**一、RAG 实现流程**

**1.1 数据准备与解析**

**a. 文档上传与存储**

- 用户通过 `/v1/document` 接口上传文档（支持 PDF、Word 等格式）。

- 文档元信息（如标题、类别、知识库 ID、文件路径等）存入 MySQL/SQLite 数据库。

- 文件内容保存到本地目录。

**b. 内容解析与分块**

- 使用 `pdfplumber` 解析 PDF 文件，提取每一页的文本内容。

- 对每页文本进行分块（chunk），支持重叠分块（overlap）以保持上下文连贯。

- 每个文本块都会生成嵌入向量（embedding），并存入 Elasticsearch 的 `chunk\_info` 索引中。

**c. 元数据存储**

- 文档的摘要信息（如前几页内容）存入 Elasticsearch 的 `document\_meta` 索引。

**1.2 检索阶段**

**a. 用户提问**

- 用户通过 `/chat` 接口提问，提问内容传入 `RAG().chat\_with\_rag()` 方法。

**b. 混合检索（Hybrid Search）**

1. **全文检索（BM25）**：使用 Elasticsearch 的 `match` 查询进行关键词匹配。

2. **语义检索（向量检索）**：使用 SentenceTransformer 对用户问题编码，在 Elasticsearch 中进行 KNN 检索。

3. **结果融合（RRF）**：将两种检索结果按 Reciprocal Rank Fusion 算法融合，得到最终候选片段列表。

**c. 重排序（Rerank）（可选）**

- 使用 `bge-reranker-base` 模型对候选片段进行重排序，进一步提升相关度。

**1.3 生成阶段**

**a. 构建提示词（Prompt）**

- 使用预定义的模板 `BASIC\_QA\_TEMPLATE`，将当前时间、检索到的相关文档片段和用户问题组合成提示词。

**b. 调用大模型生成回答**

- 使用 OpenAI API（或本地 DeepSeek 模型）生成回答。

- 支持多轮对话，历史对话记录会作为上下文传入。

**c. 返回结果**

- 将生成的回答返回给用户，并记录对话状态。