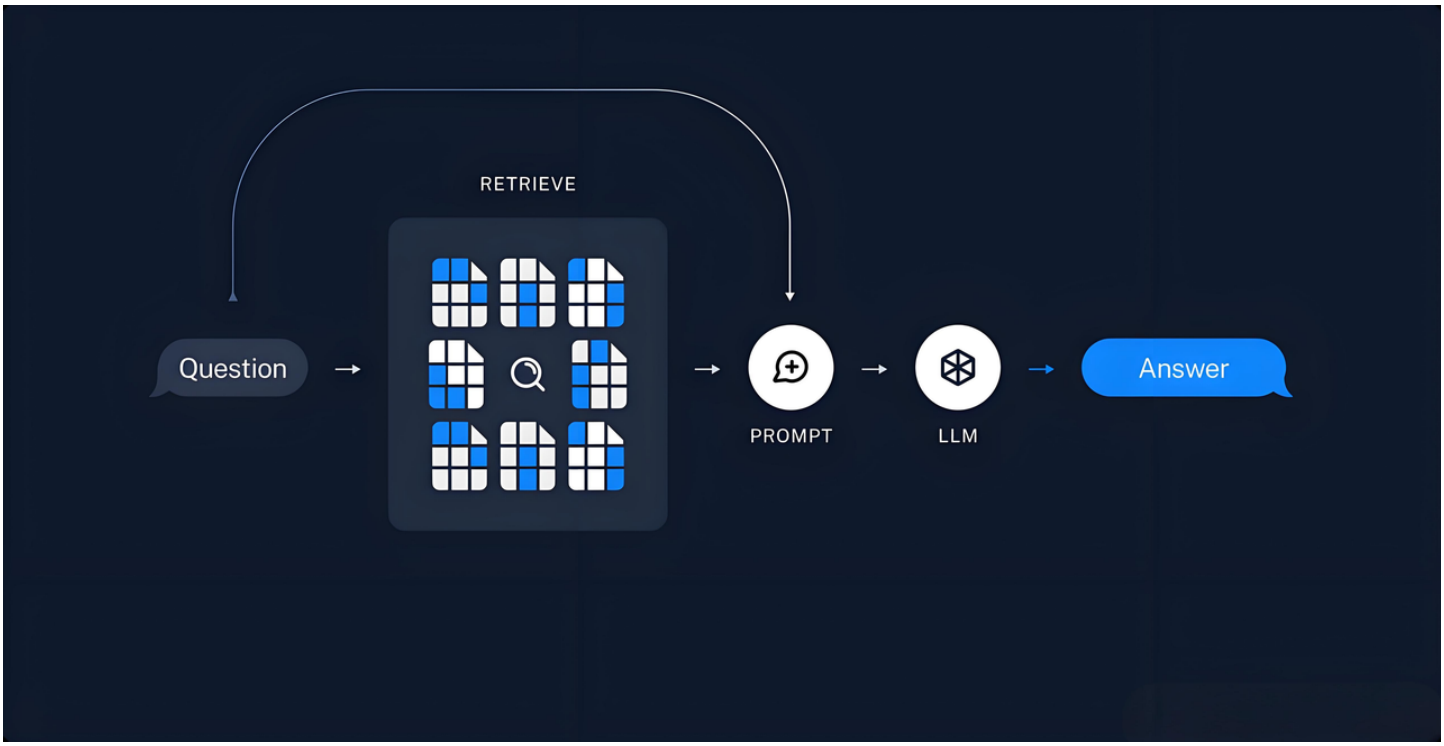
一文让你由浅到深搞懂RAG实现

1. RAG简介

RAG（检索增强生成）是一种结合检索与生成的技术，通过实时检索外部知识库中的信息，动态增强大语言模型（LLM）的生成能力，是用于解决幻觉问题、知识时效问题、超长文本问题等各种大模型本身制约或不足的必要技术。



RAG核心流程包括：

1. **问题理解阶段：**对用户问题进行改写、扩写和重构，让用户问题更利于检索；
2. **检索召回阶段：**从外部知识库（如企业文档、行业数据库）中筛选与用户问题相关的片段，并将检索结果与原始问题整合为增强提示词，输入给LLM；
3. **答案生成阶段：**LLM基于增强后的提示词，生成精准、可靠的答案。

RAG具有以下优点：

- **实时性**：动态接入最新数据（如新闻、更新文档），突破模型训练数据的时间限制；
- **专业性**：通过连接领域知识库（如医疗指南、法律条文）提升回答的准确性；
- **成本效益**：无需全量微调模型，仅维护轻量级知识库即可扩展能力；
- **可信度增强**：生成答案基于可追溯的检索证据，减少“一本正经胡说八道”的幻觉风险。

但是，不少人提出RAG的“原罪”--**一周出demo，半年用不好**。这里主要存在如下5方面的问题：

问题1：内容缺失

当用户问题不清晰或者知识库未及时更新导致检索内容缺失，LLM可能会无中生有地回答问题。

例如：用户询问“2025年杭州新能源车补贴政策”，但文档库仅更新至2024年数据，此时LLM却生成“2025年补贴政策将延续现有标准”的错误答案。

问题2：未正确排序

问题的答案在知识库中，但由于包含答案的文档没有被正确的排序，导致没有在被召回的前K个文档中（K在实际操作中不可能被设置的无限大，会基于大模型能力、成本等因素折中选择一个值）。

例如：用户查询“阿司匹林与布洛芬的副作用差异”，知识库中某篇研究论文第8页明确对比了两者差异，但该文档在检索排序中位列第12名（K=10），导致未被召回。

问题3：未在上下文中

整合策略局限性。从知识库中检索到了包含答案的文档，但在生成答案的过程中，这些文档并未被纳入上下文。当返回许多文档时，会进行文档整合以获取答案，此时会发生这种情况。

例如：用户提问“青藏高原年平均降水量”，检索到一篇地理研究报告包含该数据（第3章第2节），但系统整合时仅保留第1章气候特征概述，导致答案缺失。

问题4：未提取答案

包含答案的文档存在于上下文中，但LLM未能提取出正确的答案。通常，这是因为上下文中存在太多噪声或矛盾信息。简而言之，检索是正确的，但是LLM根据检索的文档回答问题出错。睁眼说瞎话的概率明显大于用户可以接受的概率（用户一般只能接收0.1%的错误概率）。

例如：上下文明确标注“2024年全球光伏装机容量为550GW”，但LLM受其他段落“2030年预测达1200GW”干扰，错误回答“当前装机容量接近1000GW”。

问题5：答案不完整

包含答案的文档存在于上下文中，但是LLM回答时也可能遗漏一些关键信息。

例如：上下文包含“火山喷发原因包括板块运动、岩浆房压力、地下水汽化”，LLM仅回答“板块运动导致喷发”。

2. RAG分类

RAG的特点是依赖外部数据来准确解决用户提出的问题，因此根据与外部数据的交互复杂度和认知处理深度，当前主流研究将RAG系统划分为以下四个层次：

层次1：显性事实查询

此类用户问题是外部数据中直接存在的明确事实，无需任何额外推理。

例如：根据给定的一系列关于奥运会的文档，回答：2024年夏季奥运会将在哪里举行？

层次2：隐性事实查询

此类用户问题是外部数据中的隐性事实，这些事实不是显而易见的或者分布在多个文档片段中，可能需要一定程度的常识推理或基本的逻辑推理。

例如：根据给定的一系列医疗记录，回答：最常提及的前3个症状是什么？

层次3：可解释原理查询

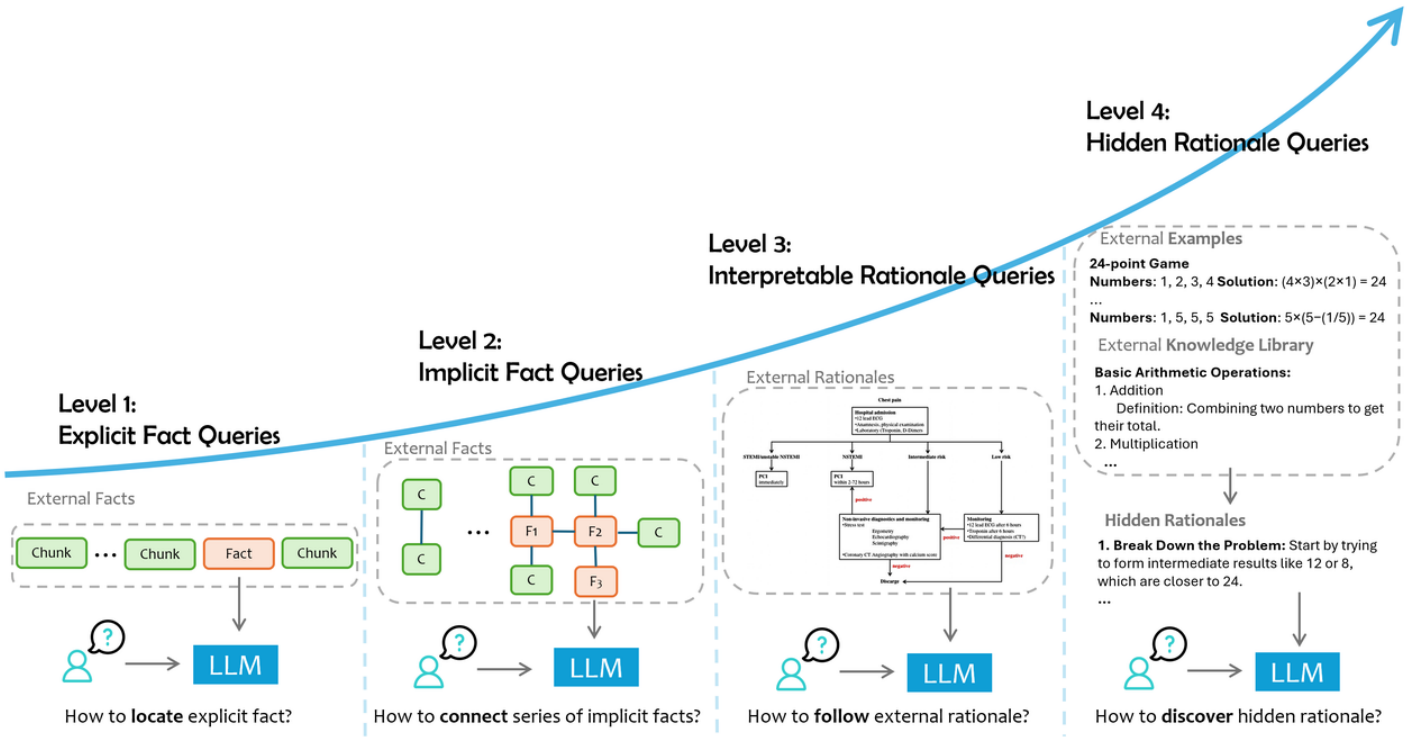
对此类用户问题的回答，不仅要求掌握事实内容，还需要理解和应用与数据上下文密不可分的领域特定原理的能力。这些原理通常在外部资源中明确提供，在一般大型语言模型的预训练阶段通常不存在或很少遇到。

例如：根据给定的胸痛管理指南，回答：应该如何诊断和治疗有特定症状描述的胸痛患者？

层次4：隐式原理查询

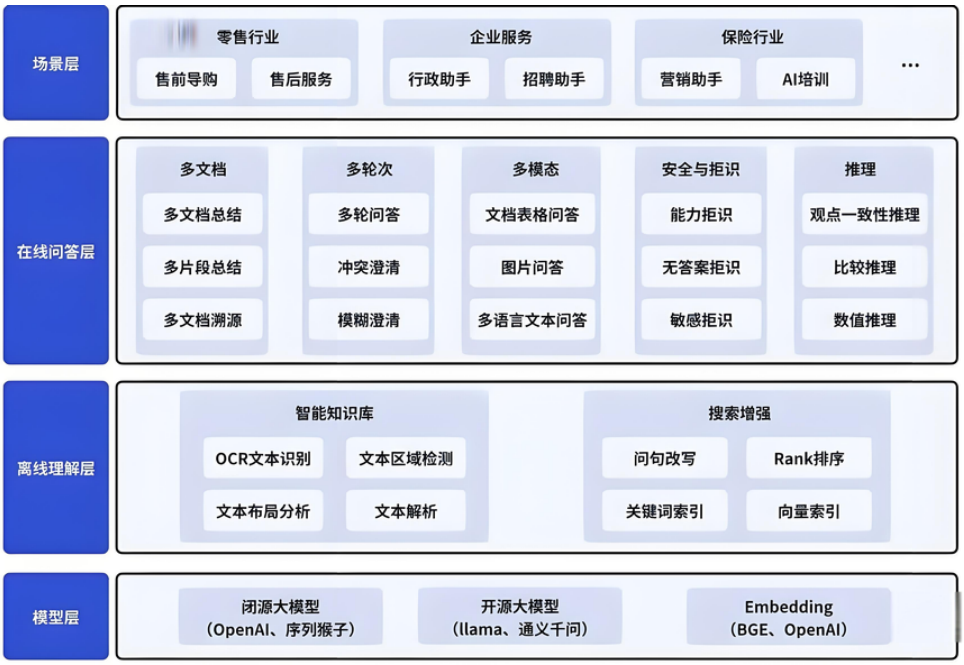
对此类用户问题的回答涉及更具挑战性的领域，其中原理没有明确记录，但必须从外部数据中观察到的模式和结果中推断出来。

例如：根据给定的一系列财务报告，回答：经济形势将如何影响公司未来的发展？



参考论文：https://arxiv.org/html/2409.14924?_immersive_translate_auto_translate=1

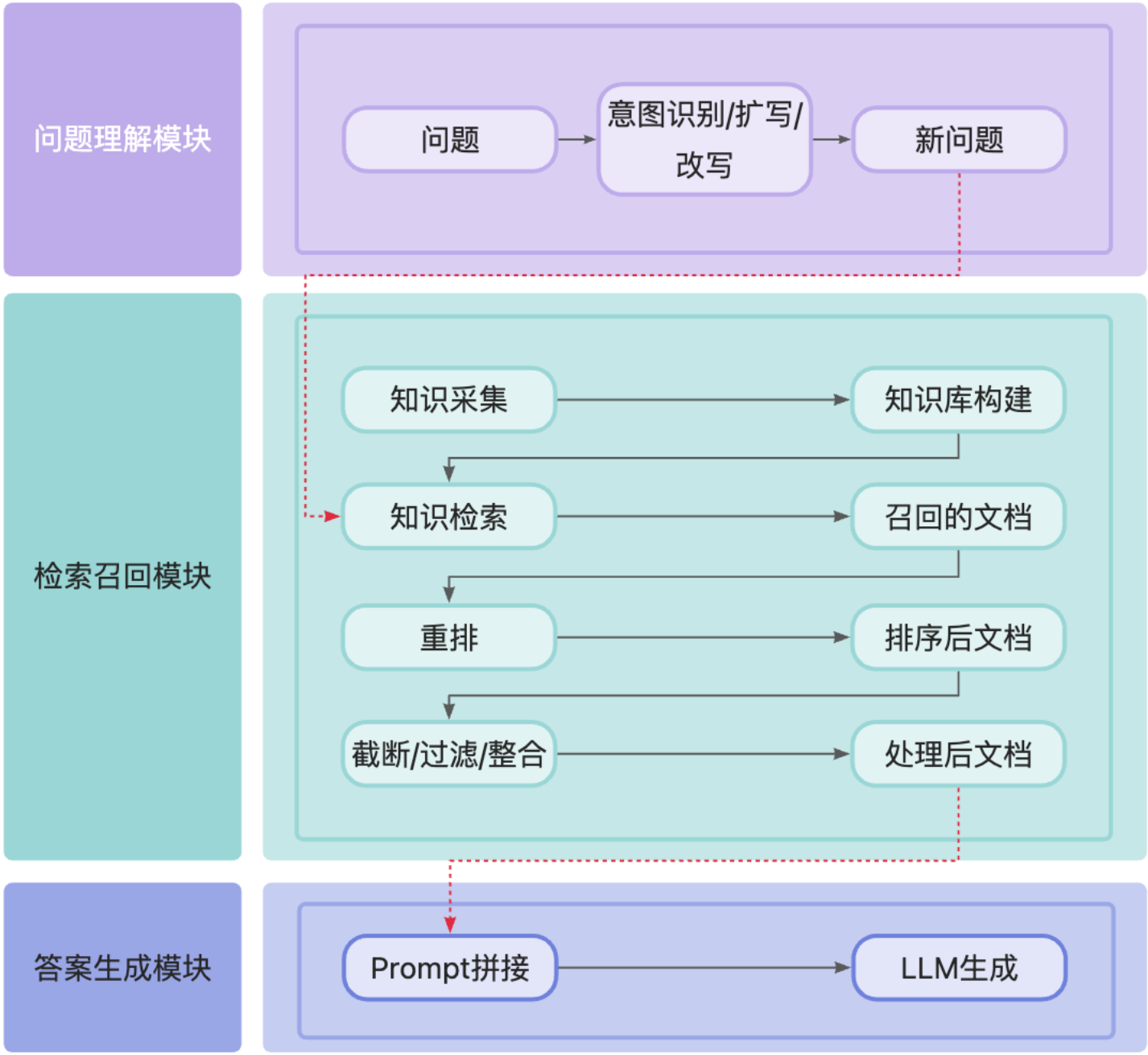
3. 产品架构



RAG产品架构包含四层：

1. **模型层：**在模型层屏蔽掉了模型的差异，不仅可以支持自有私域的LLM，也可以支持线上商用大模型，第三方的模型。此外，embedding的优化，有效的解决语言表达差异导致的检索问题，同时提高了模型的效果；
2. **离线理解层：**该层主要围绕知识库和搜索增强两个模块设计的。关于智能知识库主要负责将非结构化的文本进行处理，从而转化为检索知识库，主要包括文本解析，表格读取，OCR识别等。搜索增强通过引入问句改写、重排等模块，保证检索的精准度；
3. **在线问答层：**为了满足产品设计和交互体验需要，这里支持多文档、多轮次、多模态等；
4. **应用层：**针对不同行业的特点，预制多种场景类角色，降低产品使用门槛。

4. 技术架构



RAG技术框架分为三个主要组成部分：问题理解模块、检索召回模块和答案生成模块。

1. **问题理解模块：**该模块旨在对用户的问题进行理解或者将用户的问题生成结构化的查询，既可以查询结构化的数据库也可以查询非结构化的数据，进而提高召回率。该模块主要包括意图识别、问题改写和问题扩写；
2. **检索召回模块：**该模块旨在从给定的知识库中检索与用户问题最相关的信息。该模块主要包括文档加载、文档转换、文档嵌入&检索和文档后处理；
3. **答案生成模块：**该模块旨在将检索结果与原始问题整合后的增强提示词输入给LLM，让LLM生成精准、可靠的答案。该模块主要包括Prompt拼接和LLM生成策略。

4.1. 问题理解模块

在RAG应用中，可能会遇到从知识库中检索到与用户问题不相关的内容，这是由于：

- 用户问题过于简单，没有更多的上下文信息；

- 用户问题可能混杂无关内容；
- 用户问题有模糊词汇、歧义词汇和缩写词；
- 用户问题没有生成结构化查询。

为了解决上述问题，在RAG应用中可能需要引入问题理解模块。

4.1.1. 意图识别

对用户问题进行意图识别，解析出用户问题的隐含需求，使RAG系统能自适应地组合检索策略，平衡效率与精度。

核心方法如下：

- **数据源选择优化**
  - 通过实体识别提取用户问题中的关键要素（如产品型号、时间范围），动态匹配对应领域的知识库；
  - 若问题包含视觉关键词（如“CT影像分析”），自动切换到医学图像库，文本类问题则调用文档数据库。
- **查询策略决策**
  - 当用户需求为宏观概览（如“概括新能源政策要点”），触发摘要引擎对长文档进行压缩提取；
  - 当用户需要详细的技术细节（如“如何修复手机充电接口接触不良”），启用向量索引进行深度语义匹配。
- **多路由协同执行**
  - 对开放式问题（如“人工智能发展趋势分析”），同时调用行业报告库、学术论文库和新闻数据库，合并检索结果；
  - 在金融风控场景中，既检索合同条款库又调用法律条文库，综合生成合规建议。

4.1.2. 问题改写

对用户问题进行语义等价的重构与优化，保持核心意图不变，提升问题表达清晰度与检索精准性。

核心方法如下：

- **同义词替换：**将口语化词汇转换为专业术语（如“苹果手机”→“iPhone 15 Pro Max”）；
- **消除歧义：**通过上下文补全明确模糊表述（如“苹果”补充为“苹果公司产品”）；
- **结构优化：**拆分复合问题（如“Python安装和数据分析”拆分为Python安装和Python数据分析两个子查询）。

4.1.3. 问题扩写

在原问题基础上增加关联信息或限定条件以扩展覆盖范围，提高检索召回率与长尾信息覆盖能力。

核心方法如下：

- **关键词拓展：**添加相关术语（如“学习Python”扩展为“Python编程教程+实战项目资源”）；
- **条件补充：**增加时间/空间约束（如“新能源车续航”补充为“2024年特斯拉Model Y冬季续航数据”）；
- **领域知识注入：**融入专业参数（如“拍照好”扩展为“索尼IMX989传感器+光学防抖”）。

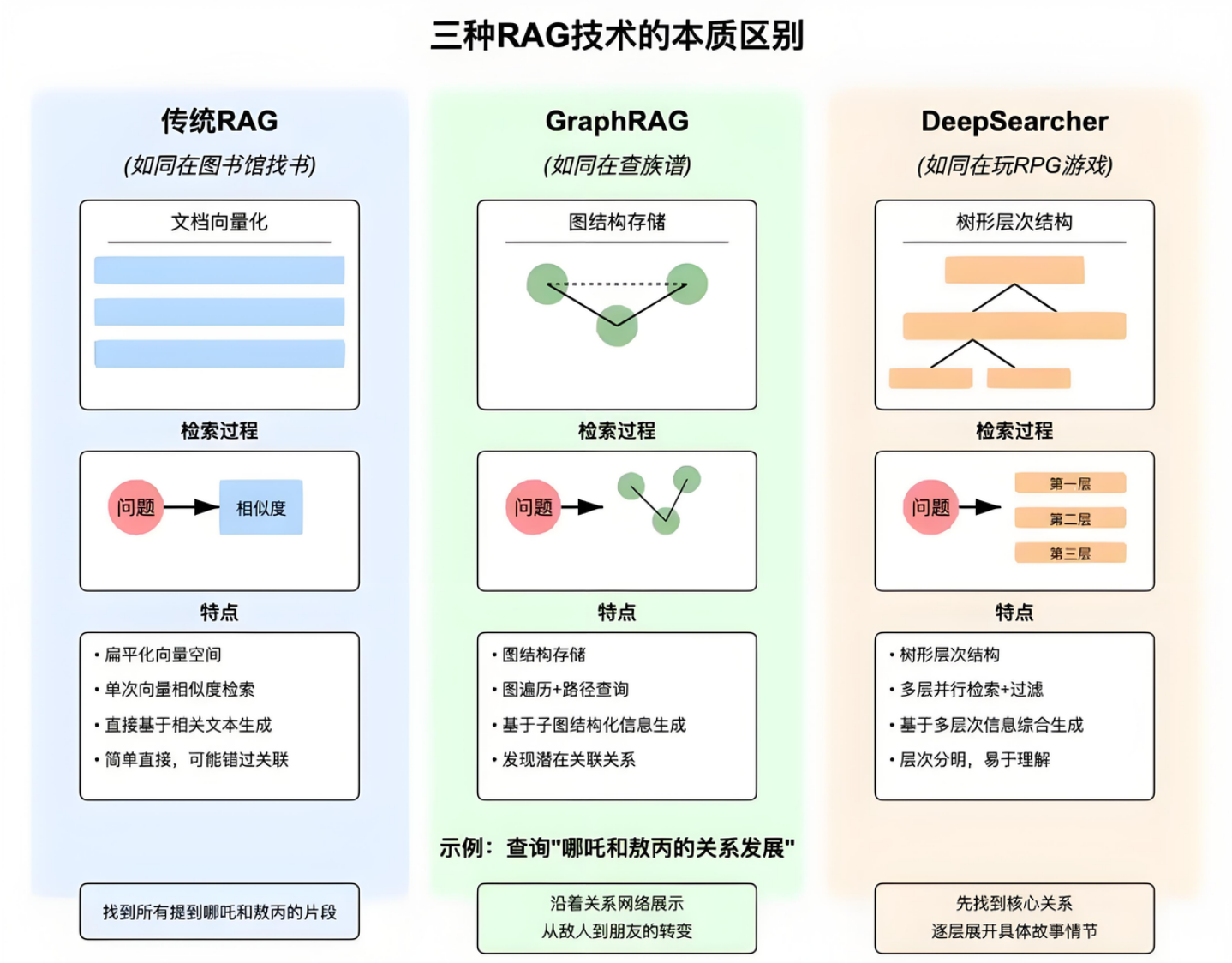
4.2. 检索召回模块

> 本文只针对文本数据进行讲解，多模态数据（图像、音频和视频）的处理比较复杂，这里先不展开讲。

4.2.1. 文档检索

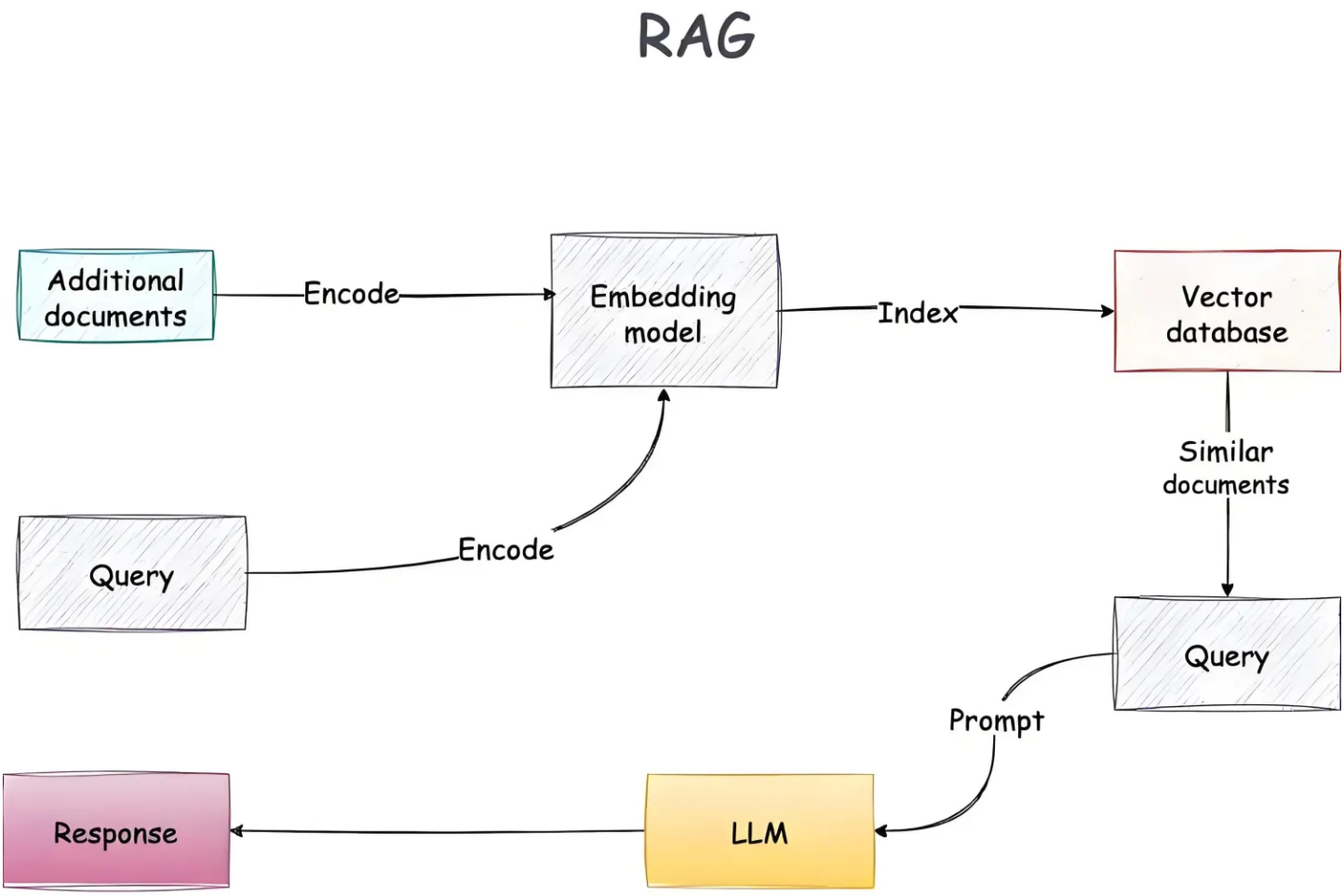
在实际应用中，一般需要获取到与用户问题最相关的那些文档数据，而不是将整个文档全部检索出来。因此检索技术的选择和实现就至关重要了，这个直接影响到RAG系统的应用效果。

不同的应用场景对应不同的检索技术，主要有传统RAG、GraphRAG以及DeepSearcher，传统RAG和GraphRAG是成熟的范式，分别适合文档检索和关系查询，而DeepSearcher代表未来方向，通过深度学习优化搜索过程，适合复杂、个性化的需求。



传统RAG

传统RAG适用于单点的事实查询，采用的是扁平化的向量空间形式，让信息之间的关系更加直观，然后通过比较文本向量之间的相似度来找到最相关的信息。



这里的技术要求是如何将一个大型的文档数据进行分割，以便可以更精确的检索到与用户问题相关的文档片段，然后放到LLM的上下文中。常见的分割方法有：

- **递归分割 (Recursive)：**基于用户定义的字符列表递归分割文本，旨在保持相关文档片段相邻。推荐作为初始分割方法；
- **HTML 分割：**基于HTML特定字符分割文本，并添加关于文本来源的元数据（基于HTML结构）；
- **Markdown 分割：**基于Markdown特定字符分割文本，并添加关于文本来源的元数据（基于Markdown结构）；
- **代码分割：**基于编程语言（如Python、JS等）特定字符分割文本，支持15种不同语言；
- **词元分割 (Token)：**基于token分割文本，存在几种不同的token测量方式；
- **字符分割 (Character)：**基于用户定义的单个字符分割文本，是一种较为简单的方法；
- **语义块分割 (Semantic Chunker)：**首先按句子分割文本，然后如果相邻句子在语义上足够相似，则将它们合并。

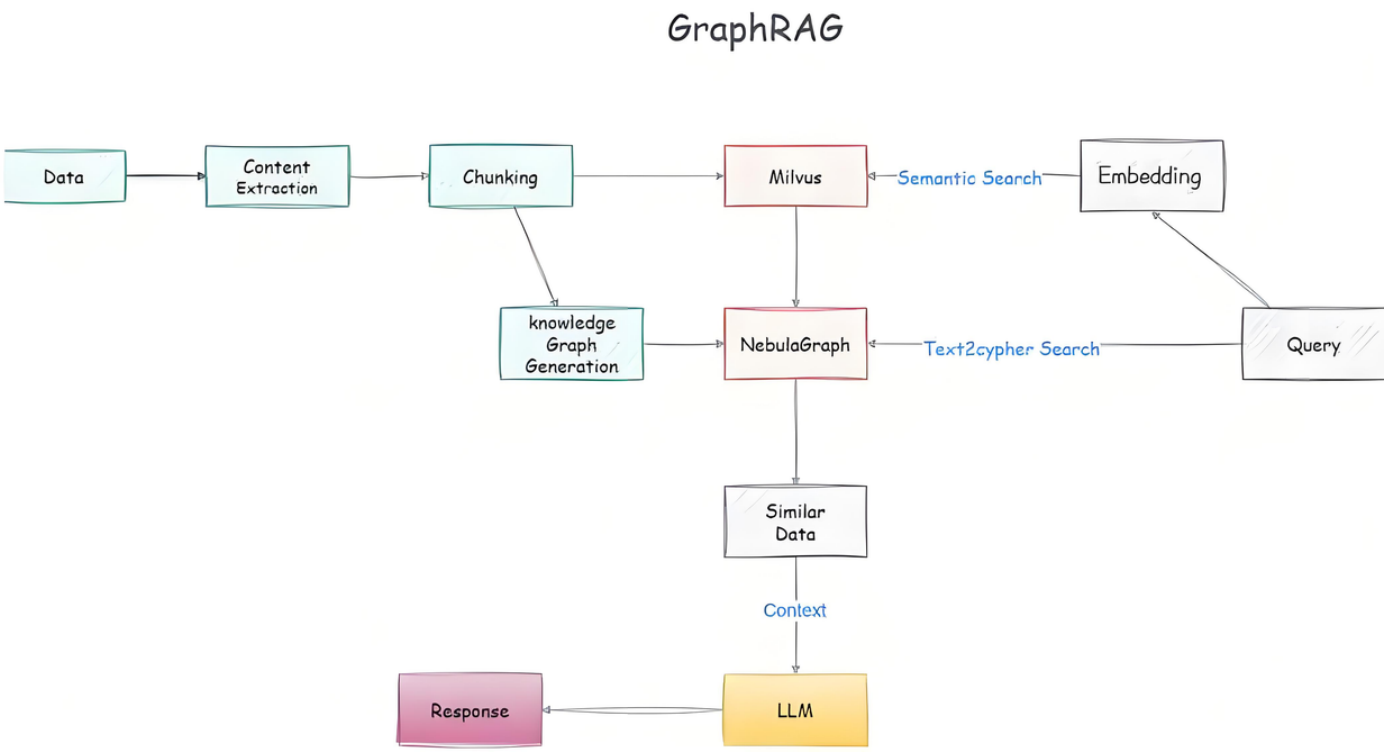
文档数据分割好了一个个小的片段之后，在使用文本嵌入模型（如 text-embedding-ada-002）将文本块转换成固定维度的向量，然后存储到向量数据库（如 Milvus）中。

```
哪吒是一个天生反骨的少年英雄,具有雷电属性,属于阐教。
他的父亲是李靖(陈塘关总兵),母亲是殷夫人。
他的师父是太乙真人,是阐教弟子。
敖丙是东海三太子,具有冰雪属性,属于龙族。
...
```

GraphRAG

GraphRAG适用于需跨实体关联的复杂问题，引入知识图谱技术构建实体关系网络，通过图数据库（如 NebulaGraph）实现多跳推理与结构化知识关联。

该技术结合了图结构的数据遍历和路径查询技术，能够更精准地定位相关信息，然后结合向量相似度检索补充局部语义信息，形成混合结果集给到LLM。



这里的技术要求是要在传统RAG的基础之上构建知识图谱：

1. **实体与关系抽取：**调用LLM识别文本块中的实体（如人名、机构名）和关系（如因果关系、隶属关系），输出为（实体、关系、实体）三元组，然后对提取的实体生成抽象描述性摘要，解决实体引用不一致问题（如别名归一化）；
2. **图结构生成：**构建同质无向加权图（节点：表示实体，权重反映实体出现频率与重要性，边：表示关系，权重通过关系实例的标准化计数计算），然后通过图数据库（如 NebulaGraph）存储图谱数据，支持分布式扩展。

```
INSERT VERTEX role (name, meteorological, faction, role_desc, voice_actor) VALUES
  "哪吒": ("哪吒", "雷电", "阐教", "天生反骨的少年英雄", "吕艳婷"),
```

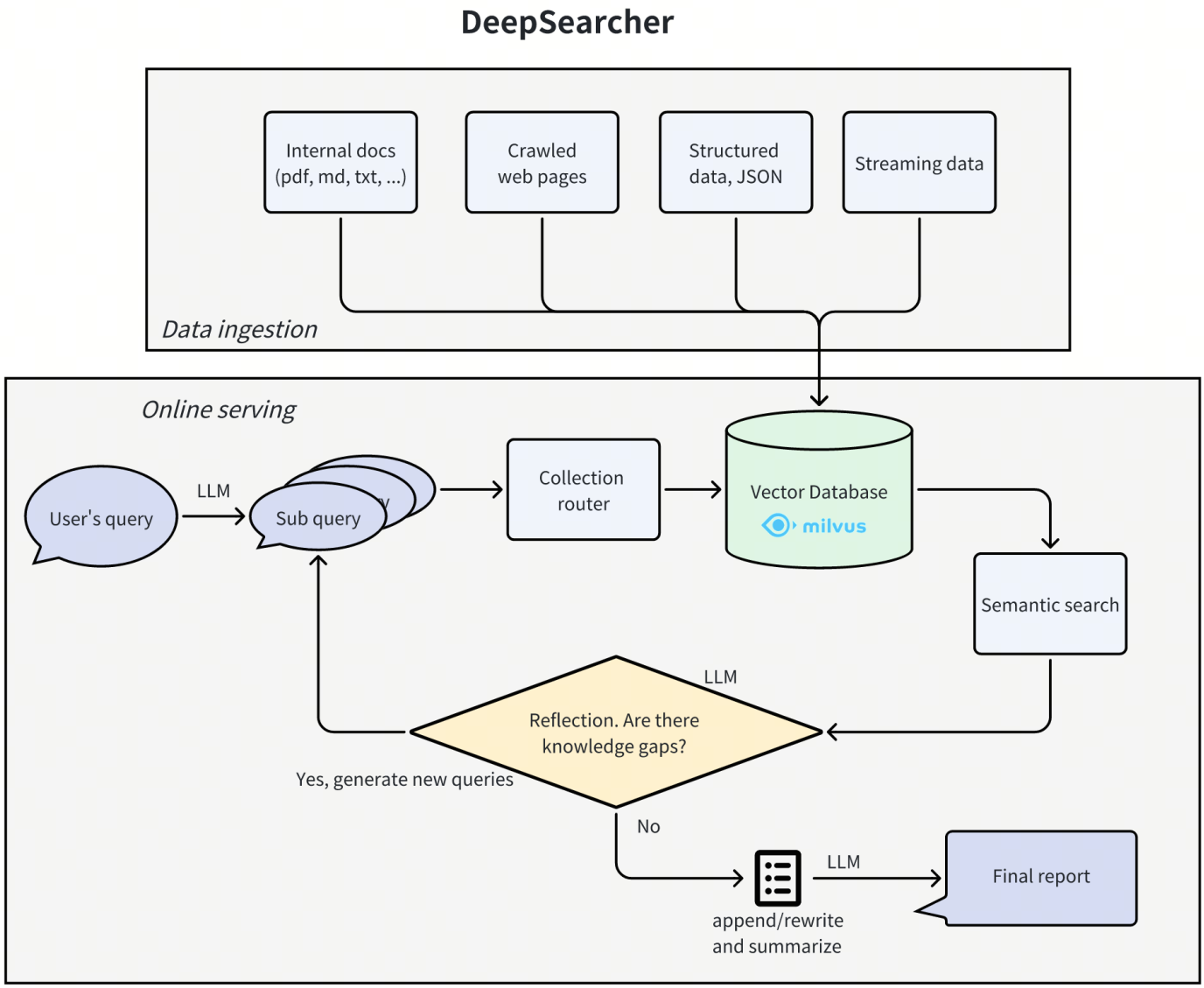
```
    "敖丙": ("敖丙", "冰雪", "龙族", "东海三太子,哪吒的挚友", "瀚墨"),
    "太乙真人": ("太乙真人", "云雾", "阐教", "哪吒的师父,阐教弟子", "张珈铭"),
    "殷夫人": ("殷夫人", "细雨", "人类", "哪吒的母亲,温柔贤惠", "绿绮"),
    "李靖": ("李靖", "木", "人类", "哪吒的父亲,陈塘关总兵", "陈浩"),
    "申公豹": ("申公豹", "风暴", "截教", "截教弟子,野心勃勃", "杨卫"),
    "敖光": ("敖光", "海浪", "龙族", "东海龙王,敖丙之父", "雨辰"),
    "敖闰": ("敖闰", "空间系", "龙族", "东海龙王之女", "周泳汐"),
    "敖顺": ("敖顺", "毒", "龙族", "东海龙王之子", "韩雨泽"),
    "敖钦": ("敖钦", "火", "龙族", "东海龙王之子", "南屿"),
    "石矶娘娘": ("石矶娘娘", "沙尘", "截教", "截教弟子", "小薇"),
    "结界兽": ("结界兽", "雾霭", "中立", "守护结界的神兽", "");
```

```
// 家庭关系
INSERT EDGE father_of VALUES "李靖" -> "哪吒": (NOW());
INSERT EDGE mother_of VALUES "殷夫人" -> "哪吒": (NOW());
INSERT EDGE father_of VALUES "敖光" -> "敖丙": (NOW());
INSERT EDGE father_of VALUES "敖光" -> "敖闰": (NOW());
INSERT EDGE father_of VALUES "敖光" -> "敖顺": (NOW());
INSERT EDGE father_of VALUES "敖光" -> "敖钦": (NOW());
// 师徒关系
INSERT EDGE teacher_of VALUES "太乙真人" -> "哪吒": (NOW());
// 朋友关系
INSERT EDGE friend_of VALUES "哪吒" -> "敖丙": (NOW());
INSERT EDGE friend_of VALUES "敖丙" -> "哪吒": (NOW());
// 敌对关系
INSERT EDGE enemy_of VALUES "申公豹" -> "太乙真人": (NOW());
INSERT EDGE enemy_of VALUES "申公豹" -> "哪吒": (NOW());
// 兄妹关系
INSERT EDGE brother_sister_of VALUES "敖丙" -> "敖闰": (NOW());
INSERT EDGE brother_sister_of VALUES "敖丙" -> "敖顺": (NOW());
INSERT EDGE brother_sister_of VALUES "敖丙" -> "敖钦": (NOW());
```

DeepSearcher

DeepSearcher适用于需动态调整的多维度需求，通过树形层次结构来组织数据，并逐步反馈迭代，使得查找特定信息时能够像浏览文件夹一样方便快捷。

该技术采用多层次并行搜索加上智能过滤的方式从大量数据中快速准确地获取所需信息，然后结合多模态检索（向量+关键词+图），最终得到最精准的信息。



角色基本信息

哪吒- 名称：哪吒- 属性：雷电- 阵营：阐教- 描述：天生反骨的少年英雄，拥有超凡的力量和勇气- 配音：吕艳婷- 性格特点：叛逆不羁，重情重义，敢于挑战命运

哪吒的关系网络- 父亲：李靖(陈塘关总兵，严厉正直)- 母亲：殷夫人(温柔慈爱，理解包容)- 师父：太乙真人(循循善诱，关爱弟子)- 挚友：敖丙(东海三太子，冰雪之力)- 敌人：申公豹(截教弟子，处处作梗)

哪吒的剧情发展- 初遇敖丙：在东海边缘的相遇，两个不同世界的少年- 修行历程：在太乙真人门下学习法术，逐渐掌握雷电之力- 友情萌芽：与敖丙从互不理解到成为挚友- 身份困扰：面对阐教弟子和凡人双重身份的矛盾- 成长蜕变：在各种挑战中突破自我，寻找真我

哪吒的能力特点- 主要法术：雷电操控，混天绫，乾坤圈- 战斗风格：灵活多变，攻击凌厉- 特殊天赋：天生具有超凡力量- 成长轨迹：从初学者到掌握强大法力

敖丙- 名称：敖丙- 属性：冰雪- 阵营：龙族- 描述：东海三太子，温润如玉的贵族少年- 配音：瀚墨- 性格特点：温和有礼，重情重义，内心坚韧

敖丙的关系网络- 父亲：敖光(东海龙王，威严庄重)- 兄弟姐妹： - 敖闰(龙女，擅长空间法术) - 敖顺(二皇子，精通毒术) - 敖钦(大皇子，掌控火焰)- 挚友：哪吒(阐教弟子，雷电之力)- 属下：结界兽(守护东海结界)

敖丙的剧情发展- 身份困扰：作为龙族继承人的责任与压力- 友情抉择：在族群立场与个人情谊间的挣扎- 能力觉醒：冰雪之力的不断提升与掌控- 性格成长：从谨慎拘谨到开朗自信- 守护之道：保

```
护东海与亲友的决心
敖丙的能力特点- 主要法术：冰雪操控,水系法术- 战斗风格：优雅从容,防守反击- 特殊天赋：天生亲和水元素- 成长轨迹：从单纯的王子到独当一面
太乙真人- 名称：太乙真人- 属性：云雾- 阵营：阐教- 描述：阐教重要弟子,哪吒的师父- 配音：张珈铭- 性格特点：智慧通达,慈悲为怀
太乙真人的关系网络- 弟子：哪吒(得意门生)- 同门：其他阐教仙人- 对手：申公豹(截教弟子)
太乙真人的剧情参与- 收徒教导：发现哪吒天赋,悉心培养- 化解危机：多次调解哪吒与各方矛盾- 守护正道：对抗截教势力的渗透
阵营势力分析
阐教- 代表人物：太乙真人、哪吒- 特点：崇尚正统,重视秩序- 立场：维护天地秩序,抵制混乱- 修行特色：注重心性修养,讲究循序渐进
阐教的理念- 修行观：重视内在修养- 处世态度：主动干预,匡扶正义- 对待人间：既重视规则,也关注个体
龙族- 代表人物：敖光、敖丙- 特点：高贵优雅,重视传统- 立场：守护东海,维护龙族利益- 统治方式：等级分明,讲究礼制
龙族的传统- 治理理念：重视血脉传承- 对外态度：谨慎自守,避免冲突- 内部规则：等级森严,重视礼法
截教- 代表人物：申公豹- 特点：包容驳杂,手段灵活- 立场：追求变革,不拘一格- 行事风格：灵活多变,善用权谋
截教的特点- 修行方式：讲究实用- 处世态度：积极进取,不拘形式- 发展策略：广收门徒,扩张势力
重要事件与剧情发展
东海危机### 事件起因- 结界异常- 势力冲突- 个人恩怨
事件发展- 哪吒与敖丙的相遇- 各方势力的介入- 矛盾的激化与升级
事件影响- 个人成长- 势力变化- 关系转变
人物关系演变### 友情的考验- 立场差异- 信任建立- 共同成长
师徒情谊- 教导方式- 互相理解- 成长蜕变
```

4.2.2. 检索后处理

经过前面的检索过程可能会得到很多相关文档，就需要进行筛选和排序。常用的筛选和排序策略包括：

- 基于相似度分数进行过滤和排序；
- 基于关键词进行过滤，比如限定包含或者不包含某些关键词；
- 让 LLM 基于返回的相关文档及其相关性得分来重新排序；
- 基于时间进行过滤和排序，比如只筛选最新的相关文档；
- 基于时间对相似度进行加权，然后进行排序和筛选。

4.3. 答案生成模块

检索召回模块基于用户问题检索出相关的文档片段，答案生成模块则是让LLM利用检索出的相关信息来生成对用户问题的回复。该模块依赖于LLM的能力以及prompt的拼接和调试。

4.3.1. Prompt拼接

用于将提示的不同部分组合在一起。您可以使用字符串提示或聊天提示来执行此操作。

- **字符串提示：**使用模板将所有查询到的文档片段和原始问题拼接在一起作为提示（如 langchain 的 promptTemplate）；
- **聊天上下文提示：**将历史对话记录、用户偏好等上下文信息、所有查询到的文档片段和原始问题拼接在一起作为提示（如 langchain 的 ChatPromptTemplate）。

4.3.2. LLM生成策略

- 策略1：是依次结合每个检索出的相关文档片段，每次不断修正生成的回复。这样的话，有多少个独立的相关文档片段，就会产生多少次的LLM调用；
- 策略2: 是在每次LLM调用时，尽可能多地在Prompt中填充文档片段。如果一个Prompt中填充不下，则采用类似的操作构建多个Prompt，多个Prompt的调用可以采用和前一种相同的回复修正策略。