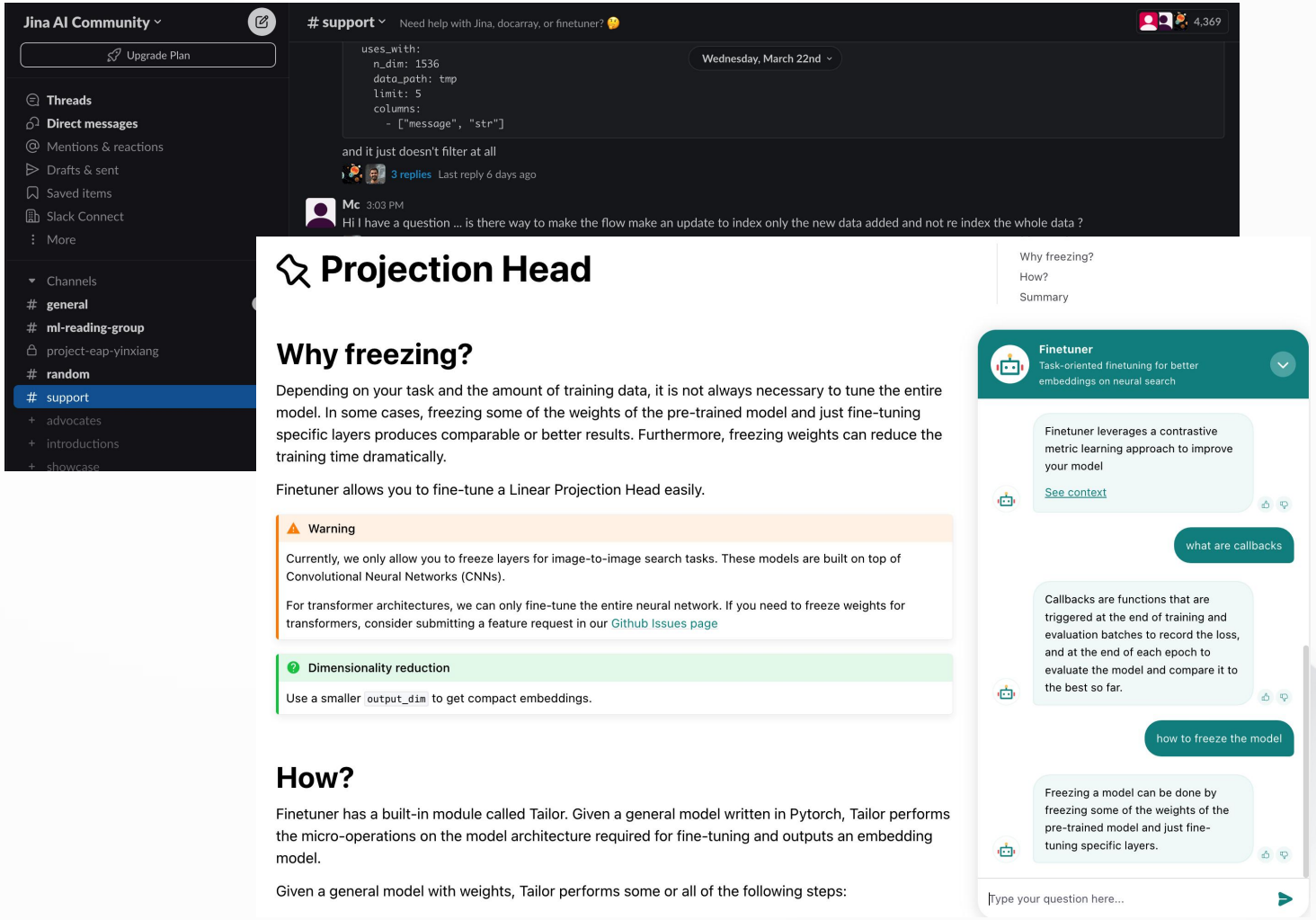

GOTC 2023
全球开源技术峰会
THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE
OPEN SOURCE, INTO THE FUTURE
AI is Everywhere 专场
本期议题：LLM在智能文档问答的应用
基于Langchain和Langchain-serve的解决方案
王楠 2023年05月28日

背景介绍

来自开源社区的大量问题

- 及时回答耗费开发者的时间
- 问题反复重复
- 用户很难通过网站的搜索框解决问题



问题定义：文档问答系统

二级标题

- 输入
  - Q：问题
  - D：文档集合
- 输出
  - A：答案

Q: What is Jina?

D: docs.jina.ai

文档问答系统

A: Jina is an MLOps framework

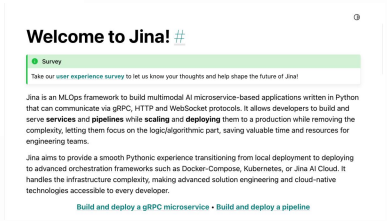
文档问答系统的算法范式

两阶段方法

- 召回阶段
  - 获取候选文档
- 阅读理解阶段
  - 抽取答案

Q: What is Jina?

D: docs.jina.ai



A: Jina is an MLOps framework

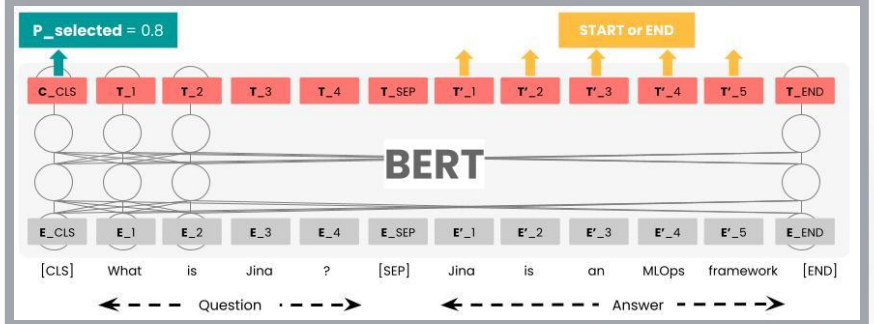
召回阶段

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

$$\text{tf}(t, d) = \log\left(1 + \text{freq}(t, d)\right)$$

$$\text{idf}(t, D) = \log\left(\frac{|D|}{|d \in D : t \in d|}\right)$$

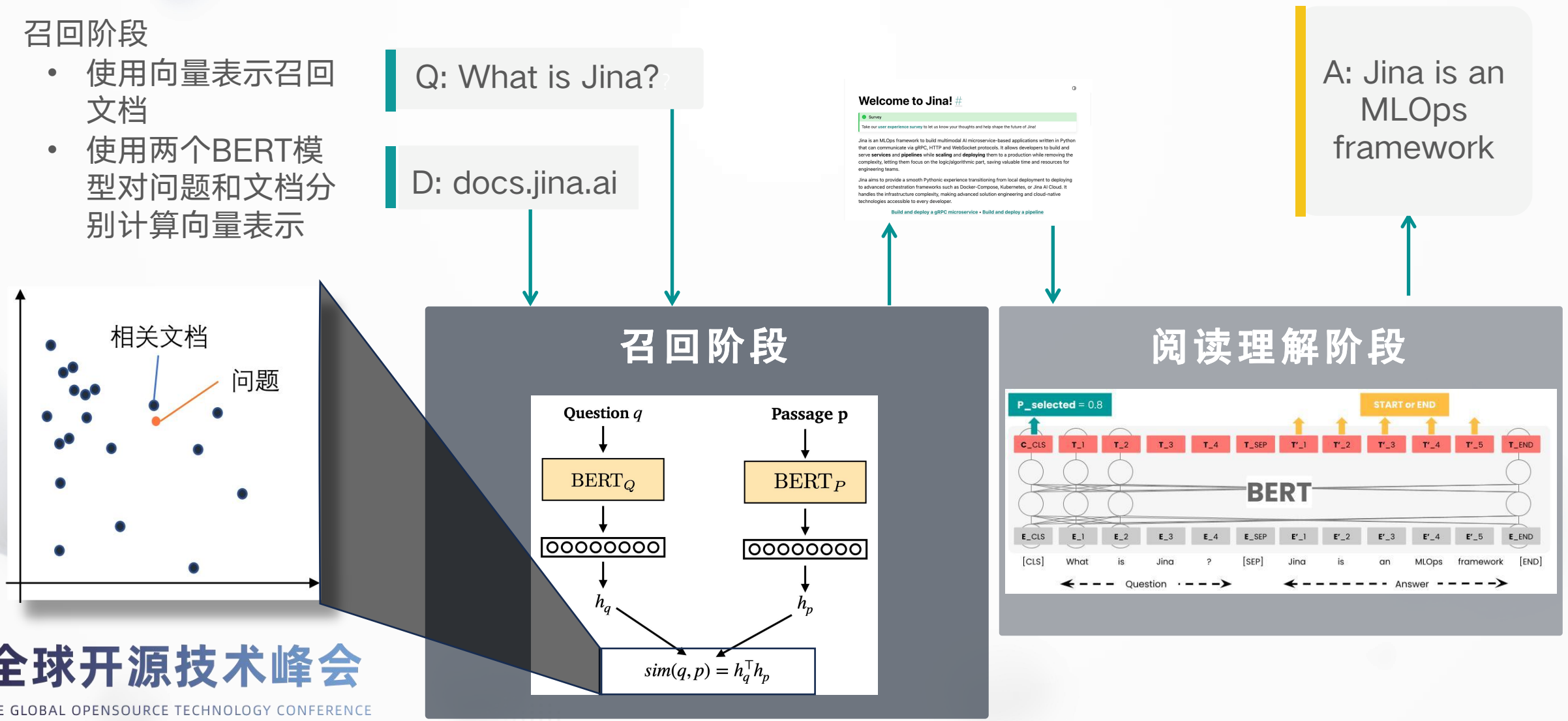
阅读理解阶段



文档问答系统的算法范式

端到端方法

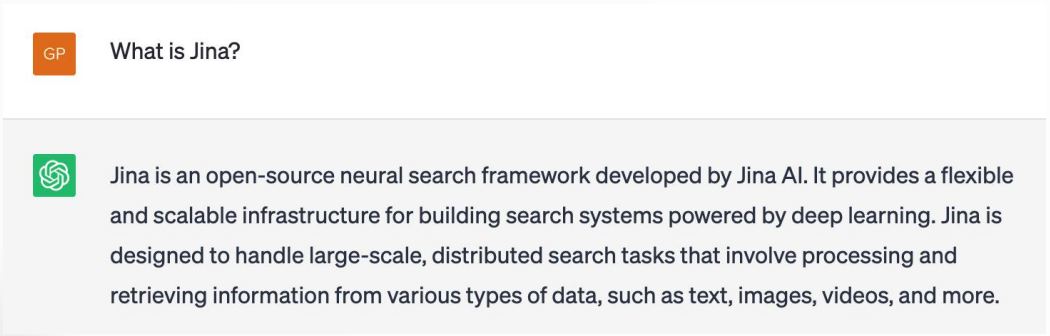
- 召回阶段
  - 使用向量表示召回文档
  - 使用两个BERT模型对问题和文档分别计算向量表示

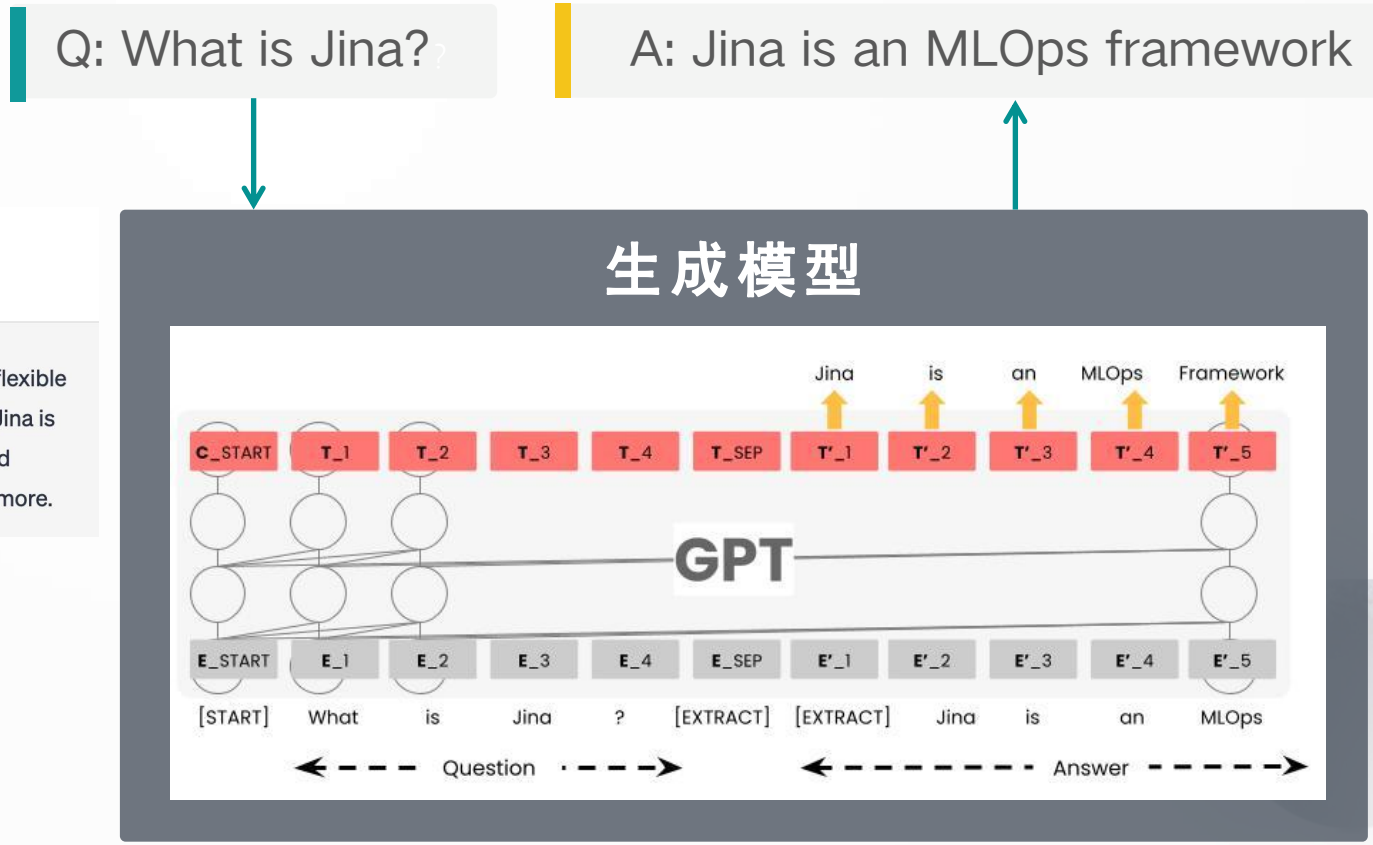


文档问答系统的算法范式

生成式方法

- 预训练语言模型可以记忆知识
- 使用以 GPT 为代表的生成式模型





文档网站问答系统的难点

模型开发成本高

- 预训练模型不达标
- 微调模型缺少语料

模型部署开销高

- 运算资源要求高
- 微调模型成本高

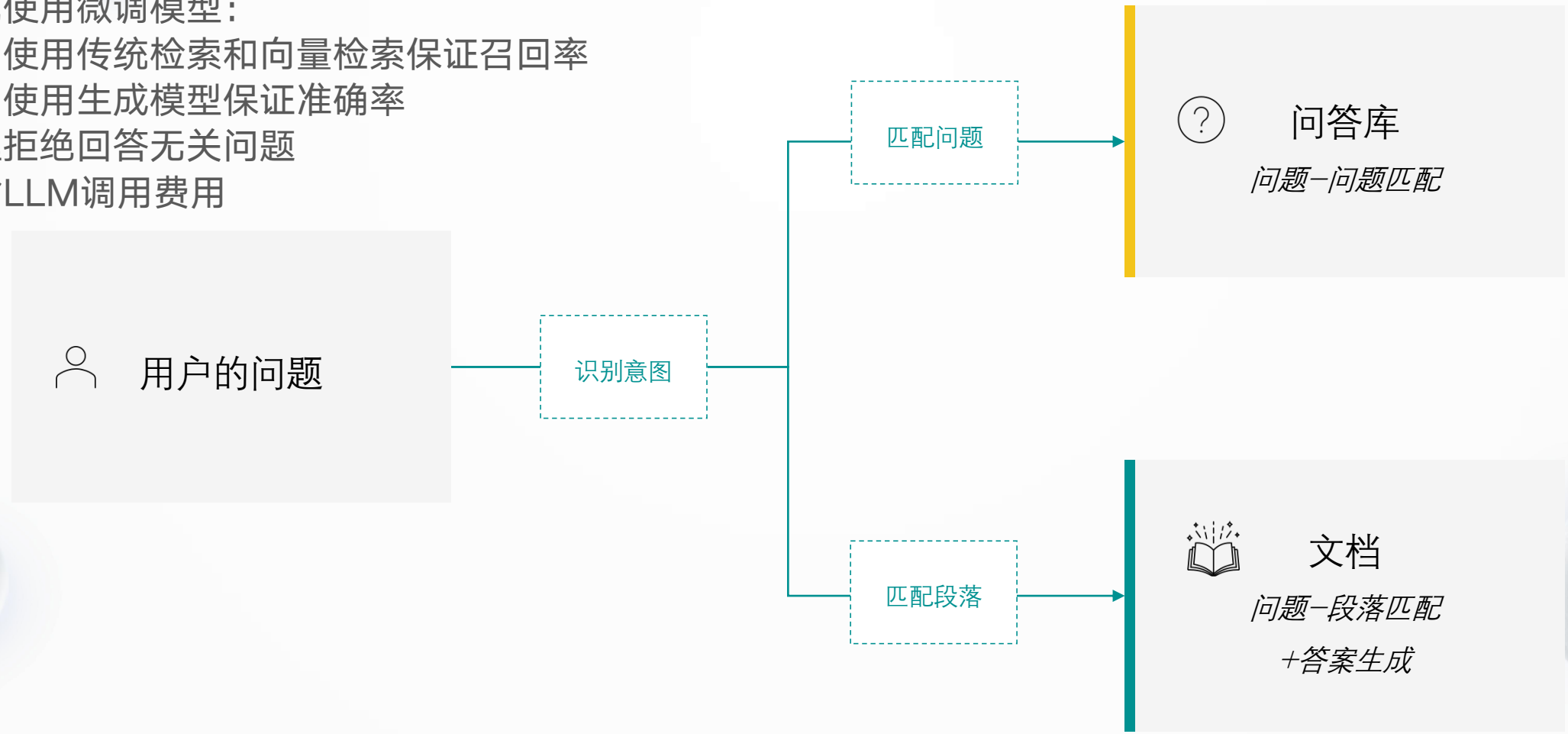
低频应用场景

- 突发性能要求高

算法解决方案

算法设计

- 避免使用微调模型：
  - 使用传统检索和向量检索保证召回率
  - 使用生成模型保证准确率
- 合理拒绝回答无关问题
- 节省LLM调用费用

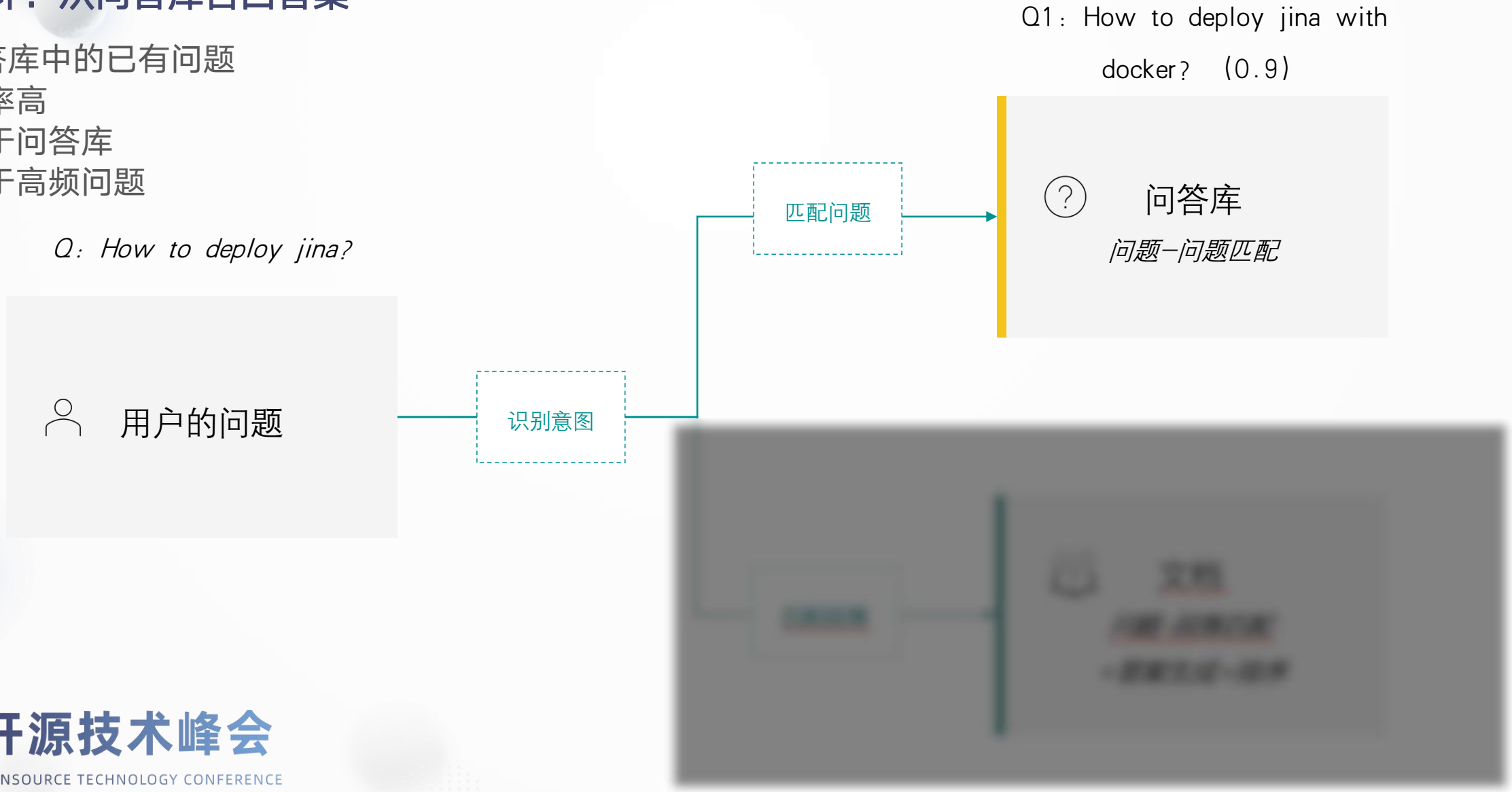


算法解决方案

算法设计：从问答库召回答案

匹配问答库中的已有问题

- 准确率高
- 依赖于问答库
- 适合于高频问题



算法解决方案

算法设计：从文档召回答案

匹配文档内容

- 覆盖面广，通用性强
- 准确率有限
- 适合于长尾问题

用户的问题

识别意图

匹配段落

文档

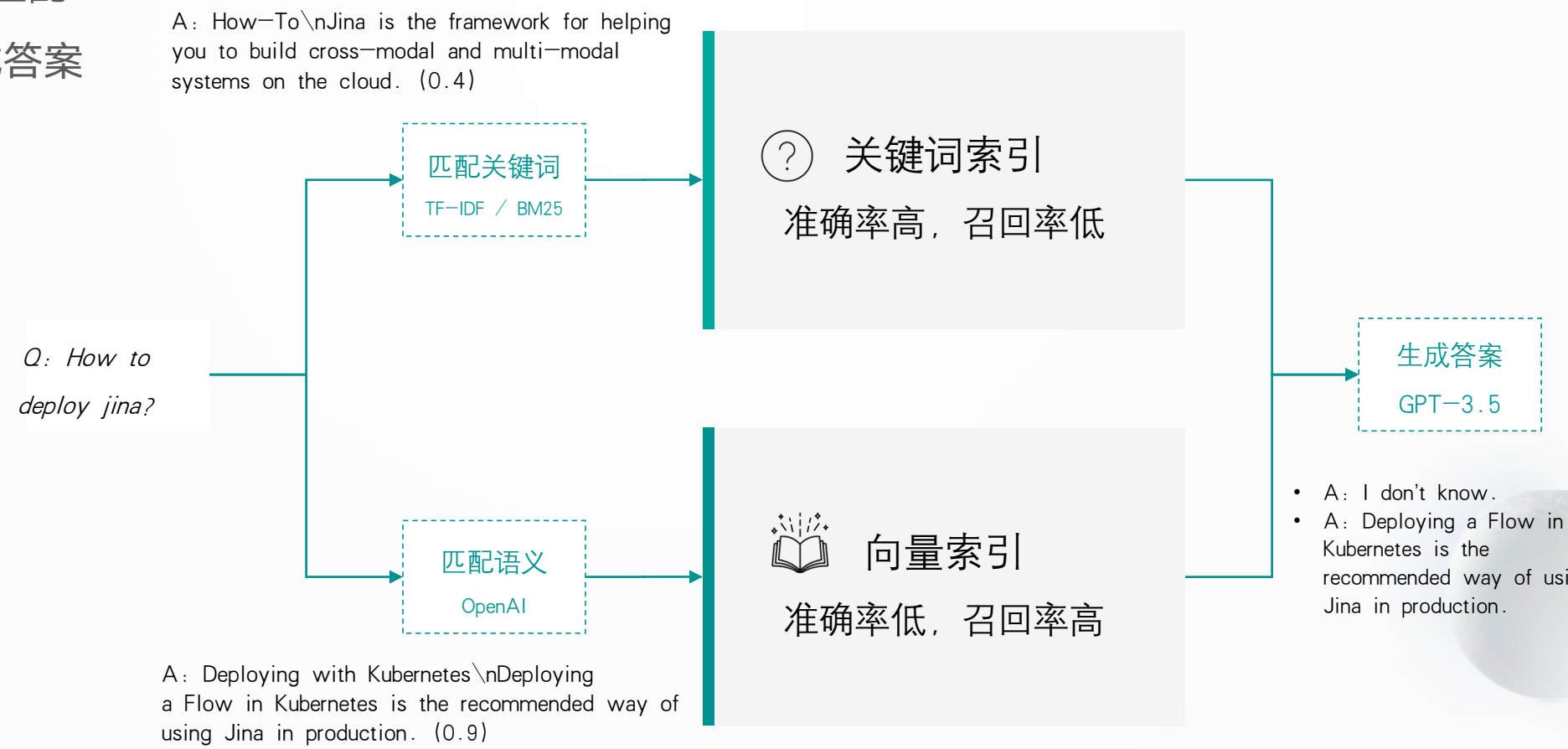
问题–段落匹配

+答案生成+排序

算法解决方案

算法设计：从文档召回答案

- 关键词匹配+向量匹配
- 使用GPT-3.5生成答案



算法解决方案

算法设计：使用GPT-3.5的生成答案

使用ChatGPT优化Prompt

BEFORE

Use the following pieces of context to answer the question at the end. If you don't know the answer, just say that you don't know, don't try to make up an answer.
——————————

{context}

Question：{question}
Answer：

AFTER

As a highly intelligent QA bot trained on billions of software documentation, your task is to provide accurate and relevant answers based on the given context. You must not rely on external knowledge but only use the information provided in the "Context" section. If you cannot infer an answer from the context, please give your best guess while emphasizing that it's a guess, not a fact. If you can't find any reliable answer, reply with "Unknown."

Remember to maintain accuracy and relevance in your responses while being flexible enough to handle various contexts and languages.

Please follow this format for your input：

- Context：{context}
- Question：{question}

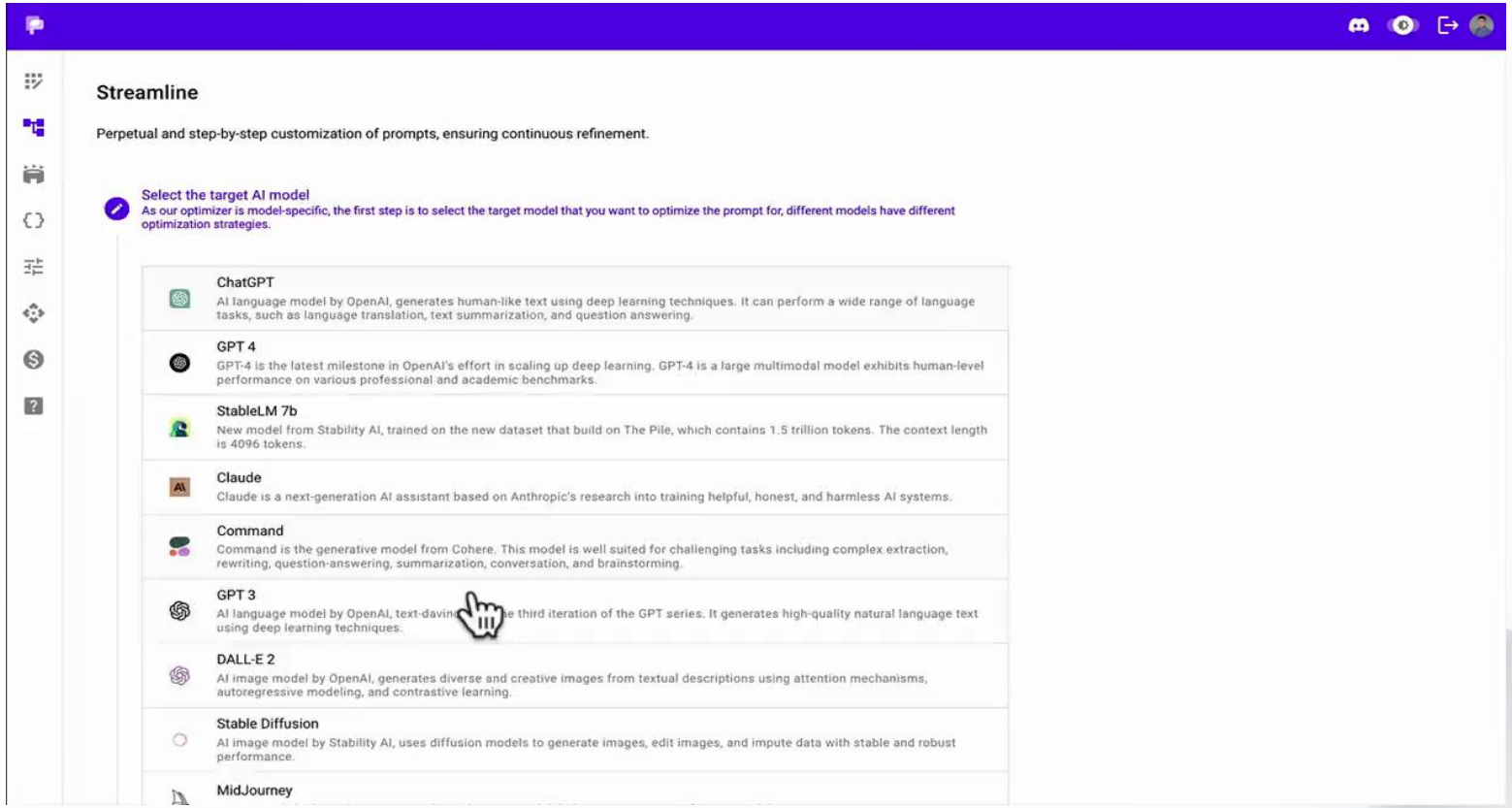
Your response should be structured like this：
- Answer：{answer}

优化提示词解决方案

PromptPerfect：使用GPT-3.5优化答案

使用ChatGPT优化prompt

1. 选择AI模型
2. 输入原始prompt
3. 优化prompt
4. 查看输出
5. 缩短prompt



工程解决方案

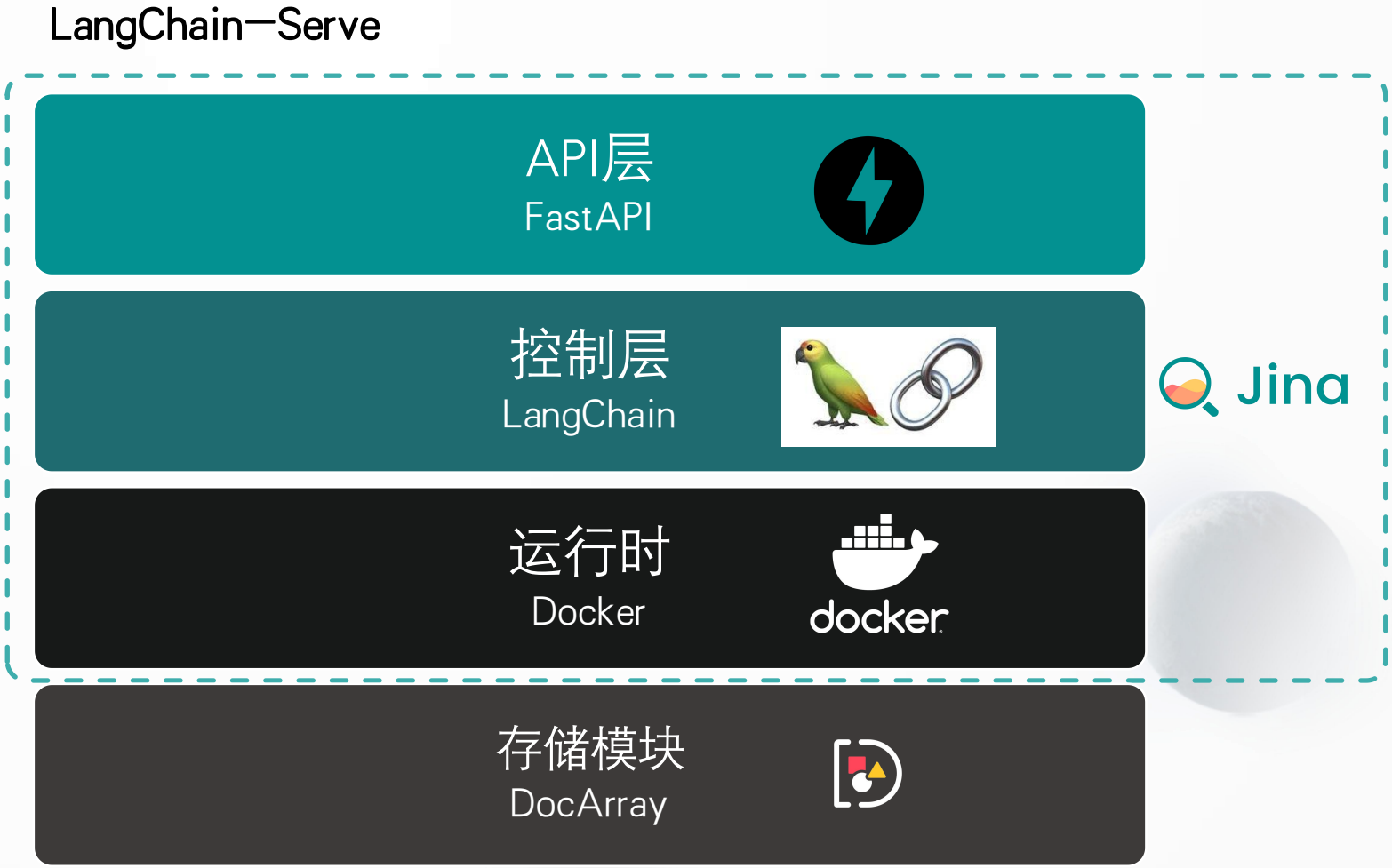
工程设计：从MLOps到LLMOps

- 控制层的智能化
  - 动态调用不同模型
  - 状态管理
- AI模型的使用范式转移
  - 大模型API调用取代传统模型部署
- 多模态数据的处理
  - 高效处理多模态数据

工程解决方案

工程设计：LLMOps

- 使用LangChain进行应用开发
- 使用LangChain-serve进行服务部署
- 使用DocArray进行存储



工程解决方案

工程设计：使用LangChain开发问答系统

- 针对LLM专门设计的开发框架
  - Prompt管理
  - 引入Agent概念
  - 适合本地调试
- 预制针对问答系统的模块(Chains)
  - ConversationalRetrievalChain

README.md

🦜🔗 LangChain

⚡ Building applications with LLMs through composability ⚡

lint passing | test passing | linkcheck passing | downloads/month 1M | License MIT | Follow @LangChainAI | LangChain

Dev Containers Open | Open in GitHub Codespaces | Stars 41k

Looking for the JS/TS version? Check out LangChain.js.

Production Support: As you move your LangChains into production, we'd love to offer more comprehensive support. Please fill out this form and we'll set up a dedicated support Slack channel.

Quick Install

`pip install langchain` or `conda install langchain -c conda-forge`

🤔 What is this?

Large language models (LLMs) are emerging as a transformative technology, enabling developers to build applications that they previously could not. However, using these LLMs in isolation is often insufficient for creating a truly powerful app - the real power comes when you can combine them with other sources of computation or knowledge.

工程解决方案

工程设计：使用LangChain-serve部署服务

- 为LangChain开发的应用提供部署服务
  - 添加@serving修饰器，即可实现云端部署
- 基于Jina开发
  - 将Chains以独立容器形式进行部署，实现弹性伸缩
  - 支持Serverless
- 服务部署在Jina AI Cloud

app.py

```python
from lcserve import serving
from langchain import OpenAI

def get_chain():
    ...

@serving
def ask(question: str) -> str:
    ...
    chain = get_chain()
    return chain.run(question=question)
```

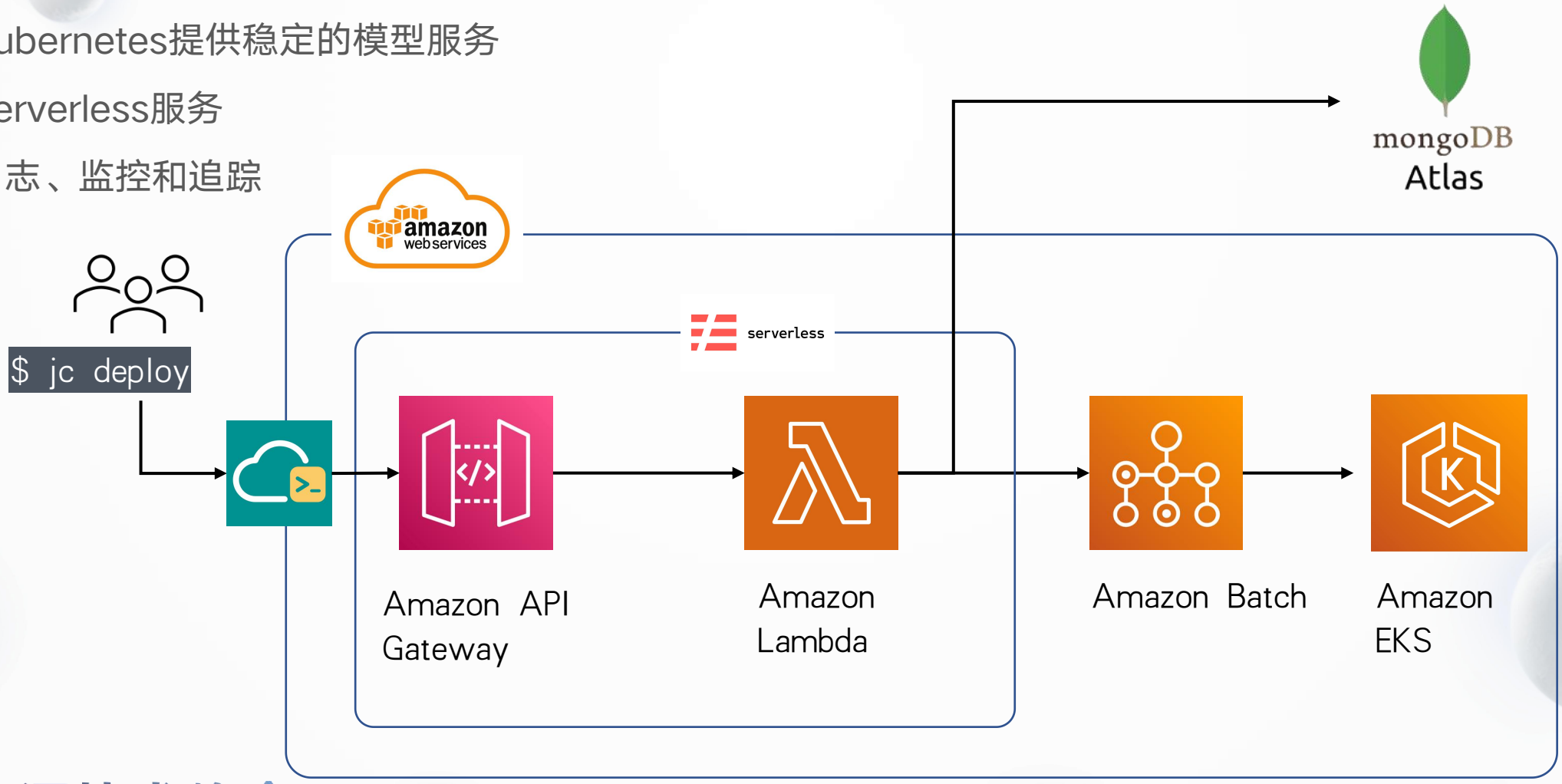
```
# start local flow
nan@gotc:~$ lc-serve deploy local app

# deploy on jina ai cloud
nan@gotc:~$ lc-serve deploy jcloud app
```

工程解决方案

工程设计：Jina AI Cloud

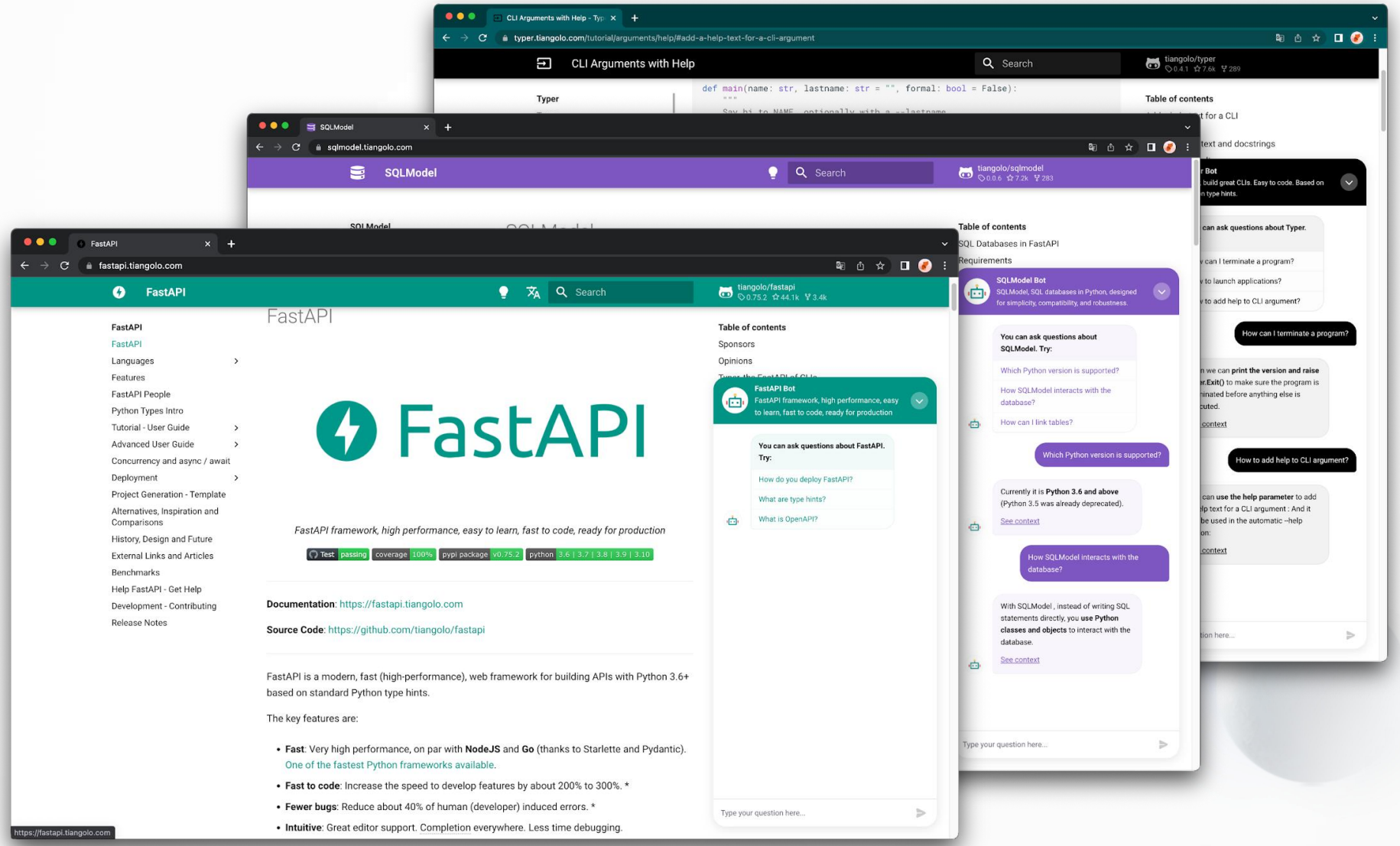
- 基于Kubernetes提供稳定的模型服务
- 提供Serverless服务
- 提供日志、监控和追踪



总结

实现效果

- 运营成本低
- 代码量小，可维护性高
- K8s部署，高可用性
- 代码开源



THANKS

2分钟填写反馈问卷

领取周边礼物

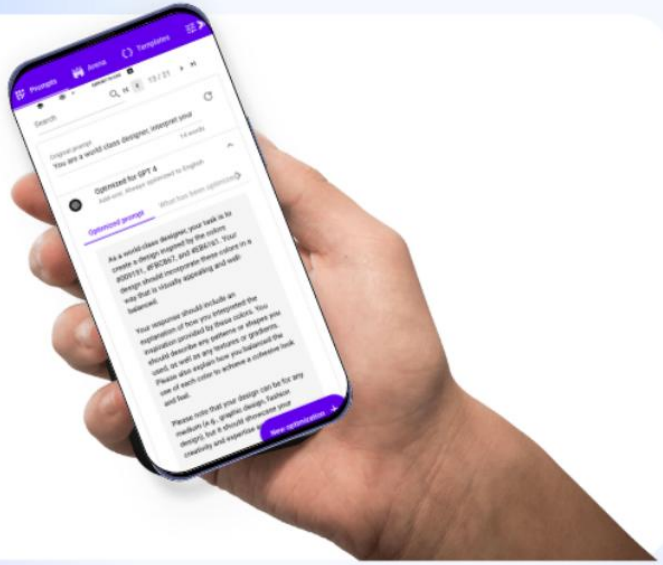
全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

优化提示词解决方案：PromptPerfect

第一个提示词自动优化产品

自动为任何大模型模型优化提示词。



支持 10+ 大模型

GPT-4, ChatGPT, StableLM, Claude, Command, GPT3 MidJourney, DALL-E, StableDiffusion, Kandinsky, Lexica

多目标优化

根据需求，自定义选择提示次优化方向，比如更快的优化，绕开道德过滤器，更短的提示词

API 调用

无缝集成到你的开发应用中。

八折优惠码：0528SH