**厂商智能客服介绍**

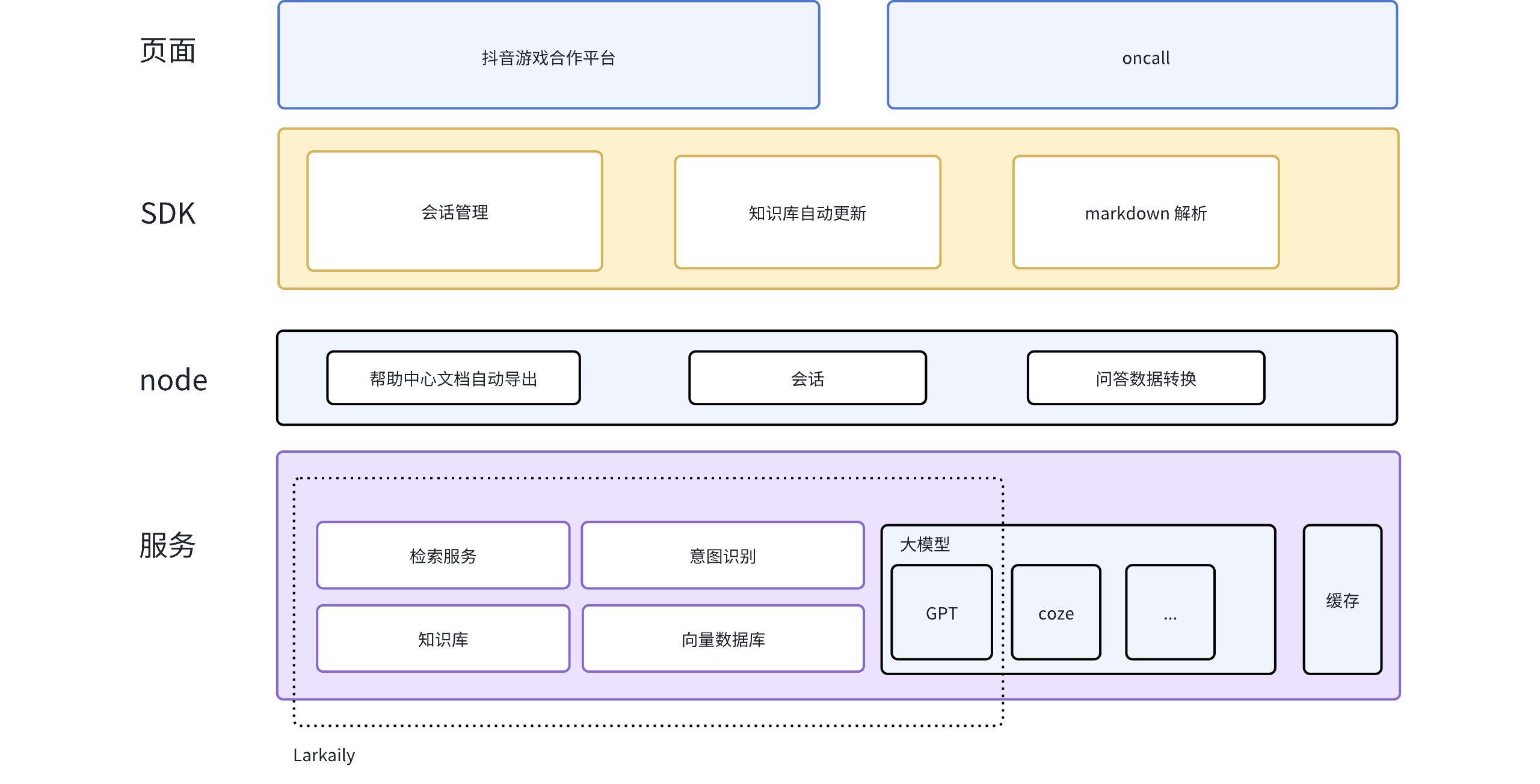
**目标**

* 提高用户体验和满意度，尽可能前置帮助用户解决问题
* 提供一些标准化的回答，为人工客服分担重复工作，减少人力成本提高效率
* 自动更新最新的平台功能回复

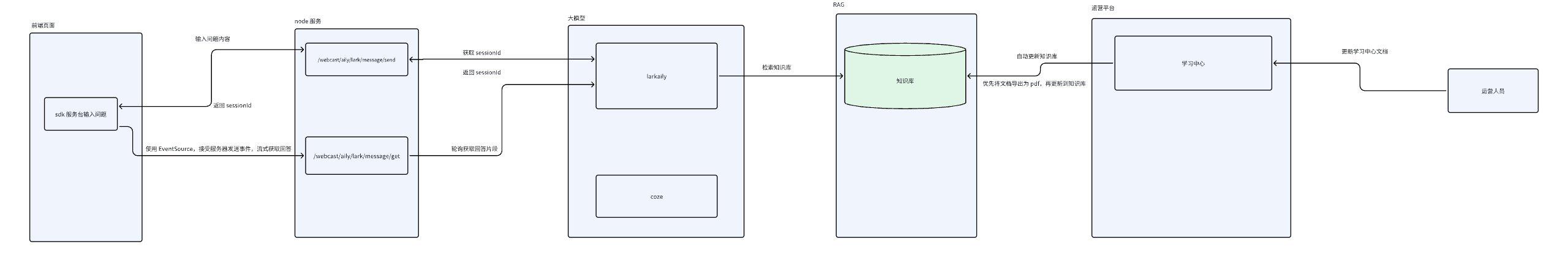
**前端交互**

|  |  |
| --- | --- |
| 描述 | 图示 |
| 如果查询不到答案可以根据用户的问题提供一些相关的问题 |  |
| 问题回复以流的形式展示，在未完成回答时会实时展示 loading，直至问题回答结束 |  |
| 如果如有需要可以使用人工服务 |  |
| 可以直接获取人工服务 |  |

**整体架构**

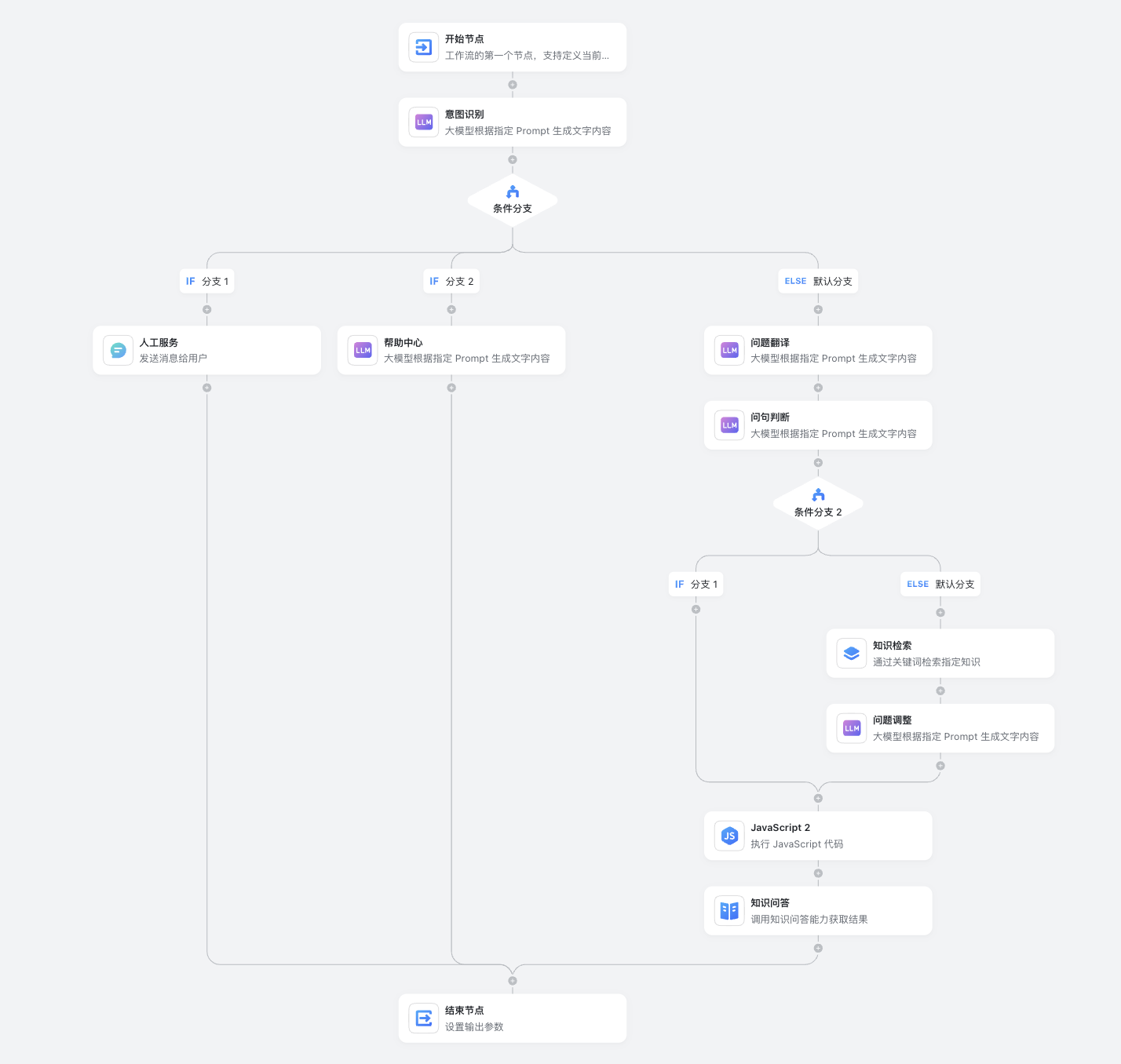


**整体流程**



**核心工作流**

* 接受用户输入后，优先**判断用户意图**，分别为
* 人工服务
* 帮助中心跳转
* 问题预处理
* 在提问处理中，需要优先将问题处理翻译成中文
* 然后判断问句是否完整，如果完整则返回翻译后的问句，
* 如果不完整，那么需要根据从知识库检索到的内容和提问问题结合，补充出一个完整的问句
* 最终将处理后的问题作为最终提问，给到 LLM 大模型，大模型会基于 RAG 外接知识库检索，组织出最终的答案，如果找不到答案，会根据知识库内容返回可能的猜你想问问题，让用户备选



**技术实现**

**大模型的使用**

**GPT原理**

GPT（Generative Pre-trained Transformer）是一种基于Transformer架构的自然语言处理模型，广泛应用于文本生成、对话系统、机器翻译等任务。GPT的核心思想是通过自回归的生成方式，在预训练阶段学习大规模文本数据中的语言规律，再通过微调（fine-tuning）使模型能够执行特定任务。

**优点**

* **强大的语言生成能力**

GPT能够生成流畅且自然的文本，具有较高的语言表现力。它能够在没有明确指令的情况下生成连贯的文章、对话、摘要等，表现出较强的语言理解和生成能力。

* **无需任务特定训练（Zero-shot / Few-shot Learning）**

GPT不需要针对特定任务进行专门的训练，而是通过预训练过程学习到广泛的语言模式和知识。它可以在**零样本学习**（Zero-shot）或**少样本学习**（Few-shot）环境下执行许多任务，比如翻译、问答、情感分析等，无需大量标注数据。

* **上下文理解**

GPT能够理解输入文本的上下文，并生成与之相关的内容。它使用**自注意力机制**（Self-Attention）来捕捉长程依赖关系，从而生成与上下文一致的连贯文本。

* **易于集成**

GPT可以作为一个基础模型，与其他系统或工具集成，提供自然语言生成和理解能力。这使得它在许多实际应用中具有很高的适应性，例如聊天机器人、智能客服、内容创作等。

**缺点**

* **幻觉问题**

GPT的**生成过程是基于概率的**，这意味着它可能会生成虚假的信息或没有依据的内容，这种现象被称为**幻觉（Hallucination）**。尽管生成的文本可能看起来合乎逻辑，但在某些情况下，模型会生成错误的事实或完全虚构的内容。

* **对事实的验证能力差**

GPT无法主动验证事实，因为它缺乏外部知识检索的能力。虽然它能够基于上下文生成看似合理的文本，但如果没有外部知识库或事实验证机制，生成的内容可能并不准确。例如，模型可能在回答问题时引入错误的日期、地点或事件。

* **缺乏深层推理能力**

GPT虽然能够生成连贯的语言，但它缺乏深入的推理能力。在面对需要多步推理、逻辑推导或复杂决策的任务时，GPT往往会表现得不如预期。例如，在需要解决数学问题或进行复杂推理时，模型可能无法给出正确的答案。

**对比与联系**



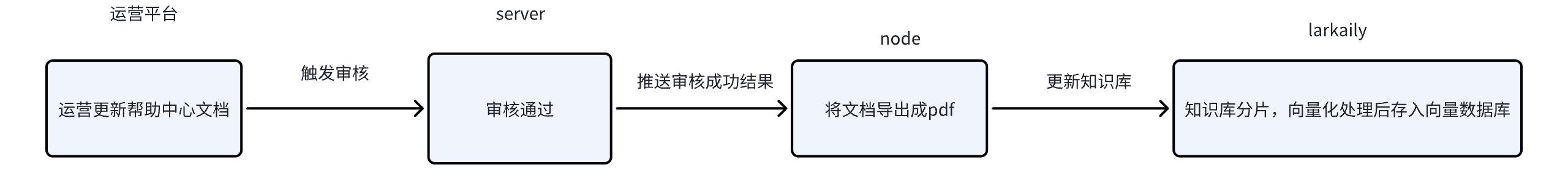
**Click the image to view the sheet.**

**知识库建设**

运营后台维护着一套通过审核的最新的帮助中心文档数据，这个数据用于提供给 larkaily 作为外接知识库来使用，在node 服务中调用飞书提供的文档 openapi，将文档下载成 pdf 的格式，再将 pdf 导入到 larkaily 的知识库中

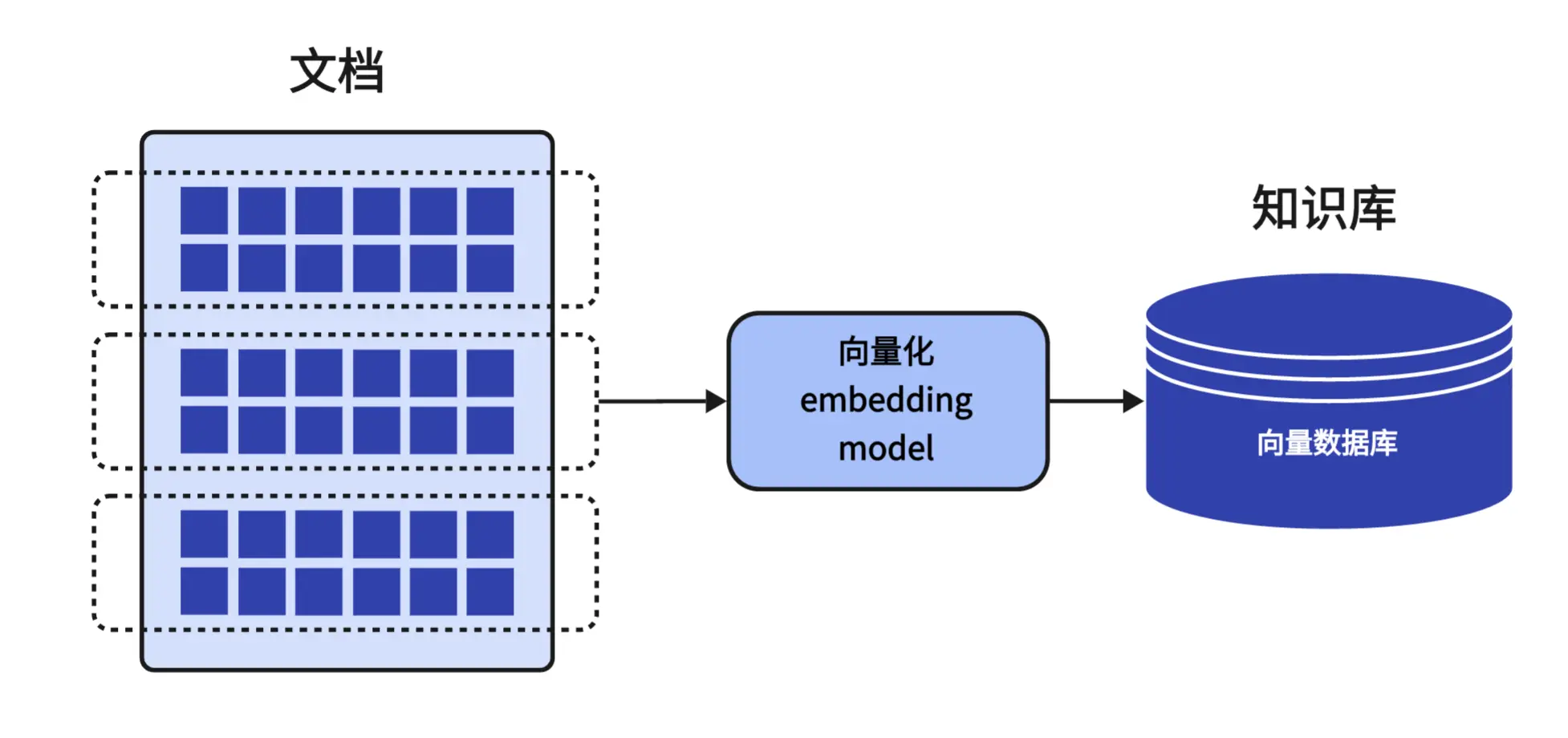
|  |  |
| --- | --- |
|  |  |

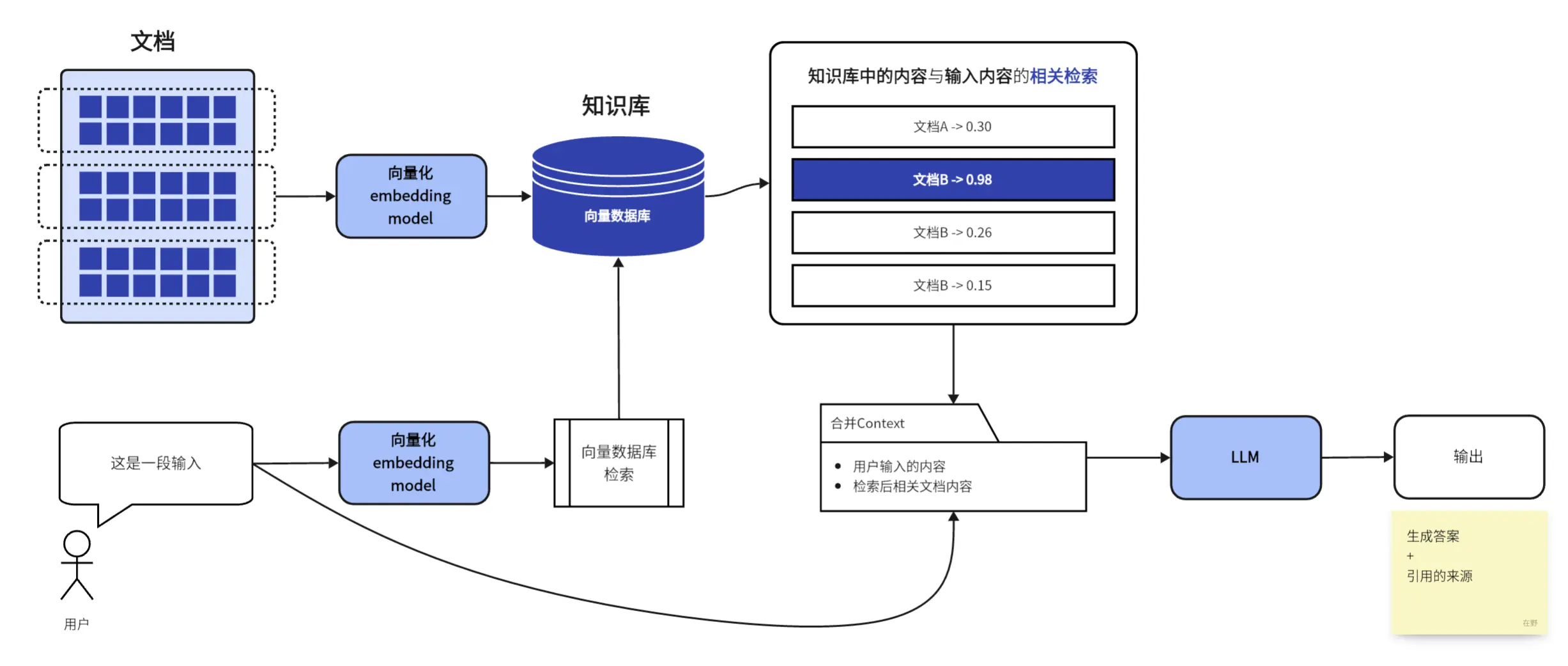
**链路设计**



**外接知识库 RAG**

检索增强生成（RAG）是指对大型语言模型输出进行优化，使其能够在生成响应之前引用训练数据来源之外的权威知识库。大型语言模型（LLM）用海量数据进行训练，使用数十亿个参数为回答问题、翻译语言和完成句子等任务生成原始输出。在 LLM 本就强大的功能基础上，RAG 将其扩展为能访问特定领域或组织的内部知识库，所有这些都无需重新训练模型。这是一种经济高效地改进 LLM 输出的方法，让它在各种情境下都能保持相关性、准确性和实用性。





**RAG 的核心原理**

RAG 的工作原理分为两个主要阶段：

* **检索（Retrieval）**

这一阶段的目标是从外部知识库中找到与输入查询最相关的信息。

* **知识库**：通常是一个包含结构化或非结构化文档的大型数据库，如百科全书、产品说明书、或者语料库。
* **检索器**：用于从知识库中检索相关信息的模块。常用技术包括：
* **基于稀疏向量的检索方法**：如 TF-IDF、BM25，它们通过关键词匹配寻找相关文档。
* **基于密集向量的检索方法**：如使用双塔（dual-encoder）架构的深度学习模型（例如 DPR，Dense Passage Retrieval），将查询和文档编码为向量，在向量空间中计算相似度。
* **嵌入向量索引工具**：如 FAISS，用于高效地在向量空间中检索最近邻的文档。

通过检索阶段，从知识库中返回一个或多个与输入查询相关的文档或片段。

* **生成（Generation）**

这一阶段将检索到的相关文档与输入查询结合起来，通过生成模型生成最终的输出。

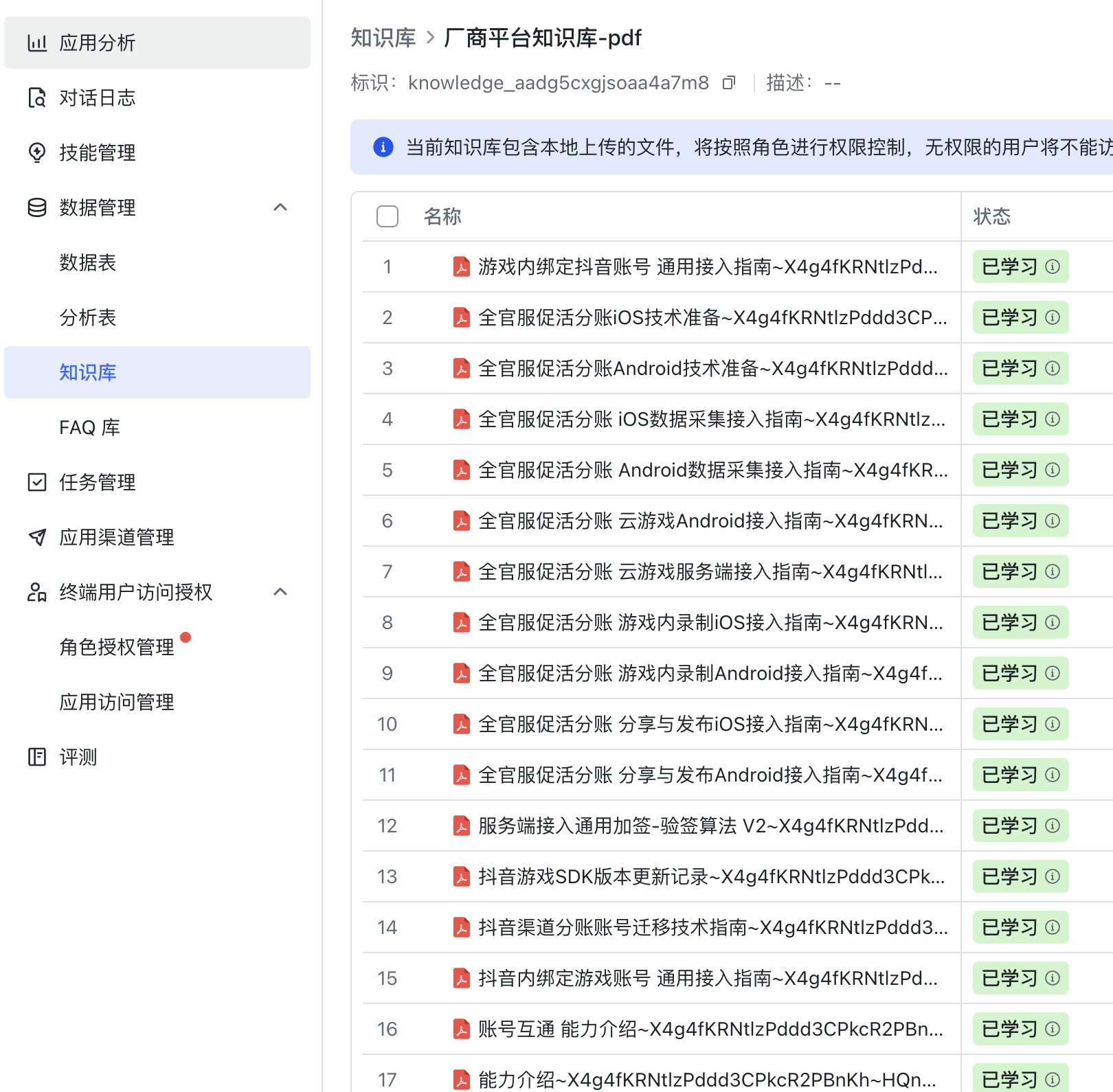
* **输入拼接**：将用户的查询和检索到的文档片段拼接成一个输入序列，输入到生成模型中。
* **生成模型**：通常是一个基于 Transformer 的预训练生成模型（如 T5、BART、GPT）。
* 生成模型会根据拼接后的输入序列生成答案，利用检索到的文档内容补充上下文知识。

**RAG 的主要特点**

* **知识动态性**：无需在生成模型中固化知识，可以通过更新知识库动态调整系统的知识范围。
* **高效性**：利用检索器缩小生成模型的知识需求，避免生成模型生成“胡编”的内容。
* **可解释性**：输出结果可以溯源到知识库中的具体文档，增强结果的可信度。

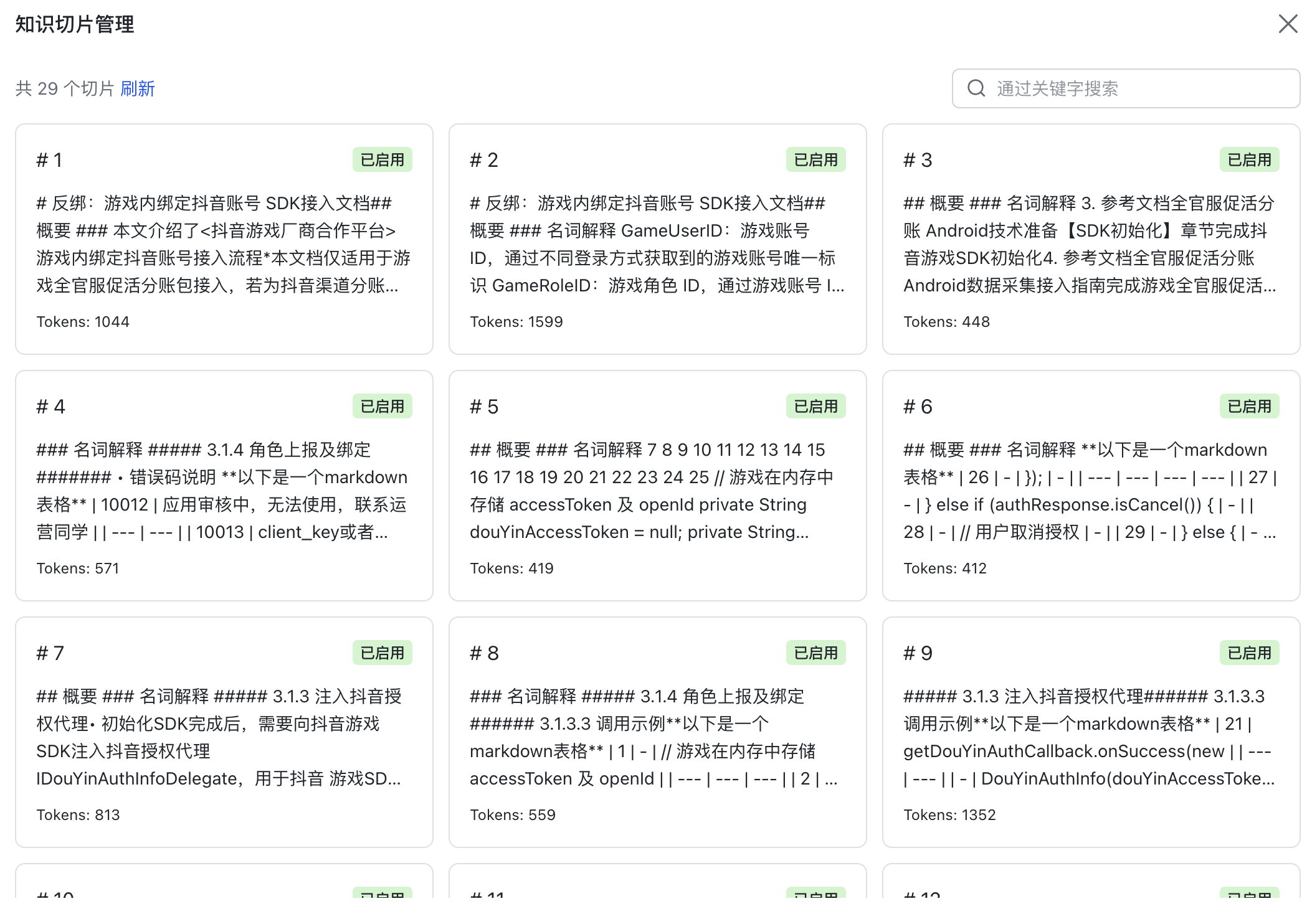
**RAG 示例**

知识库和运营平台的学习中心打通，会自动将新的学习中心文档，导出为 pdf 之后更新到知识库中，大模型能迅速学习到最新的知识。



向量化知识切片的流程包括：

* **知识拆分**：将文本按粒度拆分为小块（Chunk）。
* **按句子**：将文本按句子边界拆分。
* **按段落**：按段落或逻辑块进行拆分。
* **固定长度**：将文本按固定字数（如 200~500 字）进行切分。
* **语义分割**：利用 NLP 模型识别文本的语义边界，按内容主题进行分割。
* 文本**向量化（Embedding）**：使用嵌入模型将每个切片转化为向量表示。
* **使用预训练嵌入模型**： 如 OpenAI 的 text-embedding 模型、sentence-transformers、BERT、ERNIE 等模型。
* **生成文本嵌入向量**： 输入文本切片到嵌入模型中，输出一个固定维度的向量表示。
* **示例工具**：
* Hugging Face 的 sentence-transformers 库。
* OpenAI Embedding API，例如 text-embedding-ada-002。
* **存储向量**：将向量和对应元数据存储到向量数据库中。
* **查询检索**：用户查询通过向量相似度匹配，返回相关的知识切片。



**GraphRAG**

* 相比于普通 RAG， 一般是使用向量数据库，通过向量检索的方式，检索出相关的 TopK 片段分块 Chunks，
* GraphRAG 则是将文档内容，先处理成知识图谱的形式，其中包含了各个实体间的关联关系，查询时将相关联的实体作为上下文 Context 和问题一起给到 LLM 大模型，以提高大模型回答的准确性



**Click the image to view the sheet.**

**优点**

**增强的推理能力**

* **GraphRAG** 利用知识图谱（Graph Database）来存储结构化信息，并与大语言模型结合，帮助系统更好地理解和推理复杂的实体关系。这使得它能够有效处理需要多实体推理的任务，如复杂的问答系统和推荐系统。

**减少语言模型的幻觉问题**

* 由于知识图谱提供了明确的结构化信息，**GraphRAG** 能减少大语言模型生成虚假信息或不相关内容的风险。图数据库中的结构化关系能够帮助模型更好地推理出正确的答案。

**可解释性与透明度**

* 图数据库提供了可追溯的实体关系和推理路径，增强了系统的可解释性。在一些重要应用场景（如法律、医疗）中，能够提供对模型决策过程的清晰理解，是非常重要的优势。

**灵活性与扩展性**

* **GraphRAG** 支持动态构建和更新知识图谱，这意味着可以轻松地将新信息添加到图数据库中。随着图谱数据的增多，模型可以根据新信息自动调整和优化其推理过程。

**适应复杂关系推理**

* 图数据库非常适合处理复杂的关系推理，如产品推荐、人员关系、历史事件等，**GraphRAG** 能够通过图形化的数据结构来有效管理和推理多维度的关系信息。

**缺点**

**图谱构建与维护的复杂性**

* 构建和维护一个准确且完备的知识图谱可能是非常耗时和复杂的过程。需要高质量的数据源和大量的人工标注来确保图谱的准确性。此外，随着时间的推移，知识图谱可能需要频繁更新，以确保其保持最新状态。

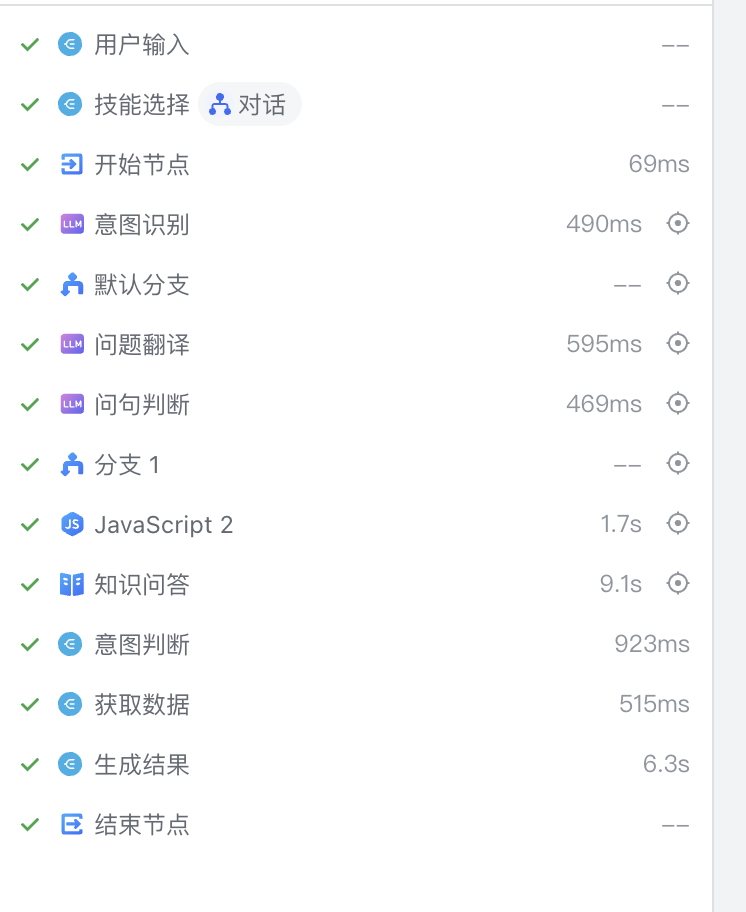
**计算和存储开销**

* 图数据库和知识图谱通常需要大量的计算和存储资源来支持其查询和更新操作。对于资源有限的环境（如小型服务器或边缘设备），运行图数据库可能会带来性能瓶颈。

**推理质量依赖于图谱质量**

* 如果知识图谱中的实体或关系数据不完整或错误，模型的推理能力也会受到影响。图谱数据的质量直接影响生成的答案准确性，因此如何维护高质量的图谱是一个挑战。

**效果展示**



**IM消息**

**服务端推送 Server-Sent Events**

* **EventSource (SSE - Server-Sent Events)**:
* **单向通信**：EventSource 仅支持服务器向客户端推送数据，客户端不能通过这个通道向服务器发送数据。
* **基于 HTTP/1.1**：EventSource 是建立在 HTTP 协议之上的，使用的是标准的 HTTP 请求（通常是 GET 请求）。
* **文本传输**：服务器发送的数据必须是**纯文本**，通常以 text/event-stream 格式发送。
* 和HTTP轮询相比，更轻量和节约服务器资源
* **WebSocket**:
* **双向通信**：WebSocket 支持全双工通信，即客户端和服务器都可以在连接建立后随时发送数据。
* **基于 WebSocket 协议**：WebSocket 使用的是独立的 WebSocket 协议，它在初始握手时使用 HTTP/HTTPS，但连接建立后切换到 WebSocket 协议。
* **二进制和文本传输**：WebSocket 支持传输二进制数据和文本数据，提供了更大的灵活性。
* 浏览器侧使用 EventSource 接收服务器事件发送，能够实现流式的内容传输
* Response 的 Content-type：text/event-stream

|  |
| --- |
| TypeScript // 浏览器 https://developer.mozilla.org/zh-CN/docs/Web/API/EventSource const eventSource = new EventSource(  `${url}`,  {  withCredentials: true,  }, );  eventSource.onmessage = event => {  // data 支持自定义  const data = event.data;  if (data.event === 'done') {  eventSource.close();  }  };    /\* 格式 event: userconnect data: {"username": "bobby", "time": "02:33:48"}  data: Here's a system message of some kind that will get used data: to accomplish some task.  event: usermessage data: {"username": "bobby", "time": "02:34:11", "text": "Hi everyone."} \*/ // 服务端 https://developer.mozilla.org/en-US/docs/Web/API/Server-sent\_events/Using\_server-sent\_events  // content-type 需要设置为 text/event-stream import { PassThrough } from 'stream'; const stream = new PassThrough(); const interval = setInterval(() => {  stream.write(`data: ${JSON.stringify({a:1, b:2})}\n\n`); },1000)  setTimeout(() => {  stream.end(); }, 10000);  stream.on('close', () => {  clearInterval(interval); }); |

**根据不同诉求返回不同指令**

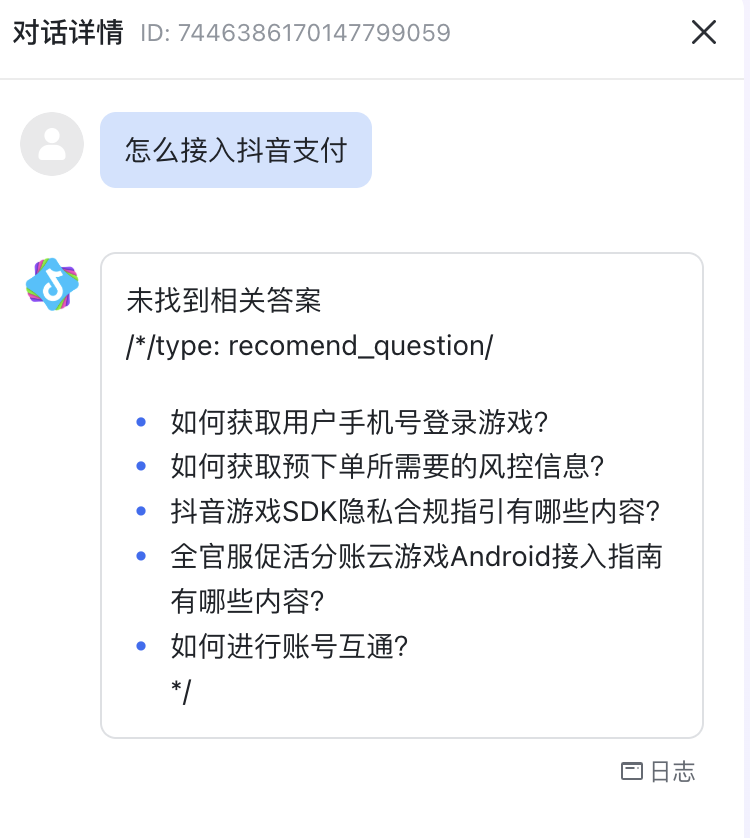
* 用户希望直接打开人工客服





* 没找到相关答案时，猜你想问





**根据来源跳转到对应的页面**

Larkaily 会返回回答中参考的知识库文档的名称，需要在文档名称中携带上页面链接信息，模版为"~[父级目录]~[文章id]"，在 node 服务中从模版里解析出对应的信息并拼接成链接返回

