DaoCloud

AI 进阶指南：基于 RAG 的 AI 应用落地

主讲人：陈佳

AI 进阶指南（上） 从理论到实践

DaoCloud

课程安排 AGENDA

07.23	深入浅出：大语言模型	张凡石	「DaoCloud 道客」高级研发工程师
07.30	基于 RAG 的 AI 应用落地	陈　佳	rDaocloud 道客」架构师
08.06	智能体的构建、迭代和高可用	尹伯昊	猴子无限 创始人
08.13	云原生技术优化模型推理	王　璠	「DaoCloud 道客」高级研发工程师
08.20	基于分布式和容器的方式微调大模型	黄敏杰	「DaoCloud 道客」高级研发工程师

Content

目录

Part 01

RAG 的基础概念

The Basic Concept of RAG

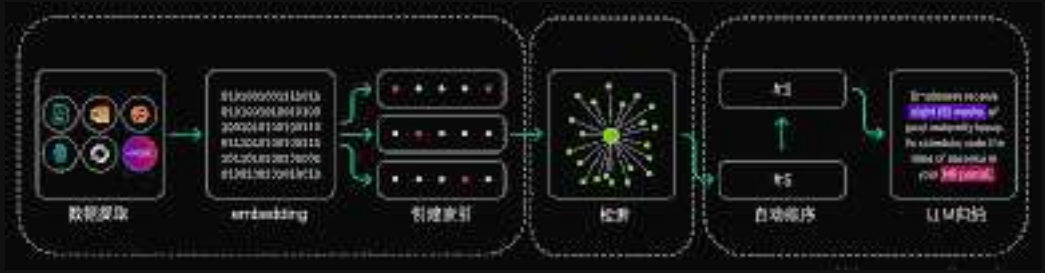
什么是 RAG（Retrieval-Augmented Generation）

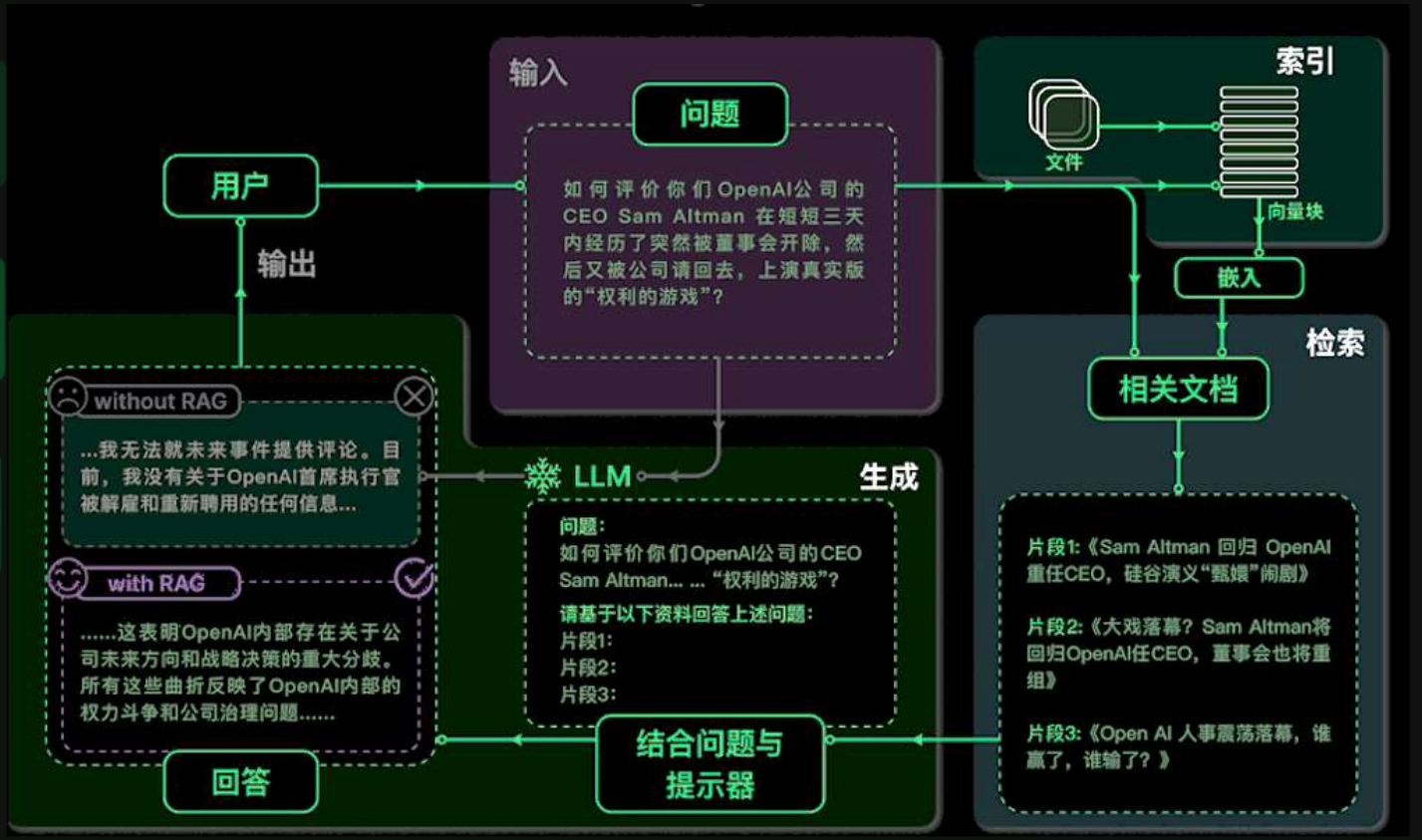
RAG 技术的核心理念是通过从大型数据库或知识库中检索相关信息，并将这些信息作为上下文输入给生成模型，从而生成更加准确和信息丰富的回答。

1 拆分. 索引：文档被拆分为块，编码为向量，并存储在向量数据库中。

2. 检索： 根据语义相似性检索与问题最相关的前 k 个块。

3. 生成： 将原始问题和检索到的块一起输入以 LLM 生成最终答案。





引用的部分论文资料：https://arxiv.org/html/2312.10997v5

为什么要使用 RAG 技术呢

增强准确性：RAG 结合了信息检索和生成模型的优势，通过从大型知识库中检索相关信息，提供更准确和相关的回答，提高生成内容的可靠性。

处理长尾问题：面对少见或独特的问题时，生成模型可能生成不准确的回答。RAG 通过检索相关信息，能更好地处理这些复杂的问题。

实时更新：生成模型的知识是固定的，而 RAG 可以通过实时检索最新的信息，提供更加及时和准确的回答，保持信息的最新性。

减少幻觉效应：生成模型有时会生成看似合理但实际错误的内容。通过结合检索机制，RAG 可以减少这种错误，提供更加可信的回答。

扩展知识范围：生成模型的知识受限于训练数据，而 RAG 能通过访问外部知识库，扩展知识范围，提供更全面的回答。

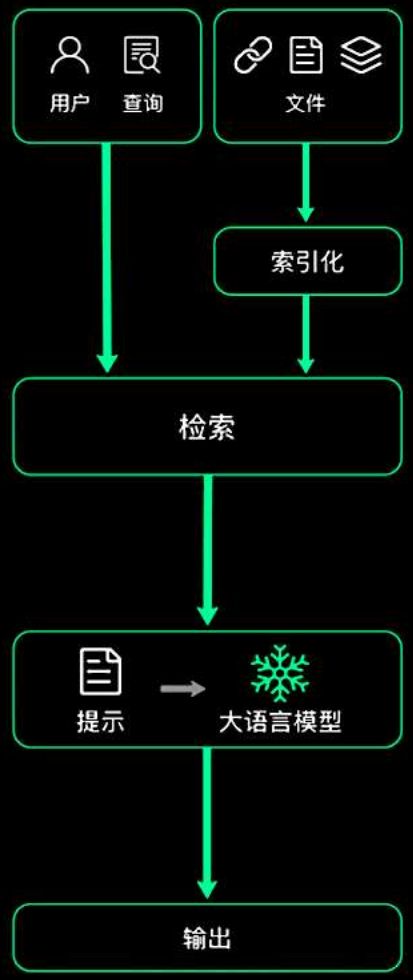
同时使用 RAG 技术可以降低训练和维护成本，提高效率和用户满意度，同时优化资源利用，具备灵活扩展性，整体上更具成本效益。

Part 02

RAG 的发展

The Development of RAG

Naive RAG 运行步骤



数据处理阶段

1.文本分块

数据清理与提取：处理 PDF、HTML、Word、Markdown 等不同格式的文件，提取纯文本格式。

文本分割：将整理后的纯文本文件分割成更小的片段，以适应语言模型的上下文处理能力。

2.向量化（Embedding）

使用 Embedding 模型，将每个文本分块向量化，形成对应的向量数据。

3.建立索引结构

将原始文本片段和向量数据对应，并创建合适的索引结构，以便后续相似度计算和快速检索。

使用阶段

1.数据检索

将用户的输入通过相同的 Embedding 模型转换为向量，根据问题向量与语料库中各文档块向量之间的相似性，选出相似度最高的前 K 个文档块，作为当前问题的补充背景信息。

2.结果生成

将搜索到的相似度最高的信息与提示词和问题一起交给大模型，处理成合适的总结回答。

Naive RAG 的局限性

检索质量

- **低精度**：检索到的文档块与查询内容不相关，导致信息错误或不连贯。
- **低召回率**：未能检索到所有相关的文档块，导致大语言模型缺乏足够的背景信息。
- **其他质量问题**：包括数据冗余和信息过时。

回应生成质量

- **虚构答案**：在缺乏足够上下文的情况下，模型可能会虚构答案。
- **错误信息**：生成错误信息和不相关的回答。
- **有害或偏见性回应**：生成有害或带有偏见的内容。

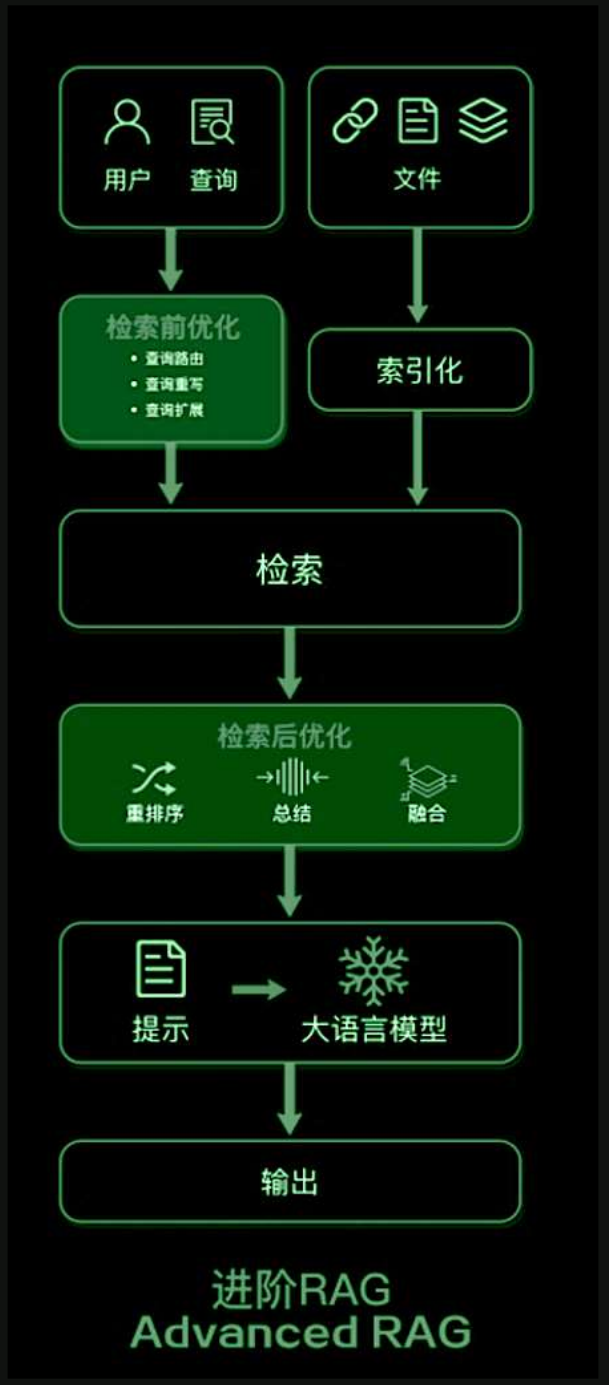
增强过程中的问题

- **上下文融入**：如何将检索到的文段上下文有效融入生成任务，避免生成内容的杂乱无章。
- **冗余和重复**：当多个检索包含相似信息时，导致生成内容的重复和冗余。

Advanced RAG

检索前（Pre-Retrieval）优化

- 增强数据粒度
- 优化索引结构
- 添加元数据信息
- 对齐优化
- 混合检索



检索后（Post-Retrieval）优化

重排序

对检索到的信息进行重新排序，将最相关的内容重新定位到提示的边缘是一个关键策略

上下文压缩

检索文档中的噪音会对 RAG 性能产生不利影响,需要压缩无关紧要的上下文，凸显关键段落，并缩短整体的上下文长度。

增强数据粒度

信息过滤（Filtering）

1.过滤掉低信息量及不能识别内容，确保语料质量。（例如 HTML 标记、无意义的词汇堆砌）。

2.分析新文档与既有文档的相似度，避免重复或相似语料的录入。

3.格式化处理，移除多余的空格和换行，保持语料的清晰和一致性。

文本解析（Parsing）

1. 对文件名、时间、章节、摘要等信息进行分析、管理，并保存对应元数据。

2. 对于图片、视频等非文字内容，进行编号和位置标记。

3、对于专用词汇进行替换，或者补充词汇字典。

切分方式（Chunking）

1.不同向量化模型会有最佳的切片大小。常规使用固定切分，需要做上下文冗余补充。

2.对于较短的文档，不进行切分，以防止信息的丢失。

3.针对不同的语料内容和使用场景，采用固定切分、语义切分、自定义切分等最适合的切分策略。

增益文本（Bbenefits）

1.根据文件主标题大纲、段落做正向填充。保证每个小标题都能总结段落的内容。

2.补充分片的元数据信息，通过元数据提升上下文的精确性和摘要能力

3.在处理短文本时，过度依赖信息增益可能导致人为的垃圾信息(spam)产生。

添加元数据信息-自动或手动添加

XX业务文档.pdf

文本类型分片1

{“source”:“XX业务文档.pdf","page":0}

文本类型分片2

{“source”:“XX业务文档.pdf","page":1}

文本类型分片3

{“source”:“XX业务文档.pdf","page":2}

文本类型分片N

{“source”:“XX业务文档.pdf","page":n-1}

文本分片metadata

QA提炼

【文本分片1】提炼的【QA类型分片】

Q|A 类型分片1

{“txtsource”:“文本类型分片1","index":0}

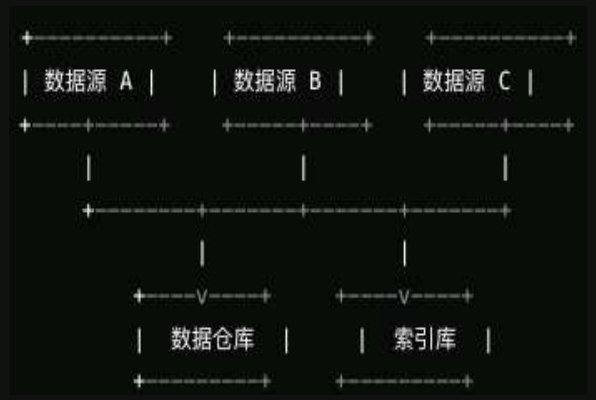
Q|A 类型分片2

{“txtsource”:“文本类型分片1","index":1}

QA分片metadata

使用大型语言模型（LLM）生成一系列文档可以回答的问题。这些问题是基于文档内容生成的，类似于可能由用户提出的问题。

对齐优化

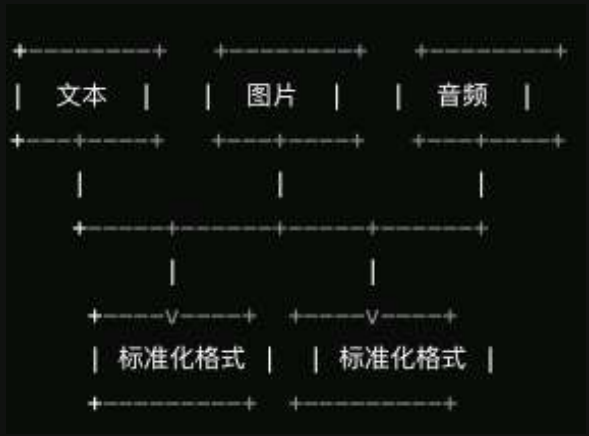


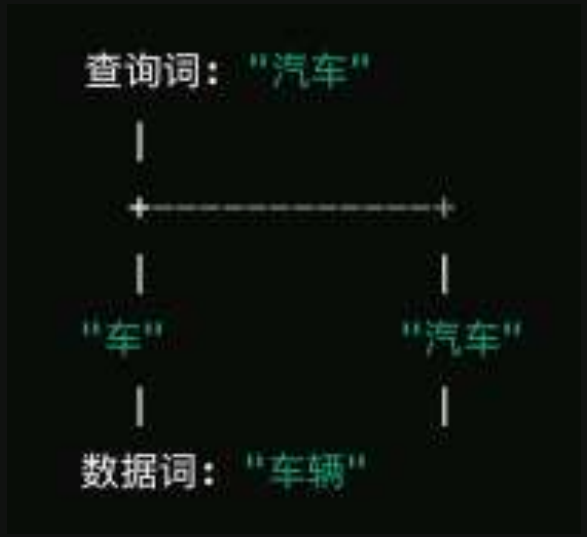
数据对齐

- **目标**: 确保不同数据源或不同数据格式之间的一致性和兼容性。
- **方法**: 融合多源数据，形成统一的索引，便于检索和处理。

格式对齐

- **目标**: 统一数据的格式和结构。
- **方法**: 将文本、图片、音频等不同格式的数据标准化，确保在索引时具有相同的结构。





语义对齐

- **目标**: 对齐查询和数据的语义，确保同一语义层次上的匹配。
- **方法**: 使用同义词库或语义网络，将查询词语与索引词语进行语义对齐，提高检索相关性。

索引对齐

- **目标**: 优化索引结构，反映数据的实际内容和查询需求。
- **方法**: 调整索引层级，增加索引项的细粒度。

优化前索引结构：
- 一级索引
  - 二级索引
    - 三级索引

优化后索引结构：
- 一级索引
  - 细粒度二级索引
    - 细粒度三级索引

通过提高数据和查询的匹配度，增强检索效果。优化策略不仅提升检索速度，还提高检索结果的准确性和相关性，使用户能够更快、更准确地找到所需的信息。

数据对齐-图文语料的处理

支持具有图文和表格类型文本做切分

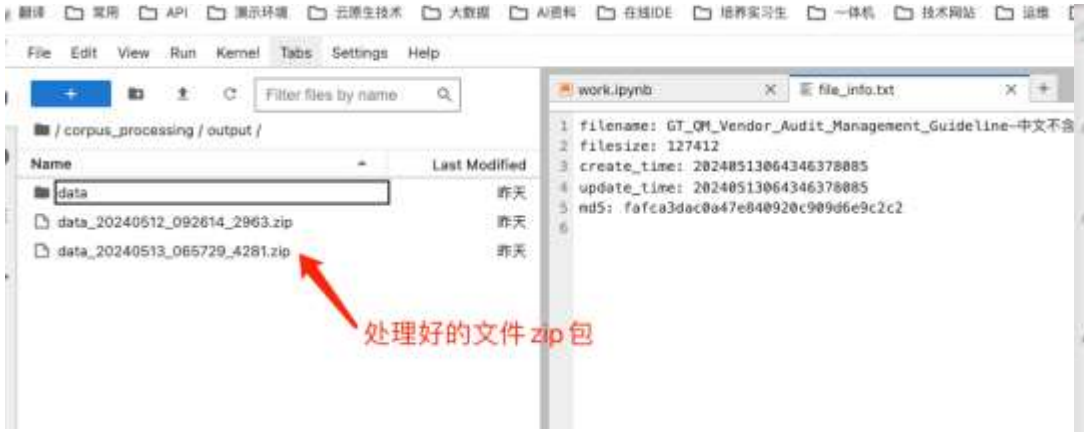
1.首先，对文档中已存在的文本信息进行整理，确保文本内容的完整性和准确性得到保留。

2.接着，识别文档中的所有图片，并为每张图片分配一个唯一的序号。我们将图片转换为特殊的标记形式，如 <pic>图片唯一编号</pic>，并将该标记放回文档中原图片的位置。系统将自动维护编号与图片的对应关系，同时也允许进行人工维护，以确保后续知识回答中图片能够原样呈现。

3.对于文档中的表格，我们将其识别并转换为数组格式的文本描述，以便于理解和处理。数组的第一行为表格标题，其余行则包含相应的数据。

使用OCR技术提取图片中的有意义文本，并将其整合到文档描述中，进一步丰富文档内容。通过这一优化流程，我们不仅提高了文档管理的效率，也保证了在后续的应用中能够准确地引用和展示图片信息。

处理后的zip导入文件信息



混合检索-按照语料做递进式检索

1.同义词库

2.行业术语库
（预处理策略）

QA 类型语料库
（一级检索策）

文本类型语料
（二级检索策）

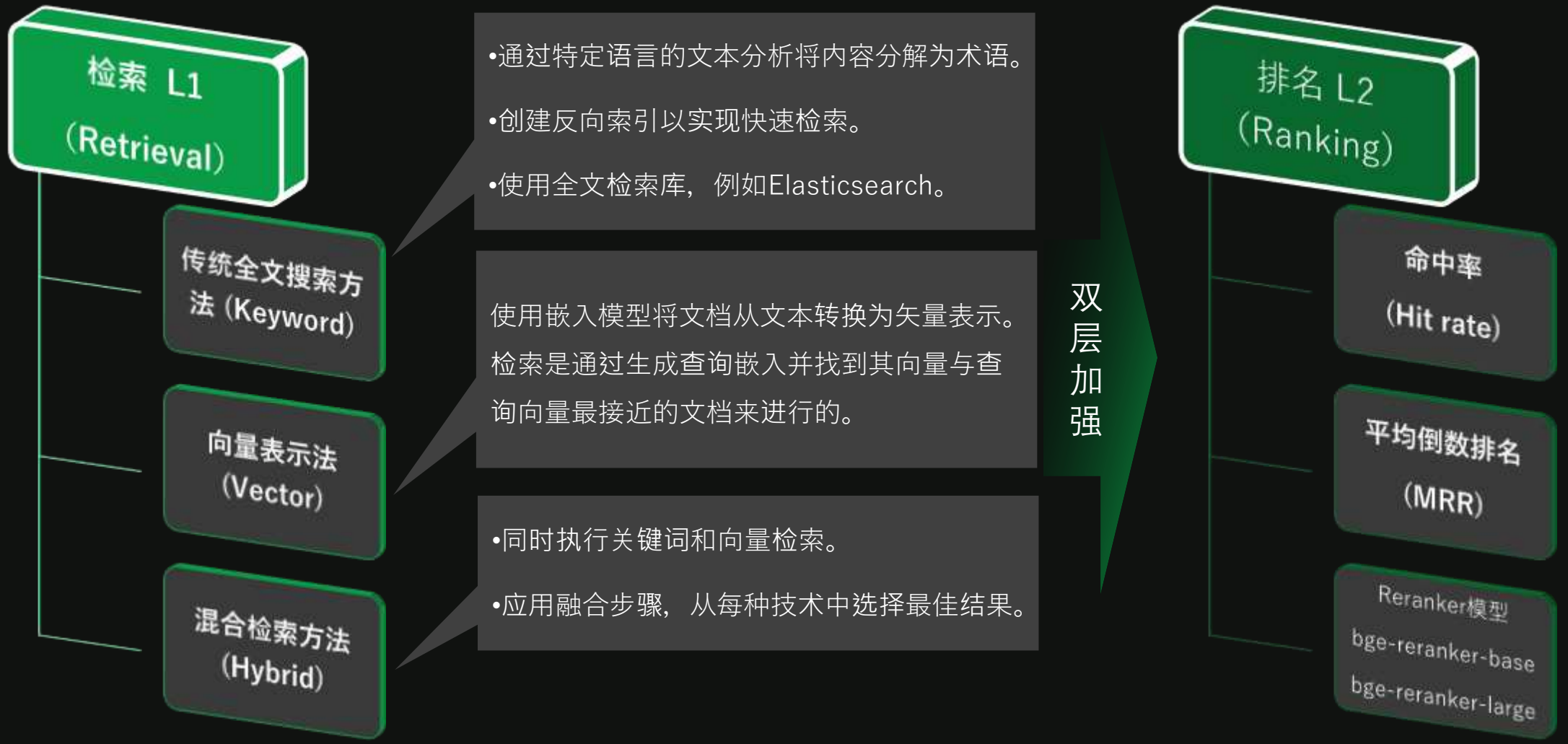
预处理策略：
为了提高问题预处理的效果，我们需要根据用户所处的行业领域，对问题中的专业术语和同义词进行精确的替换或提供相应的解释。
例如，当用户查询“CPU 的主频是多少？”我们应确保检索系统能够理解“CPU”指的是“中央处理器”，而“主频”是指其运行速度。如果检索系统错误地将“CPU”理解为某个品牌的缩写，或者将“主频”误解为音频设备的术语，那么提供的信息将完全不相关。

一级检索策略：
QA 类型语料的检索精度相对较高，我们可以仅对问题部分进行向量化检索或 L1 检索，并设置较为严格的匹配条件，比如相似度不超过 0.2，以确保检索到的语料具有高度匹配性。
但 QA 类语料还是会有总结不到的i问题，如果在 QA 检索中未能成功命中目标语料，那么可以启用二级检索策略，即 LV2 级别的检索，以作为补充和托底。

二级检索策略（用 Rerank 增强）：
普通文本类型的语料
在二级策略检索，采用混合向量+传统分词搜索后，使用 rerank 再排序打分。最在大程度上保证输送到，提示词词拼装的内容的正确性。

检索前（混合检索）+ 检索后优化（Rerank） 方式增强效果

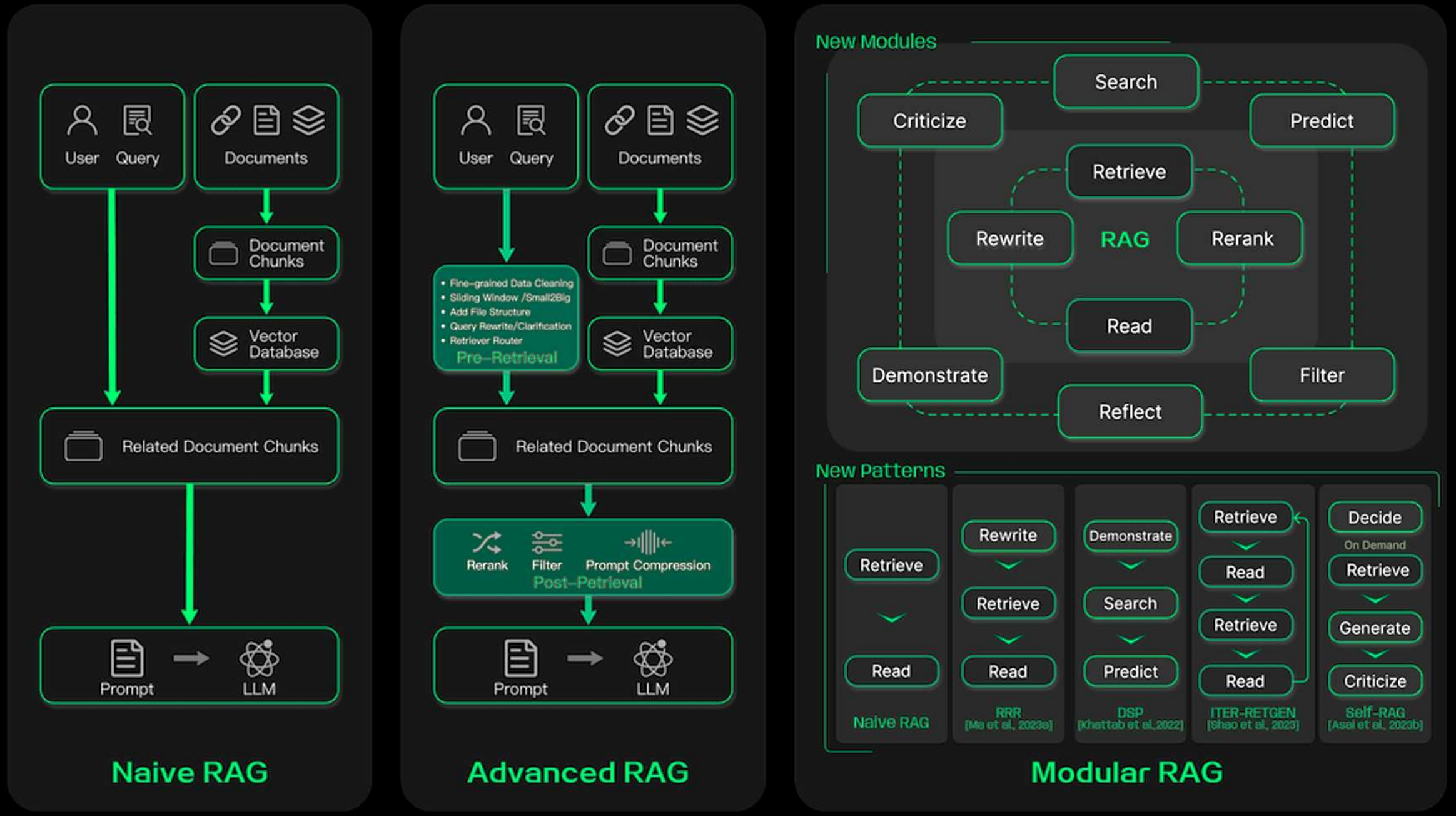
Retrieval Augmented Generation 增加检索文本分片准确性（解决模型幻觉的问题）。



Modular RAG

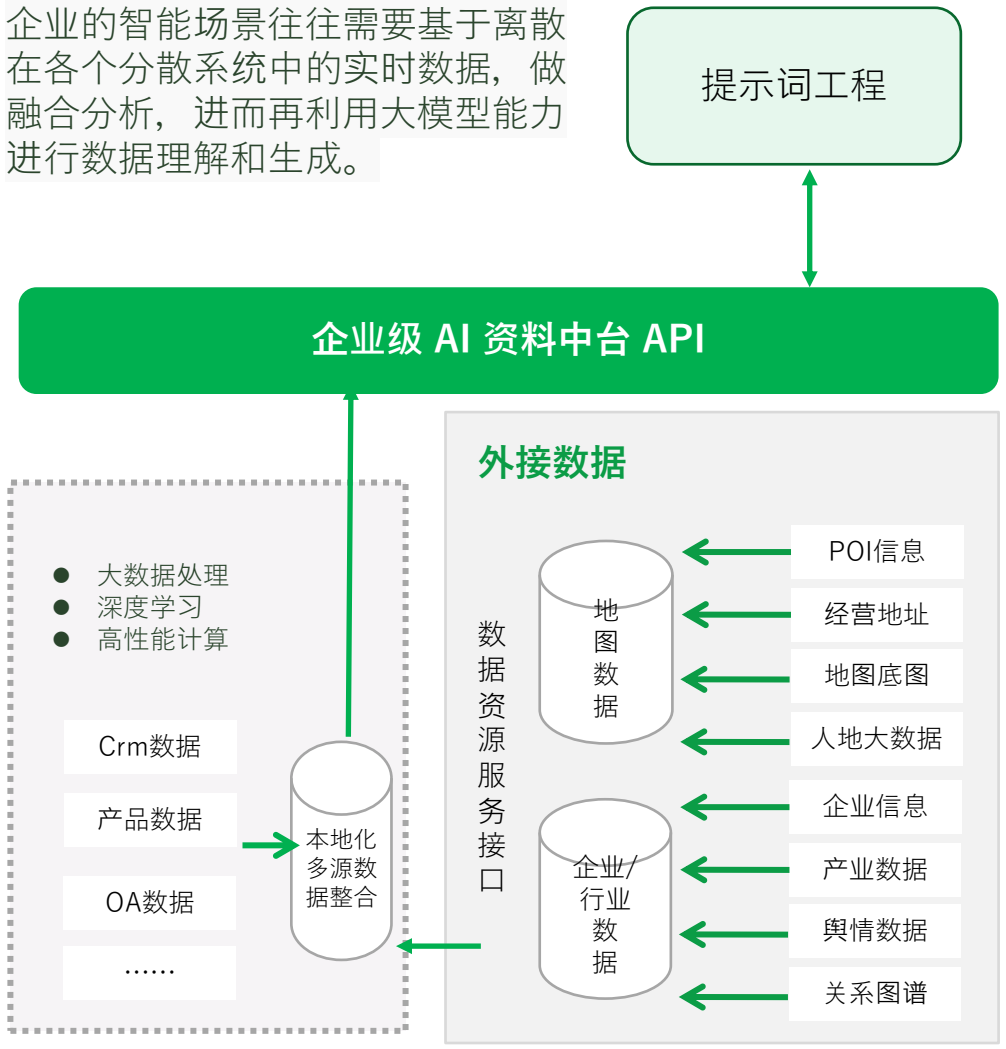
RAG框架通过引入专用组件增强了检索和处理能力：

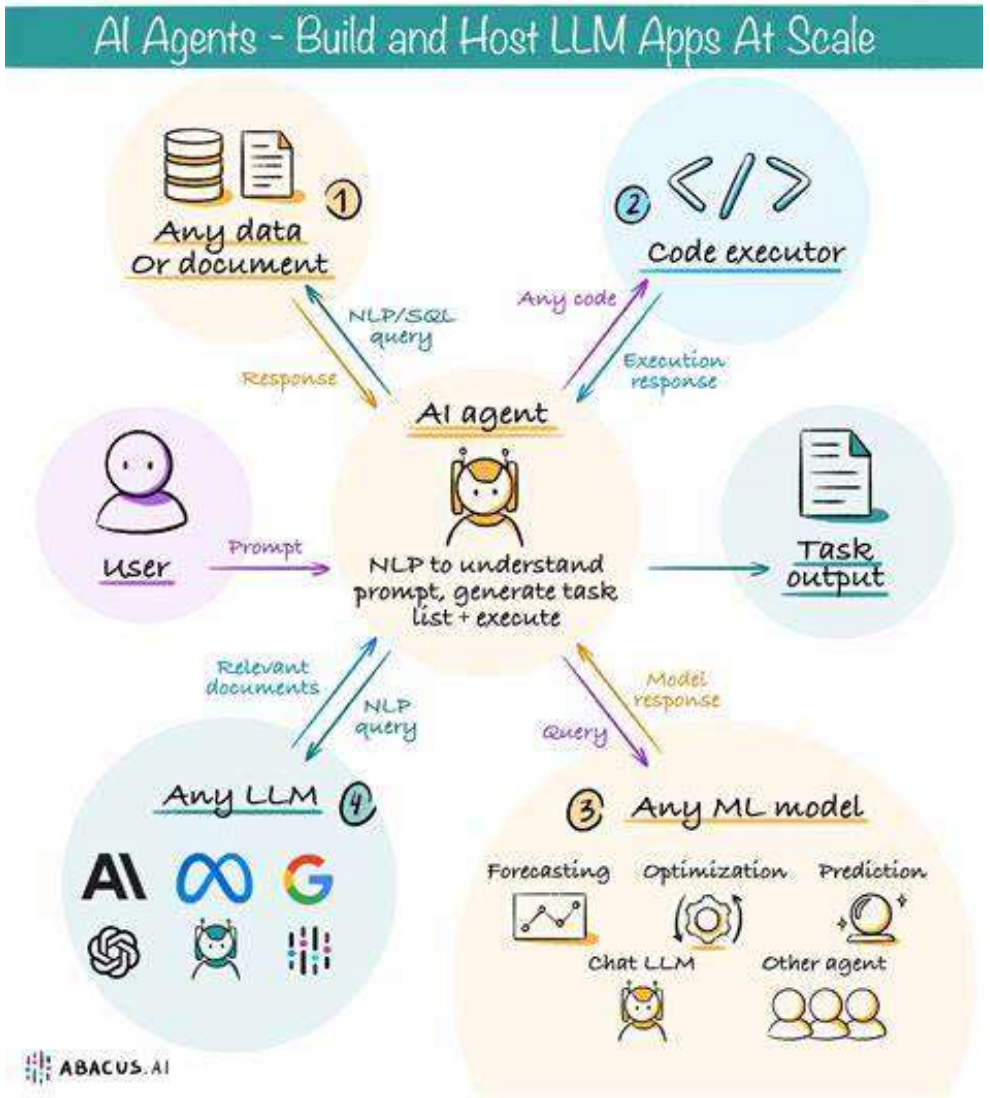
- **搜索模块Search**：适应特定场景，利用LLM生成的代码和查询语言，支持跨数据源（如搜索引擎、数据库、知识图谱）进行直接搜索。
- **融合模块Fusion**：采用多查询策略，扩展用户查询至不同视角，利用并行向量搜索和智能重新排序发现显性和变革性知识。
- **记忆模块Memory**：利用LLM的内存指导检索，创建无界内存池，通过迭代自我增强使文本与数据分布更紧密对齐。
- **路由模块Route**：在不同数据源间导航，选择最佳查询路径，支持汇总、数据库搜索及信息流合并。
- **预测模块Predict**：通过直接生成上下文减少冗余和噪声，确保相关性和准确性。
- **任务适配模块Task Adpter**：为各种下游任务定制RAG，自动提示检索零样本输入，并通过few-shot查询生成特定任务检索器这种方法简化了检索过程，大幅提高信息质量和相关性，以更高精度和灵活性满足各种任务和查询需求。


User
Query
Documents
Document Chunks
Vector Database
Related Document Chunks
Prompt
LLM
Naive RAG
• Fine-grained Data Cleaning
• Sliding Window /Small2Big
• Add File Structure
• Query Rewrite/Clarification
• Retriever Router
Pre-Retrieval
Rerank
Filter
Prompt Compression
Post-Petrieval
Advanced RAG
New Modules
Search
Criticize
Predict
Retrieve
Rewrite
RAG
Rerank
Read
Demonstrate
Filter
Reflect
New Patterns
Naive RAG
RRR
[Ma et al. 2023a]
DSP
[Khattab et al.2022]
ITER-RETGEN
[Shao et al. 2023]
On Demand
Self-RAG
[Asai et al. 2023b]
Modular RAG

RAG 的搜索不仅是语料数据-还有外部的 API 数据

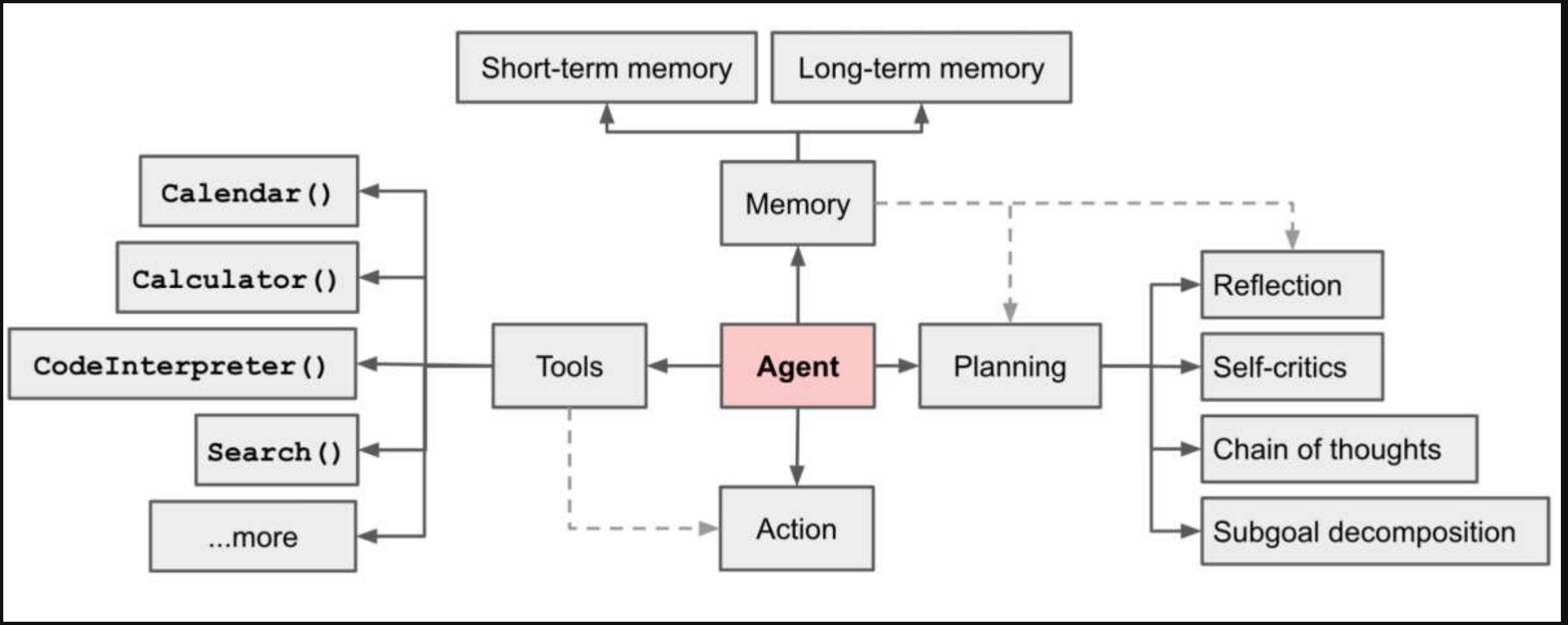
企业的智能场景往往需要基于离散在各个分散系统中的实时数据，做融合分析，进而再利用大模型能力进行数据理解和生成。



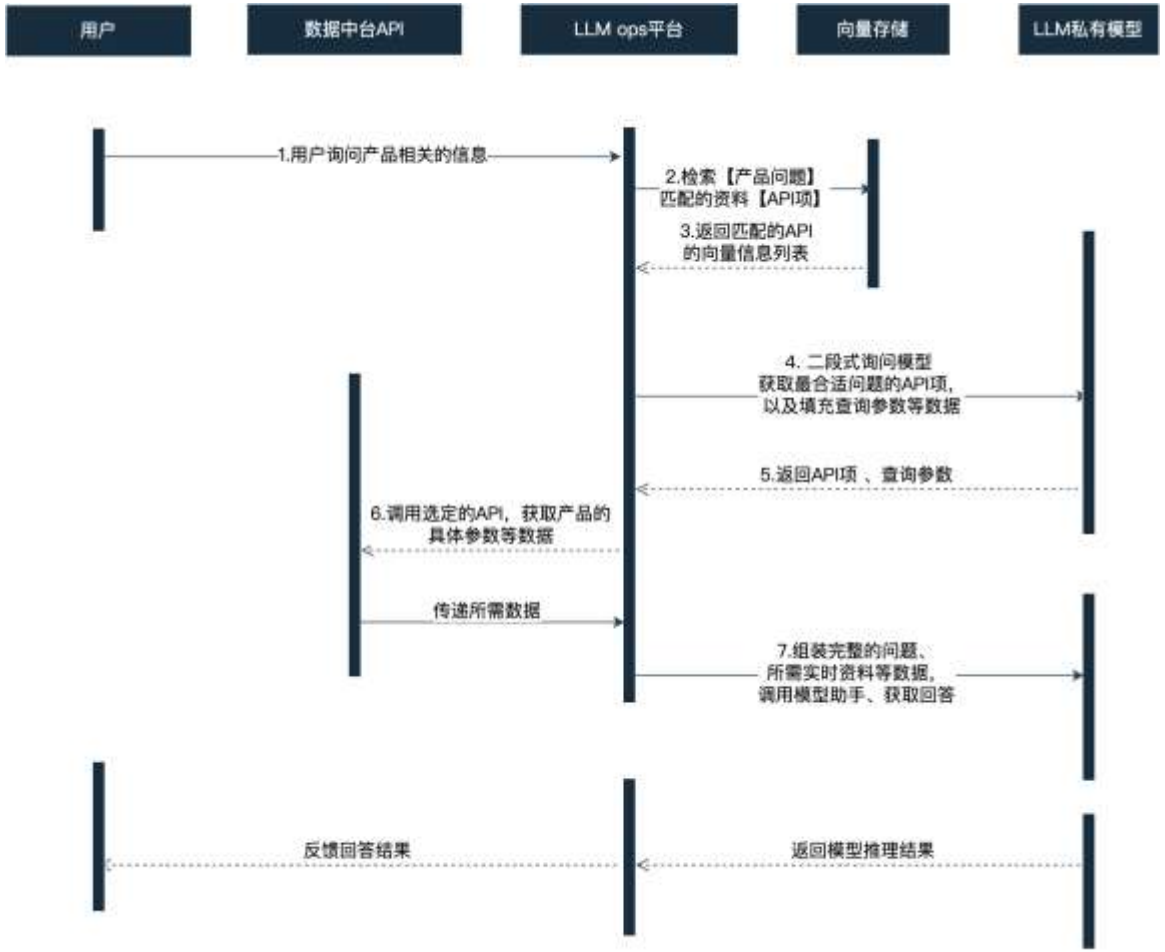


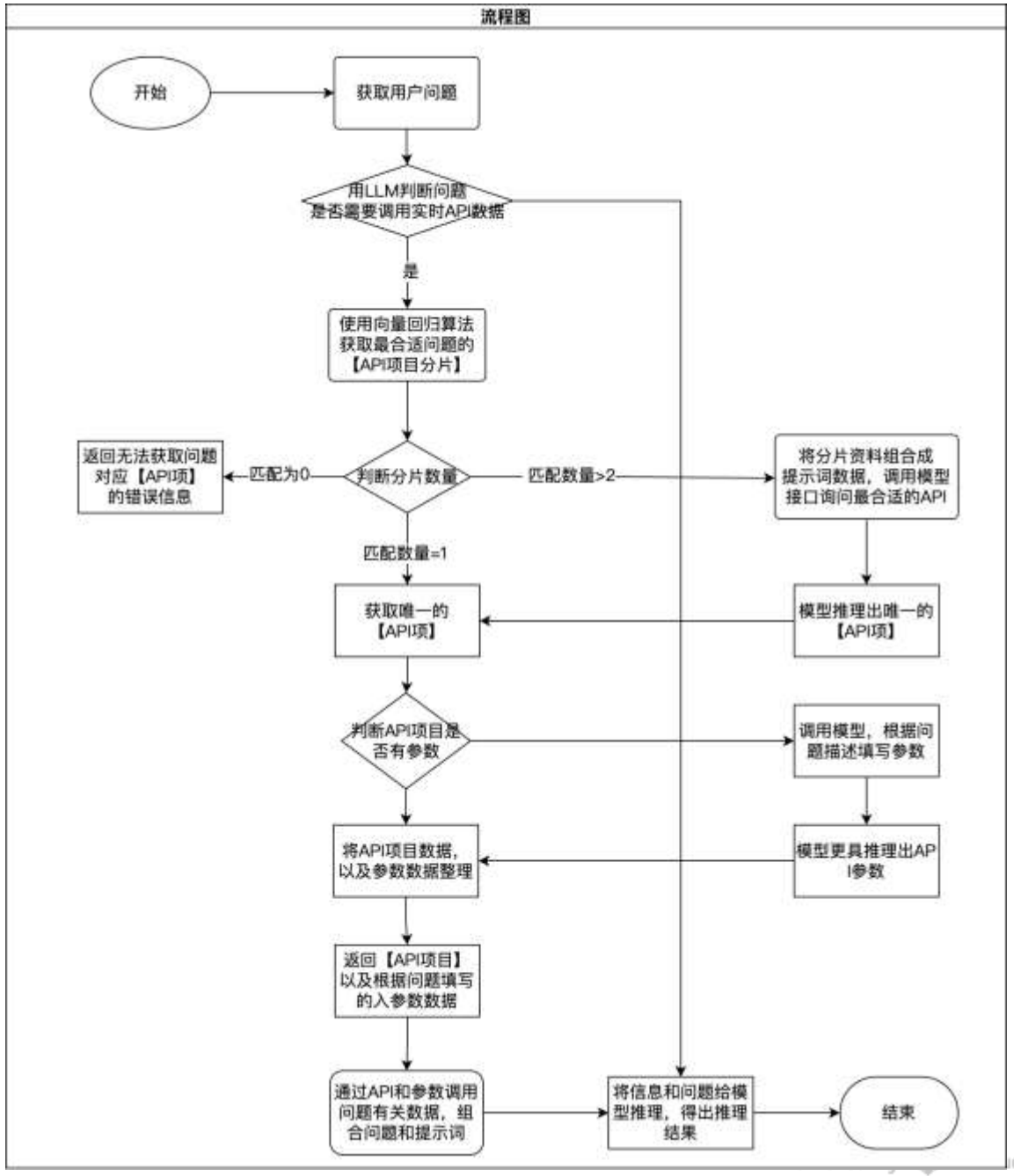
智能体（Agent）的实现架构图

由LLM充当代理（Agent）的大脑，并由规划、记忆、工具 等关键组件组成。



自构建 Agent 能力调用 API 工具





通过多段推理达成的示例

第一段推理（选定API）

一段推理（API 选择）：

提示词模版设计

变量设计：

序号	变量	变量说明	
1	{{api_list_data}}	api 列表信息	约定
2	{{question}}	问题信息	约定

提示词模版：

```
现在提供标准 json 数组数据 data，
data={{api_list_data}}。
根据 data 数组中 json 对象中的 itemName 和 itemDesc 字段 选出最符合描述 "{{question}}" 的 一个 json 对象，只选出一个最匹配的对象。最后输出该 json 对象，输出的 json 对象字段需要包括 itemCode、libraryId、itemName。不需要对选择过程做出解释，直接返回标准的 json 格式即可。
```

输入数据：

```
现在提供标准 json 数组数据 data，
data=[{"itemName":"按年度查询管理规模发展数据","itemDesc":"策略名称、销
根据 data 数组中 json 对象中的 itemName 和 itemDesc 字段 选出最符合描述 '
```

输出结果：

```
{
"itemCode": "API_0001",
"itemName": "按年度查询管理规模发展数据",
"itemDesc":"策略名称、销售渠道、产品数量、机构客户、代销机构、2019 年规模、2020 年规模、2021 年规模、目前规模、预计最大容量、该策略在运行的代表产品、机构资金占比",
"inputStruct": "开始年份是 startYear 的值,(格式为 YYYY,结束时间 endYear 的值(格式为 YYYY)。"
}
```

第二段推理（参数填充）

二段推理（填充参数，并返回）：

提示词模版设计

变量设计：

序号	变量	变量说明	
	{{nowDate}}	当前日期	约定
	{{itemCode}}	API 唯一代码	约定
	{{inputStruct}}	输入参数描述	约定
	{{question}}	问题信息	约定
	{{orgCode}}	组织机构代码	动态配置变量

提示词模版：

```
1.当前的日期为："{{nowDate}}"。
2.API 查询参数描述："{{inputStruct}}"。
3.如果参数内涉及日期区间查询，请尽量包含当前日期。
4.请根据以上信息，为问题"{{question}}"，填写合适的查询参数。以 API 查询参数描述的 json 的格式返回。
不需要对选择做出解释，直接返回 json 格式结果即可。
```

JSON

```
1.当前的日期为："2023-11-03"
2.API 查询参数描述："开始年份是 startYear 的值,(格式为 YYYY,结束时间 endYear
3.如果参数内涉及日期区间查询，请尽量包含当前日期。
4.请根据以上信息，为问题"查询近 5 年公司管理规模的数据"，填写合适的查询参数。以 A
不需要对选择做出解释，直接返回 json 格式结果即可。
```

输出结果：

```
{
"startYear": "2018",
"endYear": "2023"
}
```

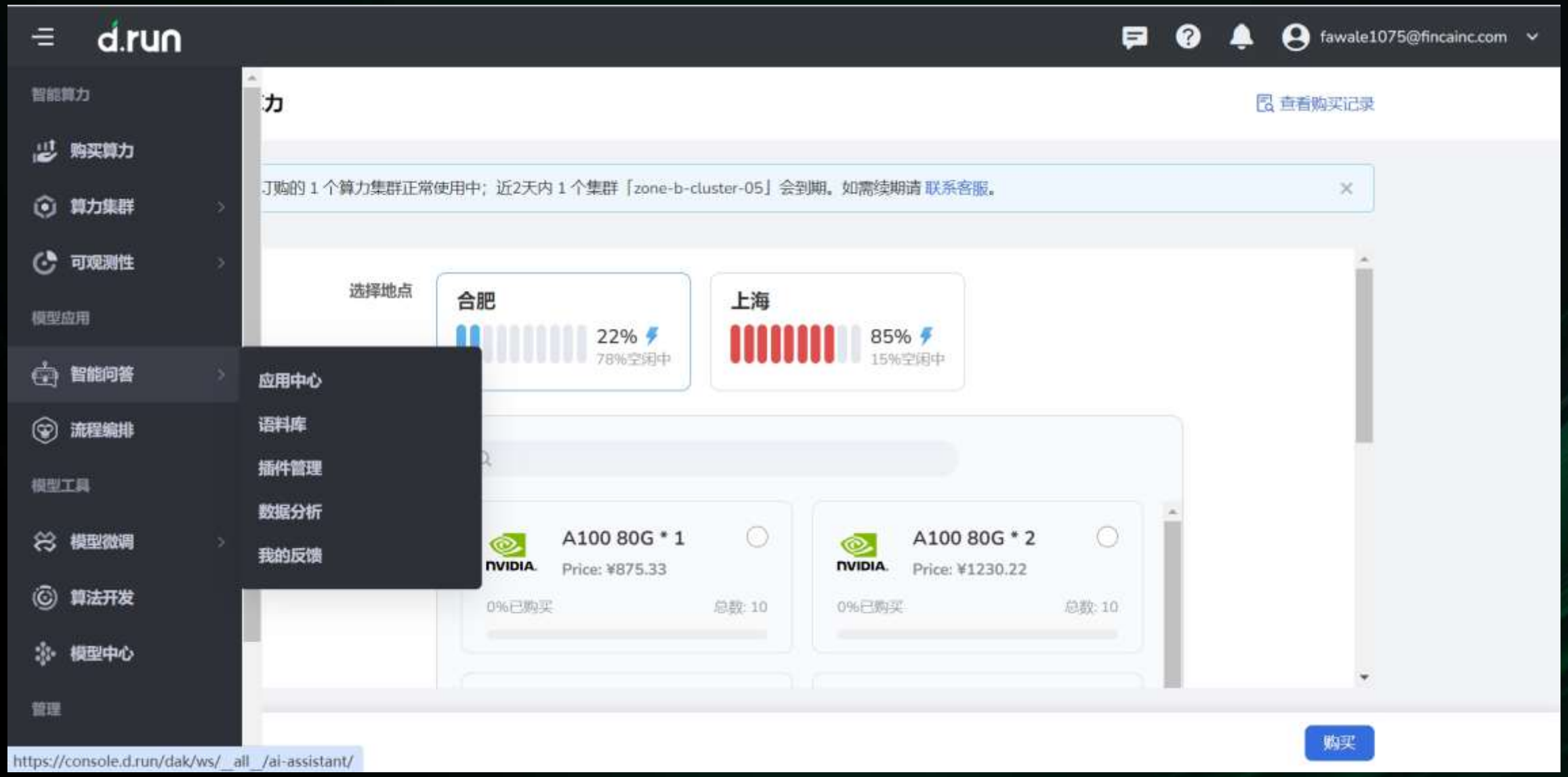
程序将参数组装填充

```
{
    "success": true,        //成功标志
    "fkid": "1719893501663842306",  //外部唯一号
    "message": "",          //返回处理消息
    "code": 200,            //返回代码
    "result":               //结果数据
        {
            //API 信息参数区域
            "itemCode": "API_0001",//API 唯一代码 CODE
            "itemName": "按年度查询管理规模发展数据",//API 名称
            "itemDesc": "策略名称、销售渠道、产品数量、机构客户、代销机构、2019 年规模、2020 年规模、2021 年规模、目前规模、预计最大容量、该策略在运行的代表产品、机构资金占比",
            //--透传的参数
            "myFkid": "1719893501663842306",//自定义唯一号（透传）
            "orgCode":"ABCD00001",          //组织机构代码（透传）
            //--模型推荐的参数
            "startYear": "2018",            //模型推理所得
            "endYear": "2023"               //模型推理所得
        },
    "timestamp": 1694677787846, //时间参数
}
```

Part 03

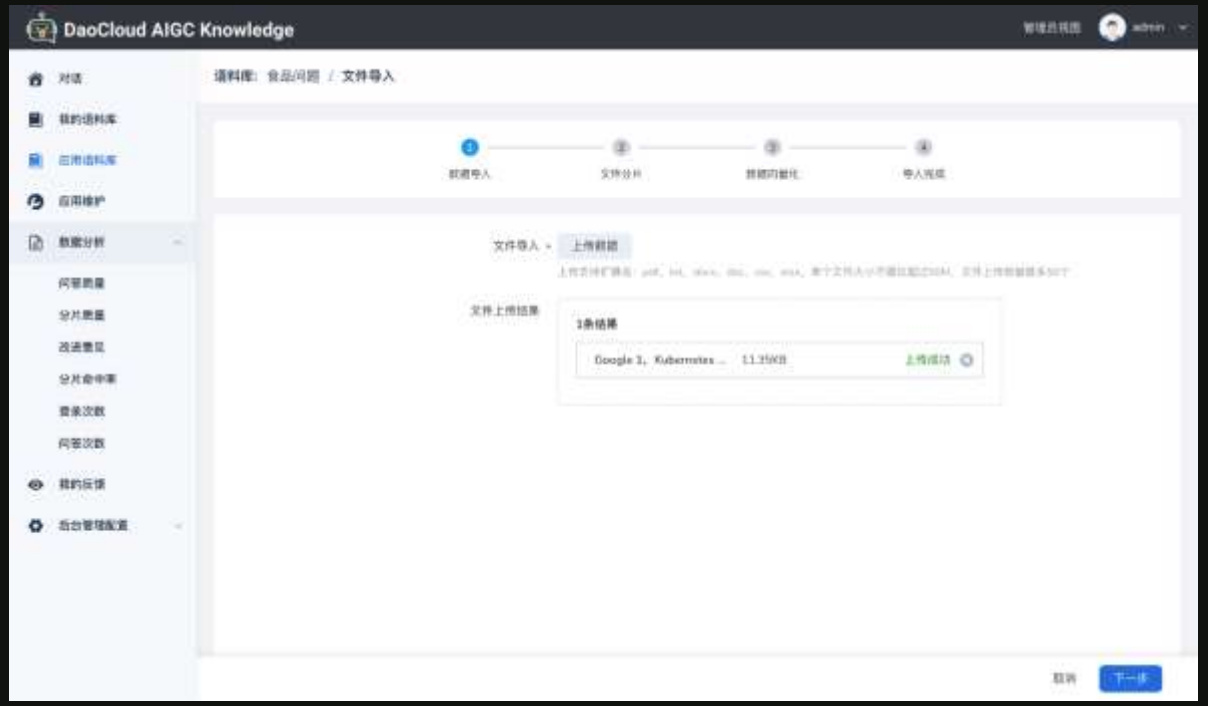
RAG案例举例

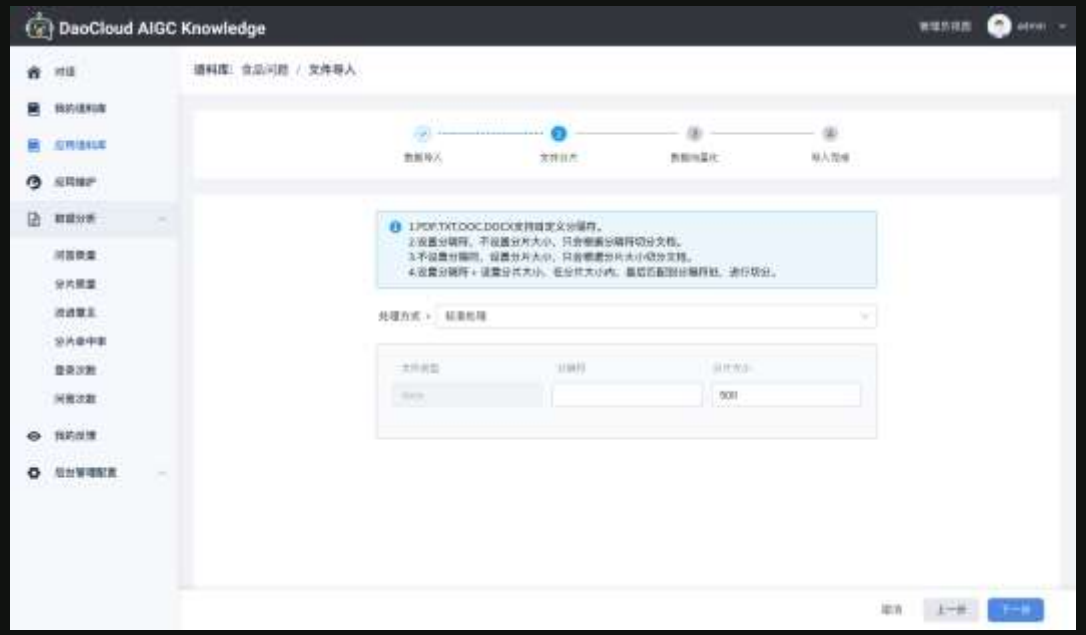
Examples of RAG Applications

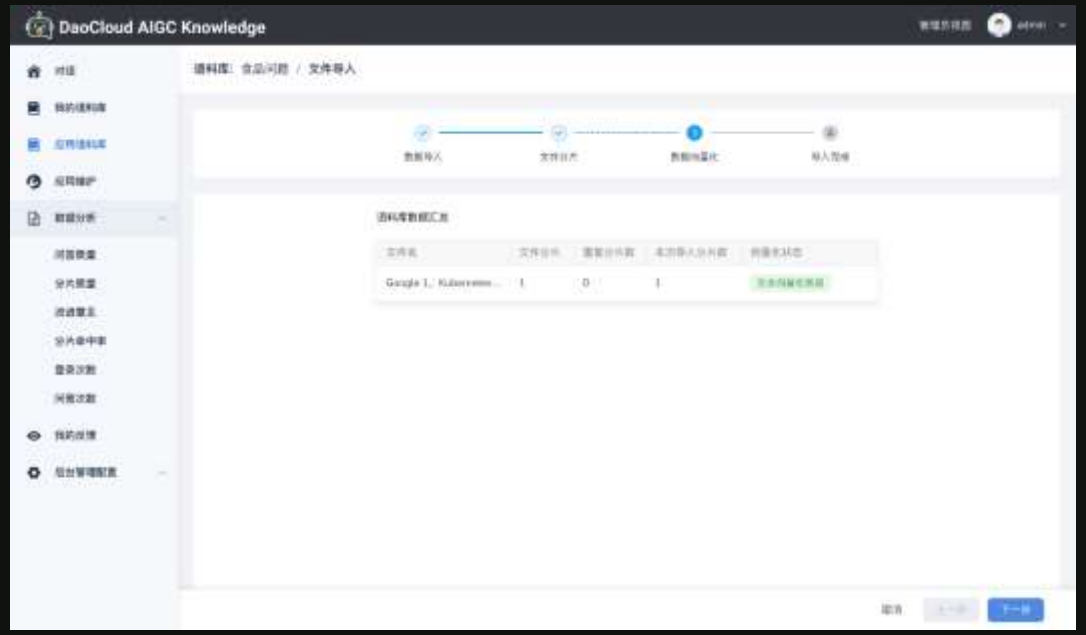


从 d.run 网站进入后点击智能问答的应用中心，搭建自己的知识库

案例 1-简单知识库-导入语料

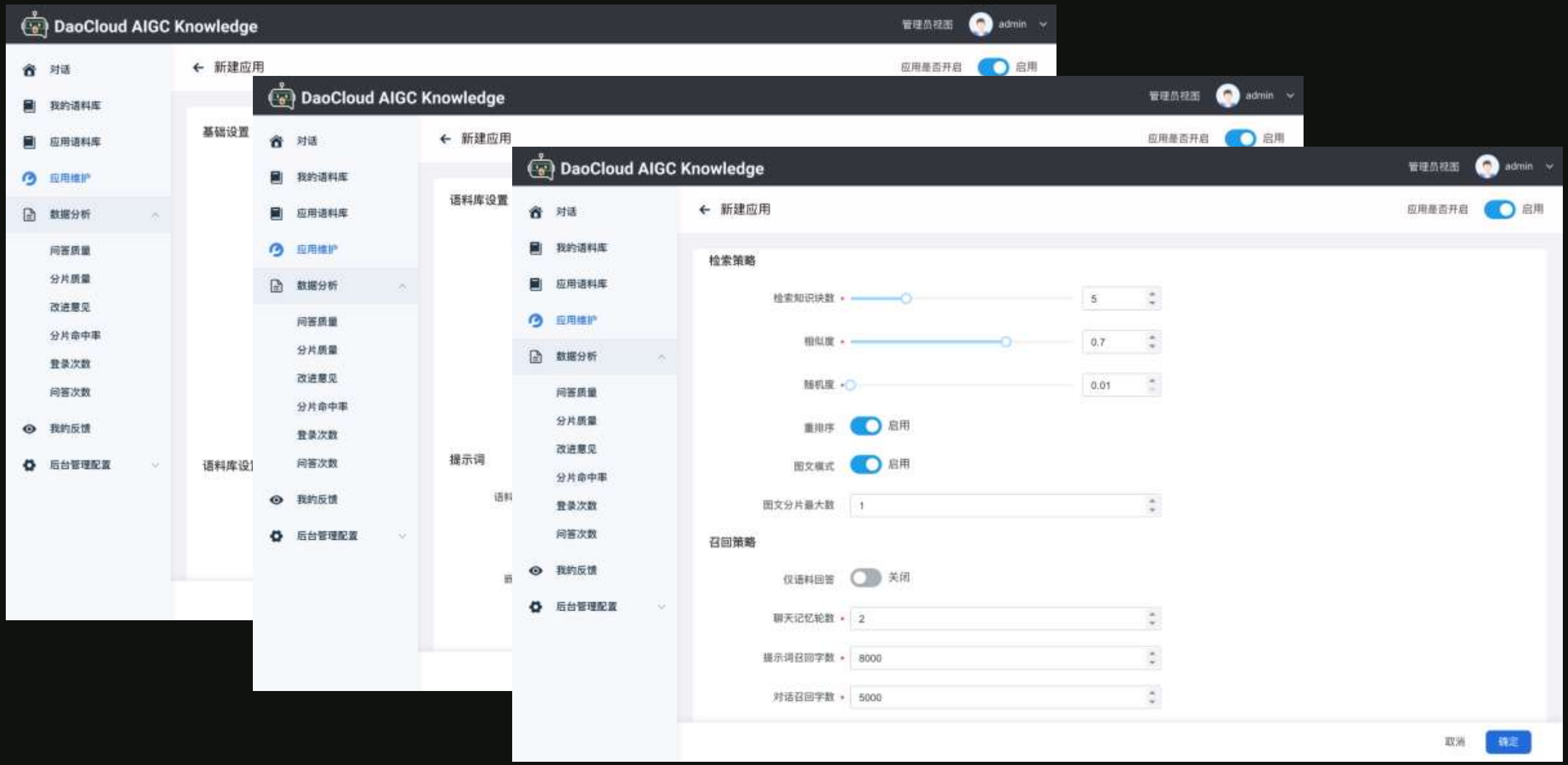






导入企业专有知识

案例1-简单知识库-创建应用



案例 1-简单知识库-发布应用并使用

csd
llama2-13b: 抱歉，我不太明白你的问题。可以请你提供更多的上下文
今天 14:29:45

图文助理
图文测试-openAI: ected+EOF
今天 14:14:12

食品安全问题是一个全球性的问题，
llama2-13b: 食品安全问题是一个全球性的问题，涉及到食品的生产、
今天 14:22:07

新建聊天
llama2-13b:
今天 14:08:57

食品安全问题是一个全球性的问题，
本地LLM通用助手1.0: 此模型者没有配置可用的LLM账号！

食品安全问题是一个全球性的问题，
本地LLM通用助手1.0: 此模型者没有配置可用的LLM账号！

今天 14:07:40

食品安全问题是一个全球性的问题，涉及到食品的生产、加工、储存、销售等各个环节。在中国，食品安全问题主要表现为食品制造过程中使用劣质原料、添加有毒物质、超量使用食品添加剂、抗生素和违禁药品的使用、农产品和禽类产品的安全状况不佳、转基因食品的潜在威胁等。此外，消费者缺乏食品方面的安全常识，以及有关部门的监管不力和法律法规体系不健全也是导致食品安全问题的原因之一。为了解决这些问题，政府已经采取了一系列措施，包括建立食品药品监督管理局、加强食品安全法规的制定和执行、提高食品安全标准等。同时，消费者也应该提高食品安全意识，了解食品安全知识，选择正规渠道购买食品，避免食用不健康的食品。如果不幸食用了不健康的食品，应该及时就医，并向有关部门举报。食品安全问题是一个全球性的问题，涉及到食品的生产、加工、储存、销售等各个环节。在中国，食品安全问题主要表现为食品制造过程中使用劣质原料、添加有毒物质、超量使用食品添加剂、抗生素和违禁药品的使用、农产品和禽类产品的安全状况不佳、转基因食品的潜在威胁等。此外，消费者缺乏食品方面的安全常识，以及有关部门的监管不力和法律法规体系不健全也是导致食品安全问题的原因之一。为了解决这些问题，政府已经采取了一系列措施，包括建立食品药品监督管理局、加强食品安全法规的制定和执行、提高食品安全标准等。同时，消费者也应该提高食品安全意识，了解食品安全知识，选择正规渠道购买食品，避免食用不健康的食品。如果不幸食用了不健康的食品，应该及时就医，并向有关部门举报。食品安全问题是一个全球性的问题，涉及到食品的生产、加工、储存、销售等各个环节。在中国，食品安全问题主要表现为食品制造过程中使用劣质原料、添加有毒物质、超量使用食品添加剂、抗生素和违禁药品的使用、农产品和禽类产品的安全状况不佳、转基因食品的潜在威胁等。此外，消费者缺乏食品方面的安全常识，以及有关部门的监管不力和法律法规体系不健全也是导致食品安全问题的原因之一。为了解决这些问题，政府已经采取了一系列措施，包括建立食品药品监督管理局、加强食品安全法规的制定和执行、提高食品安全标准等。同时，消费者也应该提高食品安全意识，了解食品安全知识，选择正规渠道购买食品，避免食用不健康的食品。如果不幸食用了不健康的食品，应该及时就医，并向有关部门举报。食品安全问题是一个全球性的问题，涉及到食品的生产、加工、储存、销售等各个环节。在中国，食品安全问题主要表现

有问题尽管问我
0/6000

发布知识库应用

案例 2-导购类应用场景 Agent场景

逻辑实现

Part 04

RAG 和 微调

RAG and Fine-tuning

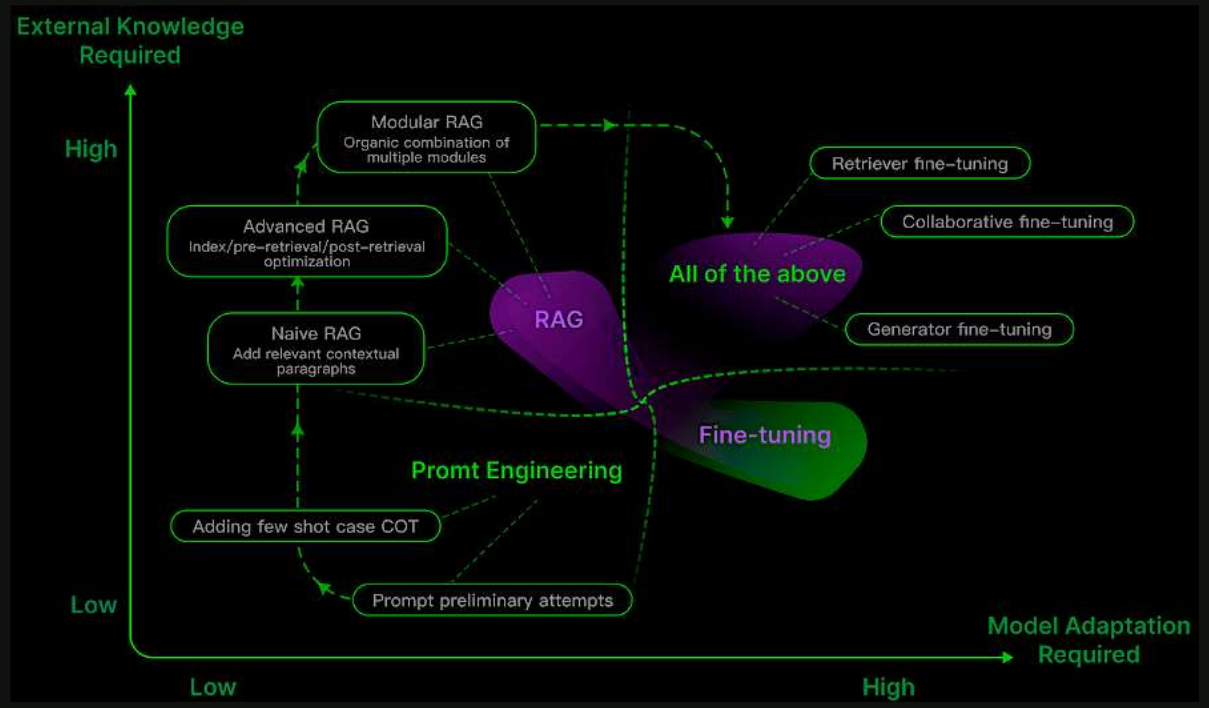
微调的介绍

适用场景

- 模仿特定结构
- 风格或格式

不适用场景

- 不适合添加全新知识
- 不适用快速迭代的场景



提升特定任务表现

在特定数据集上微调，增强准确性和效率。

定制输出

根据需求调整输出，适应特定格式、风格或结构。

强化已有知识

巩固知识库，增强对特定领域或任务的理解。

处理复杂指令

提高理解和执行复杂指令的能力，提升交互效率和响应质量。

降低训练成本

相比从头训练，微调在预训练模型上调整，显著降低计算资源和时间成本。

RAG和微调在企业场景落地上，其实是互补的关系

检索器微调（Retriever Fine-tuning）

- **定义**: 对模型中用于从大量数据中提取信息的关键部分进行特殊训练。
- **应用场景**: 适用于需要查找和利用外部知识的任务，如问答系统。
- **举例**: 就像在图书馆中训练一个图书管理员，使其能更快速、更准确地找到你需要的书。

生成器微调（Generator Fine-tuning）

- **定义**: 对模型中负责产生响应或输出的部分进行特殊训练。
- **应用场景**: 适用于文本生成模型，如创建新文本、写诗等。
- **举例**: 特别训练模型理解和运用诗歌的结构和韵律，使其生成的诗句更动人、更有感染力。

协同微调（Collaborative Fine-tuning）

- **定义**: 同时调整模型中的检索器和生成器，使其协同工作。
- **应用场景**: 提高整体性能，使检索器和生成器相互学习与配合。
- **举例**: 像一个团队中的每个成员相互学习与配合，以更高效、更完美地完成任务。

目前 RAG 技术的总结

优势

- **提高准确性**
  - **基于实际证据**：每个回答均基于检索到的实际证据，减少了错误或虚构回答的可能性，使生成的内容更加准确可信。
- **扩展性**
  - **动态性和实时响应性**：直接更新检索知识库，确保信息持续更新，无需频繁重新训练，适合动态变化的数据环境。
- **超强的定制化能力**：
  - **利用外部数据资源**：适合处理文档或其他结构化/非结构化数据库。
  - **跨语言能力和宽表对接**：对数据处理和操作要求较低。
- **可控性**
  - **透明度、可信度**：通过引用信息来源，用户可以核实答案的准确性，增强对模型输出结果的信任。
  - **安全性和隐私管理**：通过数据设置的角色和安全控制，实现更好的数据使用控制。

挑战

- **延迟的挑战**
  - **数据检索带来的延迟**：可能较高的延迟，相比之下，微调模型可以直接回应，降低延迟。
- **风格定制的挑战**
  - **信息检索和外部知识融合**：无法充分定制模型行为或写作风格。

Thanks.