“如何优化深度学习模型训练？”时，离散索引中的关键词能够快速匹配到相关文档。

**（2）实体识别：**从文档中识别出命名实体（如人名、地点、组织等），作为离散索引的一部分，增强检索的精确性。

文档内容：

2023年诺贝尔物理学奖授予了三位科学家，以表彰他们在量子纠缠领域的研究成果。

通过实体识别技术（如SpaCy、BERT-based NER）提取出以下实体：

["2023年", "诺贝尔物理学奖", "量子纠缠"]

当用户查询“2023年诺贝尔物理学奖的获奖者是谁？”时，离散索引中的实体能够快速匹配到相关文档。

**3.6.2.混合索引召回**

将BM25等离散索引与向量索引结合，通过Ensemble Retriever实现混合召回，提升召回多样性

将离散索引（如关键词、实体）与向量索引结合，通过混合召回策略提升检索效果。

本文介绍了人工智能在医疗领域的应用，包括：

1. 使用深度学习技术进行医学影像分析。

2. 利用自然语言处理技术提取电子病历中的关键信息。

3. 开发智能诊断系统辅助医生决策。

关键词抽取：["人工智能", "医疗领域", "深度学习", "医学影像分析", "自然语言处理", "电子病历", "智能诊断系统"]

实体识别：["人工智能", "医疗领域", "深度学习", "自然语言处理"]

当用户查询“人工智能在医疗领域的应用有哪些？”时：离散索引通过关键词和实体匹配到相关文档。向量索引通过语义相似度匹配到相关文档。综合两种召回结果，提升检索的准确性和覆盖率。

**3.6.3.Small-to-Big**

**Small-to-Big 索引策略：**

一种高效的检索方法，特别适用于处理长文档或多文档场景。核心思想是通过小规模内容（如摘要、关键句或段落）建立索引，并链接到大规模内容主体中。这种策略的优势在于能够快速定位相关的小规模内容，并通过链接获取更详细的上下文信息，从而提高检索效率和答案的逻辑连贯性。

小规模内容（索引部分）：

摘要：从每篇论文中提取摘要作为索引内容。

摘要1：本文介绍了Transformer 模型在机器翻译任务中的应用，并提出了改进的注意力机制。

摘要2：本文探讨了Transformer 模型在文本生成任务中的性能，并与RNN 模型进行了对比。

关键句：从论文中提取与查询相关的关键句。

关键句1：Transformer 模型通过自注意力机制实现了高效的并行计算。

关键句2：BERT 是基于Transformer 的预训练模型，在多项NLP 任务中取得了显著效果。

大规模内容（链接部分）：

每篇论文的完整内容作为大规模内容，通过链接与小规模内容关联。

论文1：链接到完整的PDF 文档，包含详细的实验和结果。

论文2：链接到完整的PDF 文档，包含模型架构和性能分析。

**Small-to-Big机制：**

**1.小规模内容检索：**用户输入查询后，系统首先在小规模内容（如摘要、关键句或段落）中检索匹配的内容。小规模内容通常是通过摘要生成、关键句提取等技术从大规模内容中提取的，并建立索引。

**2.链接到大规模内容：**当小规模内容匹配到用户的查询后，系统会通过预定义的链接（如文档ID、URL 或指针）找到对应的大规模内容（如完整的文档、文章）。大规模内容包含更详细的上下文信息，为RAG 提供丰富的背景知识。

**3.上下文补充：**将大规模内容作为RAG 系统的上下文输入，结合用户查询和小规模内容，生成更准确和连贯的答案。

**4.Qwen-Agent 构建 RAG**

**4.1. 基本介绍**

Qwen-Agent是一个开发框架。充分利用基于通义千问模型（Qwen）的指令遵循、工具使用、规划、记忆能力。

Qwen-Agent支持的模型形式：

DashScope服务提供的Qwen模型服务

支通过OpenAI API方式接入开源的Qwen模型服务

Github：https://github.com/QwenLM/Qwen-Agent

**4.2.使用Qwen-Agent将上下文记忆扩展到百万量级**

现在能够原生处理数百万字输入的大型语言模型（LLMs）成为了一种趋势。如何让一个上下文长度为8K的模型，能处理1M的上下文？

可以采取以下方法准备数据：

1. 利用一个较弱的8k上下文聊天模型构建一个相对强大的智能体，能够处理1M的上下文。

2. 随后，使用该智能体合成微调数据，并应用自动化过滤确保数据质量。

3. 最终，使用合成数据对预训练模型进行微调，得到一个强大的1M上下文聊天模型

**5.RAG质量评估**

当我们完成了一个RAG系统的开发工作以后，我们还需要对RAG系统的性能进行评估，那如何来对RAG系统的性能进行评估呢？我们可以仔细分析一下RAG系统的产出成果，比如检索器组件它产出的是检索出来的相关文档即context, 而生成器组件它产出的是最终的答案即answer,除此之外还有我们最初的用户问题即question。因此RAG系统的评估应该是将question、context、answer结合在一起进行评估。

**5.1.RAG 三元组**

标准的 RAG 流程就是用户提出 Query 问题，RAG 应用去召回 Context，然后 LLM 将 Context 组装，生成满足 Query 的 Response 回答。那么在这里出现的三元组:—— Query、Context 和 Response 就是 RAG 整个过程中最重要的三元组，它们之间两两相互牵制。我们可以通过检测三元组之间两两元素的相关度，来评估这个 RAG 应用的效果：

**（1）Context Relevance:** 衡量召回的 Context 能够支持 Query 的程度。如果该得分低，反应出了召回了太多与Query 问题无关的内容，这些错误的召回知识会对 LLM 的最终回答造成一定影响。

**（2）Groundedness:** 衡量 LLM 的 Response 遵从召回的 Context 的程度。如果该得分低，反应出了 LLM 的回答不遵从召回的知识，那么回答出现幻觉的可能就越大。

**（3）Answer Relevance:** 衡量最终的 Response 回答对 Query 提问的相关度。如果该得分低，反应出了可能答不对题。

**5.2.Ragas评估**

**5.2.1. 什么是RAGAs评估**

官网地址：https://www.ragas.io/

Ragas (Retrieval-Augmented Generation Assessment) 它是一个框架，它可以帮助我们来快速评估RAG系统的性能，为了评估RAG系统，Ragas需要以下信息:

question：用户输入的问题。

answer：从RAG系统生成的答案(由LLM给出)。

contexts：根据用户的问题从外部知识源检索的上下文即与问题相关的文档。

ground\_truths：人类提供的基于问题的真实(正确)答案。 这是唯一的需要人类提供的信息。

**5.2.2. 评估指标**

Ragas提供了五种评估指标包括：

**（1）忠实度(faithfulness)**

忠实度(faithfulness)衡量了生成的答案(answer)与给定上下文(context)的事实一致性。它是根据answer和检索到的context计算得出的。并将计算结果缩放到 (0,1) 范围且越高越好。

如果答案(answer)中提出的所有基本事实(claims)都可以从给定的上下文(context)中推断出来，则生成的答案被认为是忠实的。为了计算这一点，首先从生成的答案中识别一组claims。然后，将这些claims中的每一项与给定的context进行交叉检查，以确定是否可以从给定的context中推断出它。

**（2）答案相关性(Answer relevancy)**

评估指标“答案相关性”重点评估生成的答案(answer)与用户问题(question)之间相关程度。不完整或包含冗余信息的答案将获得较低分数。该指标是通过计算question和answer获得的，它的取值范围在 0 到 1 之间，其中分数越高表示相关性越好。

当答案直接且适当地解决原始问题时，该答案被视为相关。重要的是，我们对答案相关性的评估不考虑真实情况，而是对答案缺乏完整性或包含冗余细节的情况进行惩罚。为了计算这个分数，LLM会被提示多次为生成的答案生成适当的问题，并测量这些生成的问题与原始问题之间的平均余弦相似度。基本思想是，如果生成的答案准确地解决了最初的问题，LLM应该能够从答案中生成与原始问题相符的问题。

**（3）上下文精度(Context precision)**

上下文精度是一种衡量标准，它评估所有在上下文(contexts)中呈现的与基本事实(ground-truth)相关的条目是否排名较高。理想情况下，所有相关文档块(chunks)必须出现在顶层。该指标使用question和计算contexts，值范围在 0 到 1 之间，其中分数越高表示精度越高。

**（4）上下文召回率(Context recall)**

上下文召回率(Context recall)衡量检索到的上下文(Context)与人类提供的真实答案(ground truth)的一致程度。它是根据ground truth和检索到的Context计算出来的，取值范围在 0 到 1 之间，值越高表示性能越好。

为了根据真实答案(ground truth)估算上下文召回率(Context recall)，分析真实答案中的每个句子以确定它是否可以归因于检索到的Context。 在理想情况下，真实答案中的所有句子都应归因于检索到的Context。

**（5）上下文相关性(Context relevancy)**

该指标衡量检索到的上下文(Context)的相关性，根据用户问题(question)和上下文(Context)计算得到，并且取值范围在 (0, 1)之间，值越高表示相关性越好。理想情况下，检索到的Context应只包含解答question的信息。 我们首先通过识别检索到的Context中与回答question相关的句子数量来估计 |S| 的值。

**6.商业落地实施RAG工程的核心步骤**

1.数据集的准备（语料）

文档结构化处理：采用现代的智能文档技术

2.测试集的准备（QA对）

使用主流的 LLM 模型来根据文档来生成 QA 对

3.技术选型

NativeRAG

GraphRAG

AgenticRAG

4.构建知识库

5.测试和优化

根据不同的阶段来进行优化处理

数据预处理，结构化处理

切片策略

召回策略

重排序

RAFT

6.最终效果评估

Ragas来进行RAG性能的评估

7.生产环境部署

本地模型部署 vLLM