RAG（检索增强生成）

定义与核心思想

- 基本概念
  - RAG 是 Retrieval-Augmented Generation 的缩写，指一种将外部知识检索与大语言模型（LLM）生成能力相结合的混合架构。
  - 其核心目标是弥补纯生成式模型在事实准确性、时效性和领域专业性上的不足，通过实时引入可信外部文档提升回答质量。
- 与传统LLM的区别
  - 纯LLM（如GPT-3、ChatGLM2）仅依赖训练时固化于参数中的知识，无法访问训练后发生的新事件或私有数据。
  - RAG 模型在每次推理时动态检索相关文档片段（如维基百科段落、企业内部手册、最新财报PDF），再将检索结果作为上下文输入LLM生成答案。

技术架构

- 检索模块（Retriever）
  - 向量检索
    - 使用嵌入模型（如bge-small-zh、text2vec-large-chinese）将用户查询和文档块编码为稠密向量。
    - 在向量数据库（如Milvus、Chroma、Weaviate）中执行近似最近邻搜索（ANN），返回Top-K语义最相关的文本块。
    - 示例：用户提问“iPhone 15 Pro的钛金属边框有何优势？”，检索器从苹果官网技术文档中召回含“钛合金”“重量减轻”“强度提升”等关键词的段落。
  - 关键词/稀疏检索（可选补充）
    - 结合BM25算法对原始文本进行词频-逆文档频率加权匹配，用于增强对术语精确匹配的鲁棒性。
    - 常与向量检索融合（如Reciprocal Rank Fusion, RRF）以平衡语义相关性与字面匹配。
- 生成模块（Generator）
  - 条件化生成
    - 将检索到的文档块（通常截断为512–1024 token）与用户原始问题拼接为新提示（Prompt），格式如：“根据以下资料回答问题：[检索文本1] [检索文本2] … 问题：{用户问题}”
    - 调用冻结或微调后的LLM（如Qwen2-7B、Llama3-8B）进行条件生成，确保输出严格基于所提供证据。
  - 事实约束机制
    - 可引入引用标注（Citation）、不可生成未见信息（hallucination suppression）等策略。
    - 示例：当检索结果中未提及“iPhone 15 Pro支持Wi-Fi 6E”，模型应拒绝回答或明确声明“资料未提供该信息”。

典型应用场景

- 企业知识管理
  - 将RAG部署于内部文档系统，支持员工快速查询HR政策、IT运维手册、销售合同模板。
  - 实例：平安保险使用RAG接入《车险理赔操作指引V3.2》《反洗钱合规手册2024》，客服人员提问“异地出险如何定损？”时，系统自动定位条款第4.7节并生成结构化答复。
- 科研与教育
  - 构建论文辅助系统，从arXiv、CNKI等平台实时检索最新研究，辅助文献综述或实验设计。
  - 实例：BioMedRAG系统接入PubMed摘要库，研究人员提问“CRISPR-Cas12a在植物基因编辑中的脱靶率是否低于Cas9？”，返回3篇2023年对比实验论文的关键结论。
- 客服与政务
  - 对接政府公开文件（如国务院政策文件库、地方政府办事指南），实现政策精准解读。
  - 实例：上海“随申办”APP集成RAG，市民提问“灵活就业人员如何缴纳医保？”，系统从《上海市灵活就业人员参加医疗保险办法（沪医保规〔2022〕5号）》中提取参保条件、缴费基数、办理流程三要素生成回答。

关键挑战与优化方向

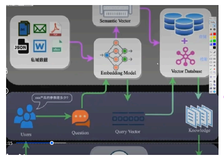
- 检索质量瓶颈
  - 语义鸿沟：查询简短（如“糖尿病用药”）vs 文档表述复杂（如“二甲双胍单药治疗2型糖尿病患者的HbA1c达标率分析”）。
  - 优化方案：采用查询扩展（Query Expansion）技术，如基于同义词词林或大模型重写查询（“糖尿病常用口服降糖药及其适用人群”）。
- 生成可靠性风险
  - 信息过载：过多检索片段导致LLM注意力分散，混淆关键证据。
  - 优化方案：实施检索结果重排序（Re-ranking），使用Cross-Encoder模型（如bge-reranker-base）对初始Top-20结果进行精细化打分并截取Top-3。
- 工程落地难点
  - 文档预处理复杂性：PDF表格识别错误、扫描版OCR噪声、多语言混排导致切块失真。
  - 实例：某银行将2000份PDF信贷合同导入RAG系统时，需定制规则过滤页眉页脚、合并跨页表格、标准化“年利率”“APR”“月息”等异构术语表达。

主流开源工具链

- 检索基础设施
  - Chroma：**轻量级嵌入存储**，支持Python SDK快速启动，适合中小规模知识库。
  - Milvus：**云原生向量数据库**，支持亿级向量毫秒检索，适配金融、电商高并发场景。
- 端到端框架
  - LlamaIndex：提供数据连接器（PDF/Notion/Slack）、索引构建、查询引擎一体化API。
  - **LangChain：模块化设计，可自由组合Retriever、LLM、OutputParser，支持自定义Agent工作流。**
- 中文专项优化
  - BGE系列模型（智谱AI）：bge-m3支持多语言、多粒度（段落/句子/词）、多任务（检索/重排/分类）**统一嵌入。现在Qwen3更加常用。**
  - Qwen2-RAG：通义千问团队发布的RAG专用微调版本，在中文法律、医疗领域检索准确率较通用bge提升12.6%（MTEB-CN榜单）。

直观理解RAG

- 形成更完整的输入提示



- 外挂一个数据库
  - 数据准备
    - 各种格式都行
    - 组织（杂乱无章到重点）-分割（每1000字分割一下处理）数据
    - Langchain工具
      -把数据自动化做结构化过程
  - Embedding比较贵一点
    - openai
    - m3e
  - 向量数据库
    - Faiss
- 为什么要做
  - 微调问题
    - 能力不足：数据质量/算力不够
    - 实时问题难解决
  - 原则上的问题
    - 公司出于安全不允许上传GPT等，需要本地数据库
    - LLM幻觉问题
    - 局限性
- RAG现存问题是什么
  - 召回率低（在本地文档找不到相关答案或者匹配错误答案）
    - 本地数据质量有问题
    - 数据极其相似
    - 具体行业问题
  - 解决方案
    - **从业务场景上去做尝试**而非技术上
    - **数据切片**
      - 长度，重合，细致等
      - 结合数据特点去设计
    - **继续借助LLM**
      - 重新整合找到的数据，embedding之后生成得分，再根据生成得分的权重（找到更相关的），生成合适的提示
      - 更结构化/标准化
    - **多分支结构**
      - 必须要精确回答的问题时
      - 直接在本地数据库匹配答案
      - 不敏感的信息
      - 前处理，结合记忆做匹配
      - 数据库/数据表分级
      - 问的问题不同，访问的表也不同，不能去一个大库去找
    - **关键词**
      - 找的更全
      - 递归（类似于参考文献的参考文献这样找下去），最后合成摘要作为输入
    - **融入更多的信息**
      - 比如：personal title，可以找与个人信息、组织结构、关系网络相关的所有信息融入进来
    - **知识图谱**
      - 本地数据处理成QA对
      - 先要LLM对数据做结构化抽取，需要什么字段，自动生成KG图谱，从图谱中抽取很多QA对，回答的时候如果有QA对相关的，先走QA对的，没有的话再走其他方法
- 评估RAG
  - Evaluating RAG applications with RAGAs
- 数据库、数据表工具包
  - SuperDB

Presented with xmind