检索增强生成

Retrieval Augmented Generation

1 什么是RAG

场景一：你是一名大学生，正在写一篇关于人工智能的论文。你不会直接凭空写出内容，而是会先去图书馆或网上查找权威资料，把相关的信息找出来，然后根据这些资料组织语言、写作和总结。

对应到 RAG：

- 你自己 = 生成模型（比如 GPT）
- 图书馆资料 = 外部知识库（可检索的信息来源）
- 查阅资料再写作 = 检索增强生成：先“查”，再“写”

RAG 就像一个善于写作的学生，在动笔前先查阅资料，写作时参考这些内容，做到既准确又丰富。

场景二：你向手机语音助手（如 Siri 或小爱同学）提问：“明天成都的天气如何？”它不会凭记忆胡乱回答，而是会去调用天气网站的实时数据，再把答案说出来。

对应到 RAG：

- 语音助手的语言表达能力 = 生成模型（擅长用自然语言组织答案）
- 实时天气数据 = 检索到的外部信息
- 组合回答 = 把检索来的事实信息转换成自然语言结果

RAG 就像一个能说会道的助理，先查准资料，再把答案说清楚。

1 什么是RAG

1.1 基础大模型的局限性

- **检索增强生成（Retrieval Augmented Generation），简称 RAG**，已经成为当前最火热的LLM应用方案。通过自有垂域数据库检索相关信息，然后合并成为提示模板，给大模型生成漂亮的回答。
- 但是在实际业务场景中，通用基础大模型往往基本无法满足需求，主要有以下几方面原因：
  - **知识的局限性：**模型自身的知识完全源于它的训练数据，而现有的主流大模型（ChatGPT、文心一言、通义千问…）的训练集基本都是构建于网络公开的数据，对于一些实时性的、非公开的或离线的数据是无法获取到的，这部分知识也就无从具备。
  - **幻觉问题：**所有的AI模型的底层原理都是基于数学概率，其模型输出实质上是一系列数值运算，大模型也不例外，所以它有时候会一本正经地胡说八道，尤其是在大模型自身不具备某一方面的知识或不擅长的场景。而这种幻觉问题的区分是比较困难的，因为它要求使用者自身具备相应领域的知识。
  - **数据安全性：**对于企业来说，数据安全至关重要，没有企业愿意承担数据泄露的风险，将自身的私域数据上传第三方平台进行训练。这也导致完全依赖通用大模型自身能力的应用方案不得不在数据安全和效果方面进行取舍。
- 而**RAG**是解决上述问题的一套有效方案。

1 什么是RAG

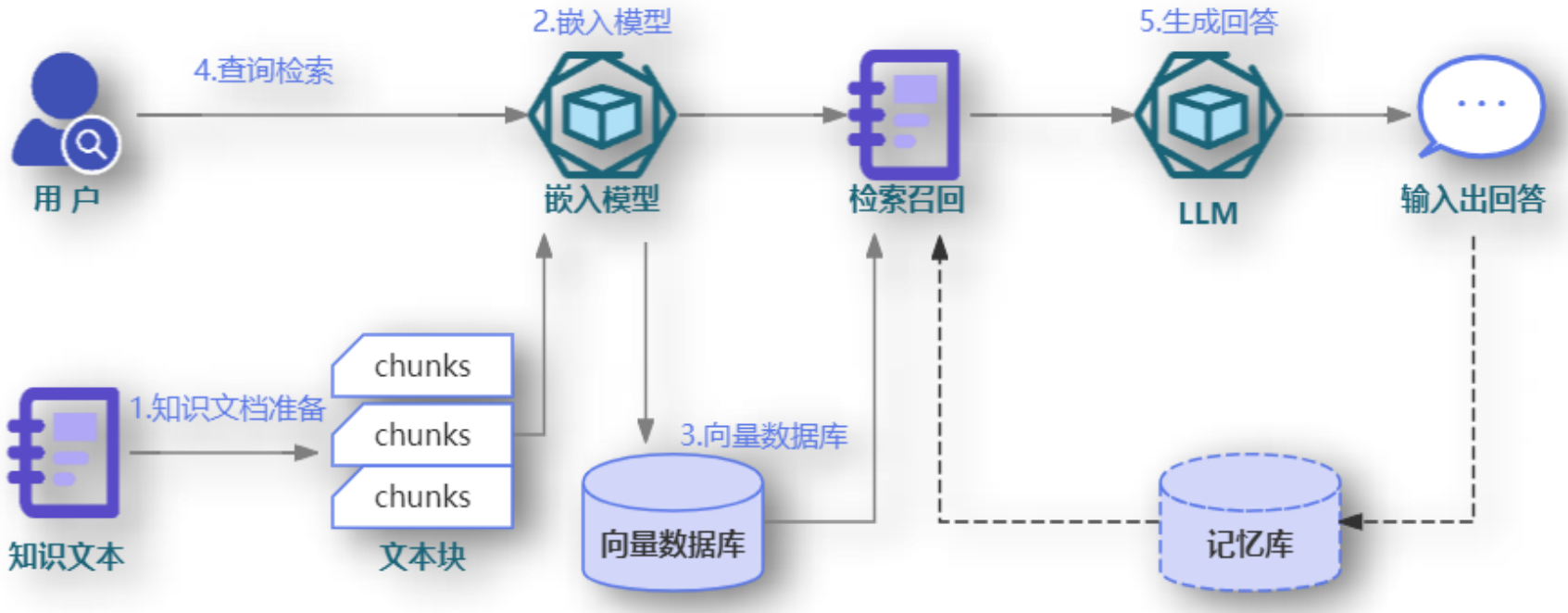
1.2 什么是RAG?

- **RAG**为生成式模型提供了与外部世界互动提供了一个很有前景的解决方案。
- **RAG的主要作用类似搜索引擎：**找到用户提问最相关的知识或者是相关的对话历史，并结合原始提问（查询），创造信息丰富的prompt，指导模型生成准确输出。其本质上应用了情境学习（In-Context Learning）的原理。
- **RAG（检索增强生成）= 检索技术 + LLM提示**
- **RAG的特点：**
  - RAG依赖大语言模型来强化信息检索和输出，单独使用，能力受限。
  - RAG能与外部数据无缝集成，较好地解决通用大模型在垂直、专业领域的知识短板。
  - 一般情况下，RAG对接的私有数据库不参与大模型数据集训练，能在改善模型性能的同时，更好地保证数据隐私和安全。
  - 同样是RAG，展现的效果并不统一，很大程度上受模型性能、外挂数据质量、AI算法、检索系统等多方面的影响。

2 RAG基本流程

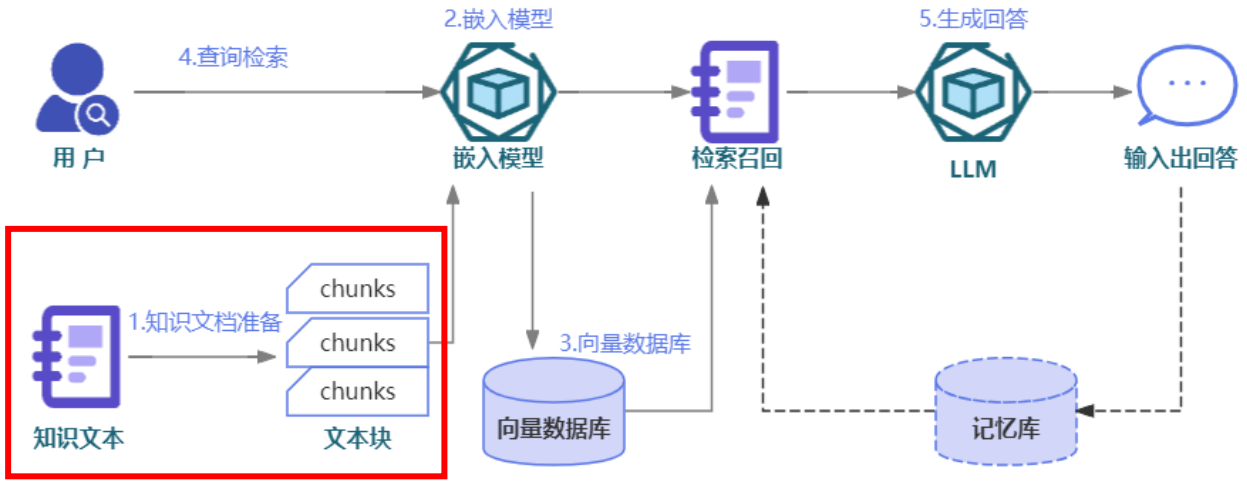
RAG包含5个基本流程：

1. 准备知识文档
2. 选择嵌入模型
3. 构建向量数据库
4. 查询检索
5. 生成回答



2 RAG基本流程

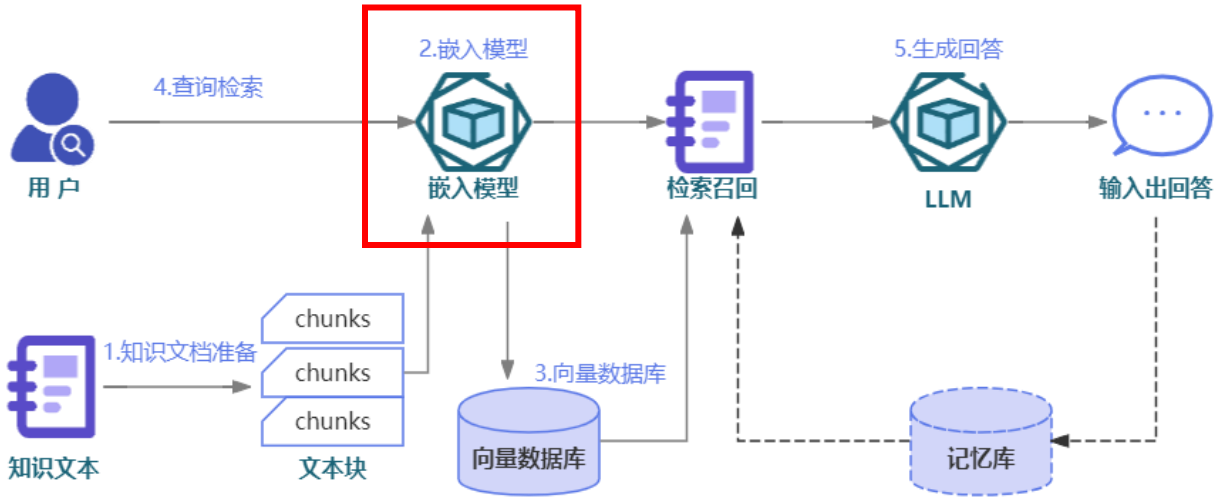
2.1 准备知识文档



- 在构建一个高效的RAG系统时，首要步骤是准备知识文档。
- 现实场景中，我们面对的知识源可能包括多种格式，如Word文档、TXT文件、CSV数据表、Excel表格，甚至是PDF文件、图片和视频等。
- 因此，第一步需要使用专门的文档加载器（例如PDF提取器）或多模态模型（如OCR技术），将这些丰富的知识源转换为大语言模型可理解的纯文本数据。
- 例如，处理PDF文件时，可以利用PDF提取器抽取文本内容；对于图片和视频，OCR技术能够识别并转换其中的文字信息。
- 此外，鉴于文档可能存在过长的问题，我们还需执行一项关键步骤：**文档切片**。我们需要将长篇文档分割成多个文本块，以便更高效地处理和检索信息。这不仅有助于减轻模型的负担，还能提高信息检索的准确性。

2 RAG基本流程

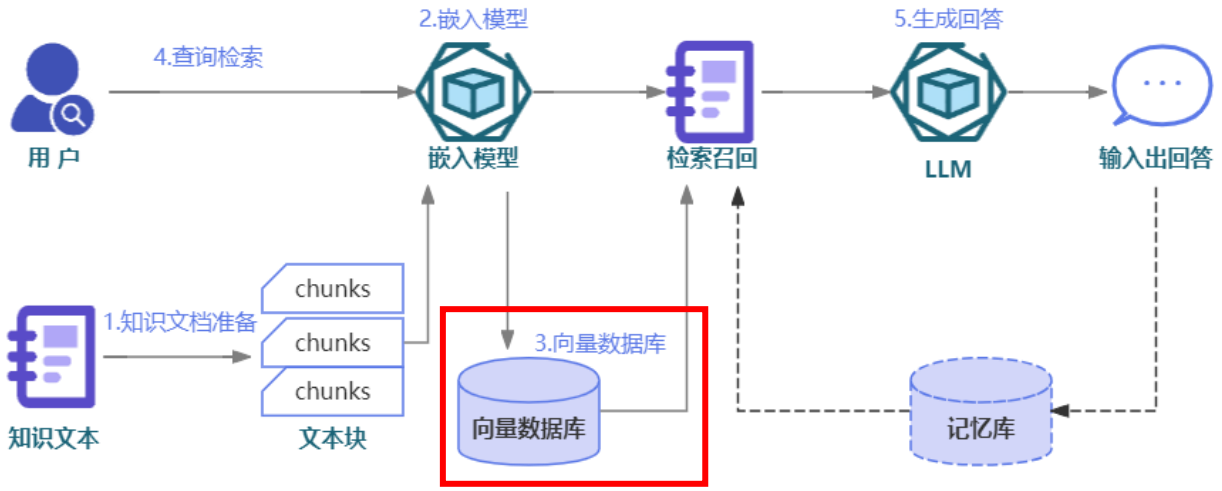
2.2 选择嵌入模型



- 嵌入模型的核心任务是将文本转换为向量形式。
- 我们使用的日常语言中充满歧义和对表达词意无用的助词，而向量表示则更加密集、精确，能够捕捉到句子的上下文关系和核心含义。
- 这种转换使得我们能够通过简单计算向量之间的差异来识别语义上相似的句子。嵌入模型是连接用户查询和知识库的桥梁，确保了系统回答的准确性和相关性。

2 RAG基本流程

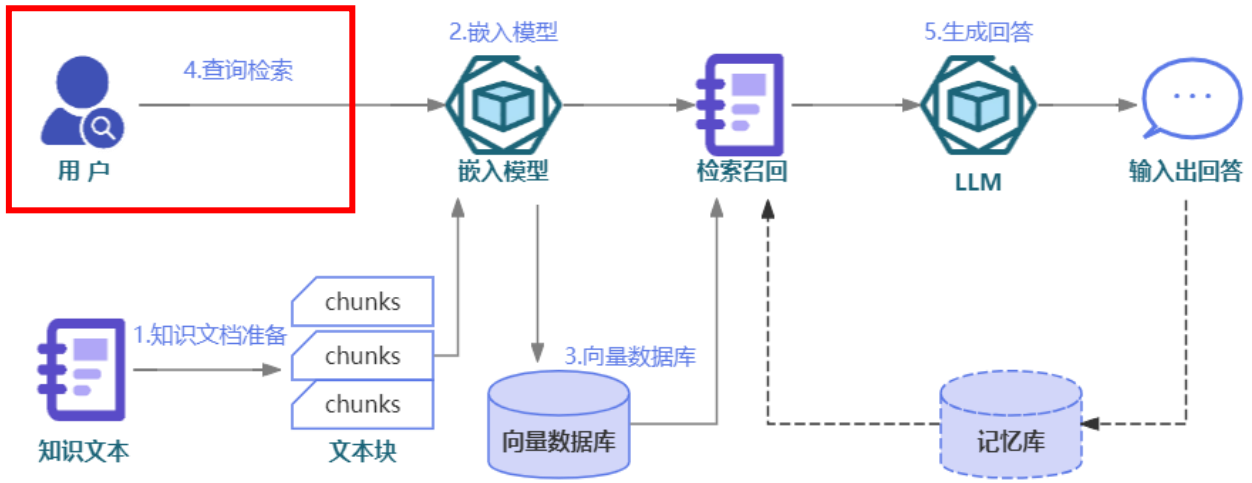
2.3 构建向量数据库



- 向量数据库：顾名思义，是专门设计用于存储和检索向量数据的数据库系统。
- 在RAG系统中，通过嵌入模型生成的所有向量都会被存储在这样的数据库中。
- 这种数据库优化了处理和存储大规模向量数据的效率，使得在面对海量知识向量时，我们能够迅速检索出与用户查询最相关的信息。

2 RAG基本流程

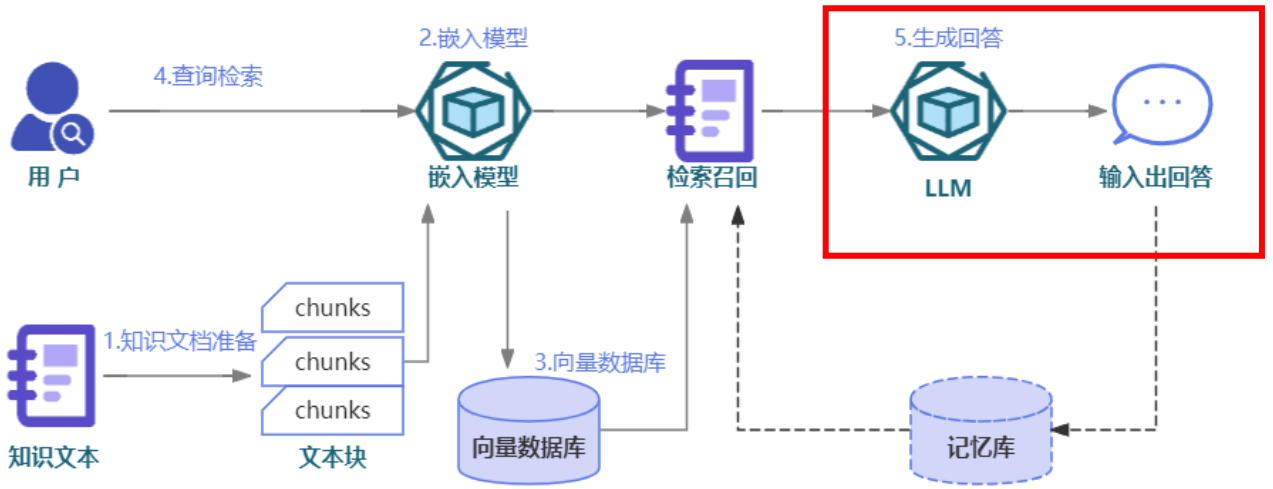
2.4 查询检索



- 再经过上述几个步骤的准备后，我们就可以开始进行处理用户查询了。
- 首先，用户的问题会被输入到嵌入模型中进行向量化处理。
- 然后，系统会在向量数据库中搜索与该问题向量语义上相似的知识文本或历史对话记录并返回。

2 RAG基本流程

2.5 生成回答



- 最终将用户提问和上一步中检索到的信息结合，构建出一个提示模版，输入到大语言模型中，静待模型输出答案即可。

3 RAG优化策略

3.1 知识文档准备阶段

3.1.1 数据清洗

- **基本文本清理：**规范文本格式，去除特殊字符和不相关信息。去除重复文档或冗余信息。
- **实体解析：**消除实体和术语的歧义以实现一致的引用。例如，将“LLM”、“大语言模型“和“大模型”标准化为通用术。
- **语文档划分：**合理地划分不同主题的文档，不同主题是集中在一处还是分散在多处?如果作为人类都不能轻松地判断出需要查阅哪个文档才能来回答常见的提问，那么检索系统也无法做到。
- **数据增强：**使用同义词、释义甚至其他语言的翻译来增加语料库的多样性。
- **用户反馈循环：**基于现实世界用户的反馈不断更新数据库，标记它们的真实性。
- **时间敏感数据：**对于经常更新的主题，删除过期的文档、或者对过期的文档更新。

3 RAG优化策略

3.1 知识文档准备阶段

3.1.2 文本分块

- **固定长度分块：**简单直接，计算需求低。保留块之间的重叠，确保块之间上下文语义不会丢失。
- **内容分块：**根据文档的具体内容、标点符号（句号）进行分块。
- **递归分块：**通过重复地应用分快规则来递归地分解文本。例如先按照段落分隔，检测块的大小是否超过阈值，若超出则进一步按换行符、句子、空格逐级分隔。
- **从小到大分块：**把文档进行从大到小所有尺寸进行分割，并保留层级结构。
- **特殊结构分块：**对Markdown文件、Latex文档及各种主流代码语言分割器。

3 RAG优化策略

3.2 嵌入模型阶段：嵌入模型选择

- **嵌入模型的作用：**将文本映射为向量，便于后续的相似度计算、分类或其他下游任务。
- **静态词向量 vs. 动态词向量**
  - Word2Vec 等静态模型：训练完成后，每个词的向量固定不变，无法区分同形异义词。
  - BERT 等基于自注意力的模型：根据上下文动态生成词向量，同一个词在不同语境下可有不同表示。

    “我买了一张*光盘*” → 指圆形盘片

    “*光盘*行动” → 指把餐盘里的食物吃光
  - 静态词向量无法区分，动态词向量可根据句意分别表示。
- **选择嵌入模型的建议**
  - 参考 Hugging Face 发布的 MTEB 排行榜，对比各模型在不同任务上的性能
  - 确认所选模型是否支持中文或其他目标语言，查阅模型说明文档。
- **垂直领域微调的权衡**
  - 优点：理论上可让模型更好地理解行业术语
  - 缺点：对高质量标注数据要求高、需要投入大量人力物力，且实际效果未必显著，可能得不偿失

3 RAG优化策略

3.3 向量数据库阶段：将向量数据与元数据结合

- **元数据（Metadata）：**“描述数据的数据”，是对主数据（Primary Data）属性、结构或上下文信息的补充说明，用于管理、组织和检索数据。元数据本身不承载业务内容，而是描述这些内容的相关信息。
- **元数据的主要类型**
  - 描述性元数据（Descriptive Metadata）：用于标识和检索主数据，通常包括标题、摘要、关键词、作者等。
  - 结构性元数据（Structural Metadata）：描述主数据的内部结构或组织方式，如章节目录、页面顺序、文件夹层级、数据库表和字段结构。
  - 管理性元数据（Administrative Metadata）：支持数据管理与维护，包含创建日期、修改日期、访问权限、格式、版本号和版权信息等。
- **向量与元数据结合：**在向量数据库中，不仅可以存储向量，还可将非向量化的“元数据”与之一同保存，以增强检索能力。
- **元数据的作用**
  - 标注向量，辅助在海量向量中进行快速筛选；
  - 在处理搜索结果时，为排序和过滤提供依据。

3 RAG优化策略

3.4 查询检索阶段

- **背景问题：**
  - 元数据难以有效区分不同查询上下文
  - 单一索引难以满足多样化信息需求
- **多重索引技术：**
  - 将信息需求按类别划分，建立多个专用索引：
    - 摘要类问题索引、具体答案索引、时间敏感问题索引
  - 系统根据查询性质选择最合适的索引
- **多级路由机制：**
  - 查询根据复杂度、信息类型等特点自动路由
  - 精准匹配至单个或组合索引
  - 优化资源分配与处理效率

3 RAG优化策略

3.5 生成回答阶段

- **大语言模型（LLM）：**生成回答的核心环节，根据检索到的上下文生成最终回答。
- **选择 LLM 的考量因素**
  - 模型开放性：开源模型 vs 专有模型
  - 推理成本：计算资源与响应速度
  - 上下文长度：可处理的最大文本长度
- **RAG 系统搭建框架**
  - LlamaIndex：强大的数据索引和检索工具
  - LangChain：多功能框架，使开发人员能够高效地设计LLM应用
- **框架调试与可视化工具**
  - 定义回调函数（Callbacks）
  - 查看模型使用了哪些上下文片段
  - 检查检索结果来自哪些文档