基于InternLM和LangChain搭建你的知识库

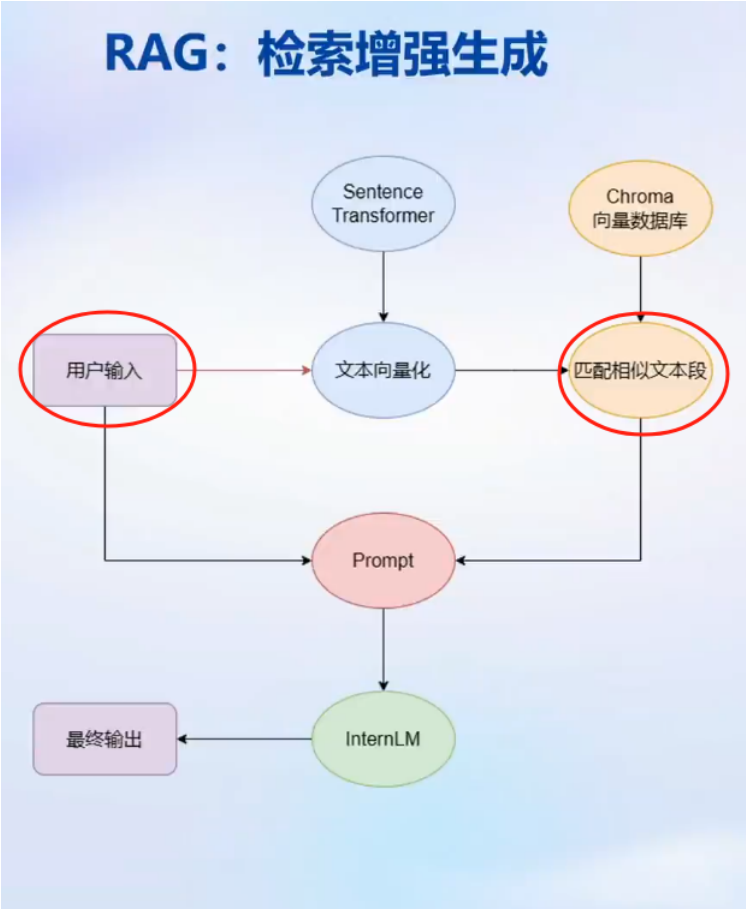
官方文档：InternLM/tutorial/langchain/readme.md

官方视频：基于 InternLM 和 LangChain 搭建你的知识库 bilibili

1. 大模型开发范式

1. LLM的局限性
    - 知识时效性受限：如何让LLM能够获取**最新**的知识
    - 专业能力有限：如何打造**垂直领域**大模型
    - 定制化成本高：如何打造**个人专属**的LLM应用
2. RAG
    - **低**成本
    - 可**实时**更新
    - 受**base-model**影响大
    - 单次回答知识**有限**
3. Finetune
    - 可**个性化**微调
    - 知识覆盖面**广**
    - 成本**高昂**
    - **无法**实时更新

2. RAG原理

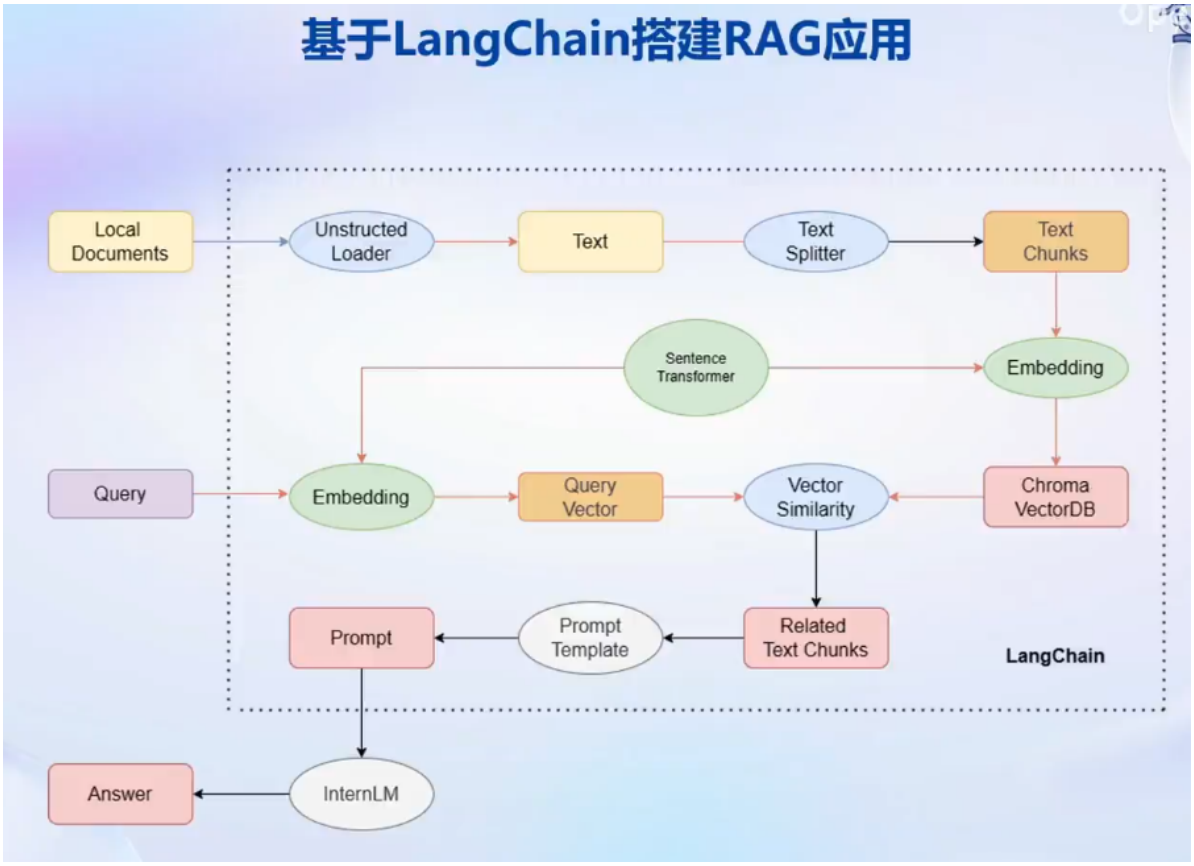


将原本的**用户输入**结合**检索到的相似文本段**，**共同**作为模型输入的prompt

3. LangChain简介

为各种LLM提供通用接口来简化应用程序开发流程，帮助开发者自由构建LLM应用

- 核心组成模块
    - Chains: 将组件组合实现端到端应用，通过一个对象封装实现一系列LLM操作；
    - Eg. 检索问答链，覆盖实现了RAG的全部流程。



上述过程都可封装

4. 构建向量数据库

- **确定源文件类型**，针对不同类型源文件选用不同的加载器
    - 核心在于将带格式文本转化为**无格式字符串**
- 由于单个文档往往超过模型上下文上限，我们需要对加载的文档进行**切分为多个chunk**
    - 一般按字符串长度进行分割
    - 可以手动控制分割块的长度和重叠区间长度
- 使用向量数据库来支持语义检索，需要将**文档向量化存入向量数据库**
    - 可以使用任意一种Embedding模型来进行向量化
    - 可以使用多种支持语义检索的向量数据库，一般使用轻量级的Chroma

5. RAG优化建议

- 基于RAG的问答系统性能核心受限于：
    - 检索精度
    - Prompt性能
- 一些可能的优化点
    - 检索方面

        - 基于语义（而不是字符长度）进行chunk分割，保证每一个chunk的语义完整
        - 给每一个chunk生成概括性索引，检索时匹配索引（而不是全文）
    - prompt
        - 迭代优化prompt策略