RAG 知识讲解

2025.10.26

本次讲解将围绕 What（是什么）、Why（为什么） 和 How（怎么做） 三个方面展开，系统介绍 RAG 的基本概念与原理。本文档内容侧重基础，旨在帮助所有同学快速入门 RAG 相关知识，为后续深入学习更高级的 RAG 技术奠定坚实基础。

一、什么是 RAG（What）

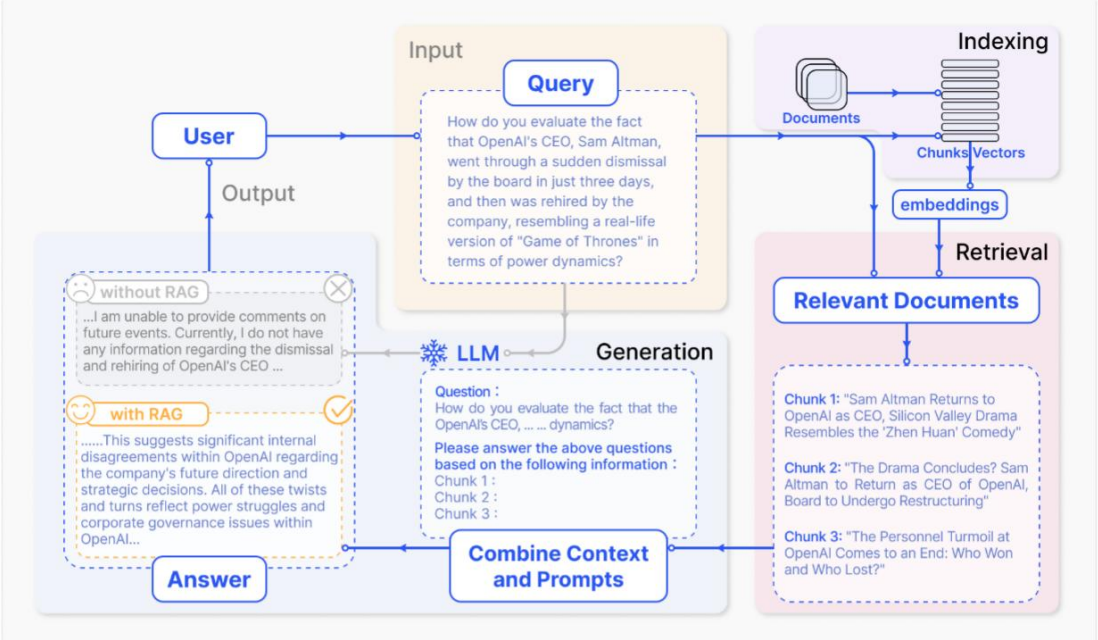
RAG 全名：
Retrieval-**A**ugmented **G**eneration 检索增强生成

从英文语法的角度来理解：
Retrieval-Augmented = “用检索到的信息来增强（某个过程）”。
在 RAG 的上下文中，这个“被增强的过程”就是语言模型的文本生成（Generation）。
因此，Retrieval-Augmented Generation = “用检索到的知识来增强生成过程”。

正如他的名字一样，RAG 的核心三部分：
• **Retrieveal**: Responsible for querying external data sources such as knowledge bases, APIs, or vector databases. Advanced retrievers leverage dense vector search and transformer-based models to improve retrieval precision and semantic relevance.
• **Augmentation**: Processes retrieved data, extracting and summarizing the most relevant information to align with the query context.
• **Generation**: Combines retrieved information with the LLM’s pre-trained knowledge to generate coherent, contextually appropriate responses.
参考文献：AGENTIC RETRIEVAL-AUGMENTED GENERATION: A SURVEY ON AGENTIC RAG https://doi.org/10.48550/arXiv.2501.09136

整体来说，RAG 就是由这两个关键部分组成：
real-time data retrieval + LLM （检索器 + 生成器）

RAG 的核心流程如下（如下图所示）：
用户提问 → 检索模块（从知识库找相关文档） → 生成模块（LLM + 文档 → 回答）



RAG 的发展历程

Naive RAG、Advanced RAG、Modular RAG、Graph RAG、Agentic RAG（当下最流行的）每一代都在解决上一代的瓶颈。

What began as simple keyword-based retrieval has transitioned into sophisticated, modular, and adaptive systems capable of integrating diverse data sources and autonomous decision-making processes. 最初简单的基于关键字的检索已经转变为复杂的、模块化的和自适应的系统，能够集成不同的数据源和自主决策过程。

1. Naive RAG（基础版 RAG）

最早的 RAG 形式。“检索（BM25 / TF-IDF） + 拼接文档 + 生成回答”

1.1 特点：

使用关键词匹配检索（BM25、TF-IDF）；从静态数据库中取文档；将检索结果简单拼接后输入给 LLM。

1.2 优点：

简单、易实现（非常适合教学/原型）；适用于事实型问题（如百科问答）。

1.3 缺点：

不理解语义，仅能做词面匹配；检索结果经常“偏题”；输出缺乏连贯性；对大数据集的扩展性差。

一句话总结：“会查资料，但不会理解上下文。”

2. Advanced RAG（进阶语义检索）

让模型“理解”查询内容，不再仅靠关键词。

2.1 特点：

Dense Embedding 向量检索（如 DPR、bge 等）；Reranker 重排序（排序模型如 bge-reranker）；跳步检索（multi-hop）支持复杂推理问题。

2.2 优点：

语义相关度高（理解同义词、上下文）；回答更精确、更自然；能应对复杂问题（跨文档推理）。

2.3 缺点：

计算量增大、耗时更长；大规模数据下索引和检索效率问题.

一句话总结：

“能理解语义，但还不够灵活。”

3. Modular RAG（模块化 RAG）

把 RAG 拆分成可组合的“模块”，像搭积木一样灵活配置。

3.1 特点：

模块化组件：Retriever / Reranker / Generator 可自由组合；支持 Hybrid 检索（稀疏+稠密）；可插入外部工具或 API；支持可配置流水线（Composable Pipelines）。

3.2 优点：

灵活：可替换检索器、生成器或 reranker；可拓展：适合多领域系统（如金融、医疗）；持实时数据调用（API、数据库）。

3.3 缺点：

系统设计更复杂；需要人工调优模块组合。

一句话总结：

“让 RAG 变成了一个可扩展的系统架构。”

4. Graph RAG（图结构 RAG）

把知识组织成“图”（graph），让模型能理解实体关系与层级结构。

4.1 特点：

构建知识图谱（Node + Edge）；图结构检索（Graph-based Retrieval）；关系推理（Relational Reasoning）。

4.2 优点：

理解实体之间的关系；支持复杂跳步推理（multi-hop）；减少幻觉，增强可

解释性。

4.3 缺点：

构建和维护知识图成本高；依赖高质量的结构化数据；系统复杂度高、实时性较差。

一句话总结：

“让 RAG 学会了‘关系推理’。”

5. Agentic RAG（智能体 RAG）

引入智能体（Agent），让系统能自动决策、反思、优化。

5.1 特点：

Agent 能自主决定检索策略（如检索几次、去哪查）；具有反思（Reflection）与 迭代优化（Iterative Refinement）；可组合多智能体协作（Orchestrator-Worker 模式）；动态调整工作流（如实时任务规划）。

5.2 优点：

真正动态、智能地完成复杂任务；支持多步推理、多数据源集成；高精度、高适应性。

5.3 缺点：

智能体协调复杂（需要良好调度机制）；算力消耗大（多 Agent 并发）；实现与调试难度高。

一句话总结：

“RAG 从工具升级为能思考的系统。”

简单总结一下 RAG 发展的必然性：

RAG 的发展路径体现出 AI 系统从“数据驱动 → 知识驱动 → 自主驱动”的演进。

阶段	驱动力	核心突破
Naïve / Advanced	数据增强	让 LLM“查资料”
Modular / Graph	知识结构化	让 LLM“理解知识关系”
Agentic	自主智能	让 LLM“自己决定如何查资料”

二、为什么选择 RAG（Why）

LLM 的局限性：

1. 静态知识：模型训练数据固定，无法反映最新信息（如 2025 年的新研究）；
2. 幻觉问题：模型可能生成“看似合理但错误”的内容；
3. 缺乏可验证性：无法提供来源或依据，难以追溯信息来源；
4. 缺乏领域适应：无法直接访问企业内部文档或外部实时知识。

RAG 是“**检索 + 生成**”双管齐下，本质是通过实时外部检索增强 LLM 的回答能力，让通用 LLM 在我们特定任务中表现最佳。

For example：

现在我们学习小组有一个资料库，这里面全是我们每个人整理好的学习资料。现在我想建立一个知识问答系统，去对我们资料库中的知识进行查询学习，这个时候如果我们去问 LLM，他虽然可以回答我们的问题，但得不到我们想要的答案，因为他肯定是不知道这个资料库的，他无法根据这个资料库来回答。

所以，我们要选择加入 RAG 技术来构建这个知识问答系统，相当于外挂一个知识库，让 LLM 只负责很好的润色、组织词汇和逻辑、补充一些内容即可。（你想要怎样的回答风格、样式等，可以通过自定义 prompt 实现）

三、怎么构建一个基于 RAG 的系统（How）

首先明确需求，确定[**知识库的范围与格式**]，将资料库文档进行[**分块**]处理并转化为向量存储至[**向量数据库**]。接着选择[**合适的 LLM**]作为生成模型，通过 API 或本地部署方式接入。查询时，用户问题经编码后在向量库中[**检索**]相似片段，将相关文本与问题拼接成[**新提示词**]输入 LLM，由其生成自然语言回答。

整个过程依赖高质量的检索与精准的提示设计，确保输出内容准确、连贯且源自资料库。

关于**检索器的构建**，我有话要说：

我们需要明确需求，确定知识库的范围与格式，将资料库文档进行分块处理并转化为向量存储至向量数据库。在这个过程中，可以说每一步都有自己的学问，每一步都需要踩踩雷尝试过后才知道哪个方法是最适合自己当前项目的。

有些主流的方法或许不太适用你的项目、数据，这时候需要自己好好思考一下我们手中有的数据，去自己 create 一个最合适的方法。请记住，你的数据就是你最大的财富。（knowing your domain is your biggest assest for increasing retrieval accuracy.）

这一个步骤往往是最麻烦的，因为我们需要处理数据！！！

这一步往往也会决定我们整个 RAG 系统的性能。

以我之前做过的一个知识问答 RAG 系统为例（如下图所示）：

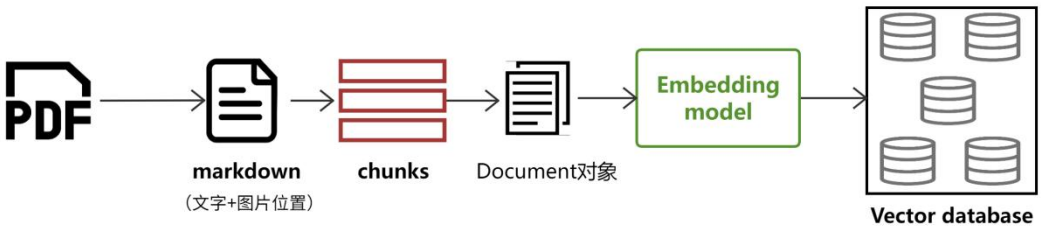
用 PaddleOCR（也有多种选择）识别并转为 markdown 格式文档；

在对文档进行切块时（Chunk），也有多种方法；

Embedding 模型我们选用的为 bge-large-zh-v1.5，也是多种选择，但 bge 这个是 top；

在检索时，也有多种选择，向量（密集检索）、bm25（稀疏检索）、混合（二者权重怎么决定也需要考虑）。

可以看到，这其中的每一个环节都需要根据实际项目去选择尝试。



下面我这里有一个十分简易但又步骤齐全的 demo 系统，我们来一起看看吧。代码已上传至 github。

补充说明：

本文档参考文献如下（推荐大家去阅读）：

1. AGENTIC RETRIEVAL-AUGMENTED GENERATION: A SURVEY ON AGENTIC RAG https://doi.org/10.48550/arXiv.2501.09136

2. Retrieval-Augmented Generation for LargeLanguage Models: A Survey https://doi.org/10.48550/arXiv.2312.10997