AI 大模型开发工程师
之基于CVP架构的本地知识库

讲师：李希沅

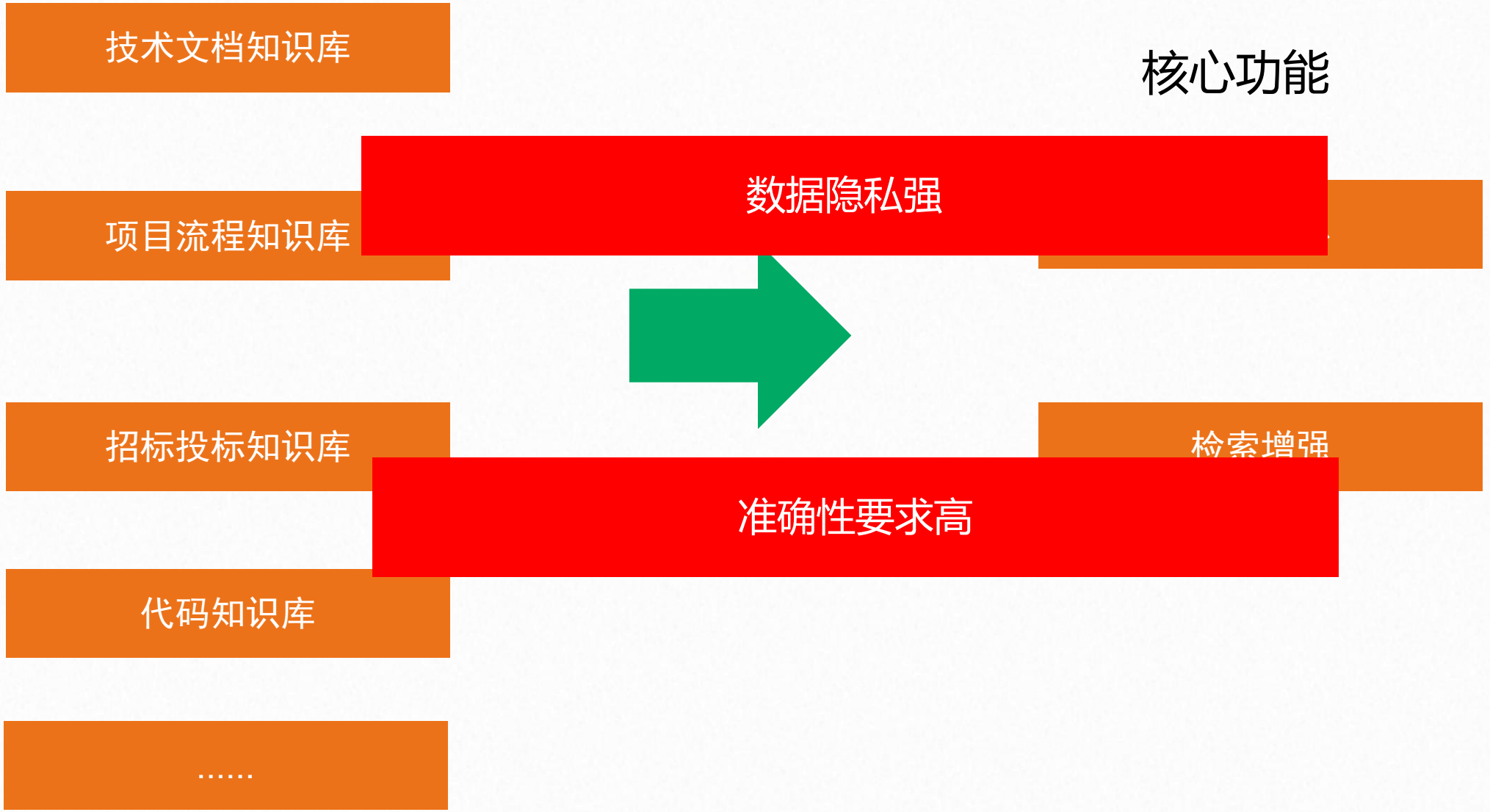
目录

1 基于CVP架构的本地知识库需求分析

01、CVP架构模式

ChatGPT + VectorDB + Prompt 架构模式（RAG）

02、知识库场景分析



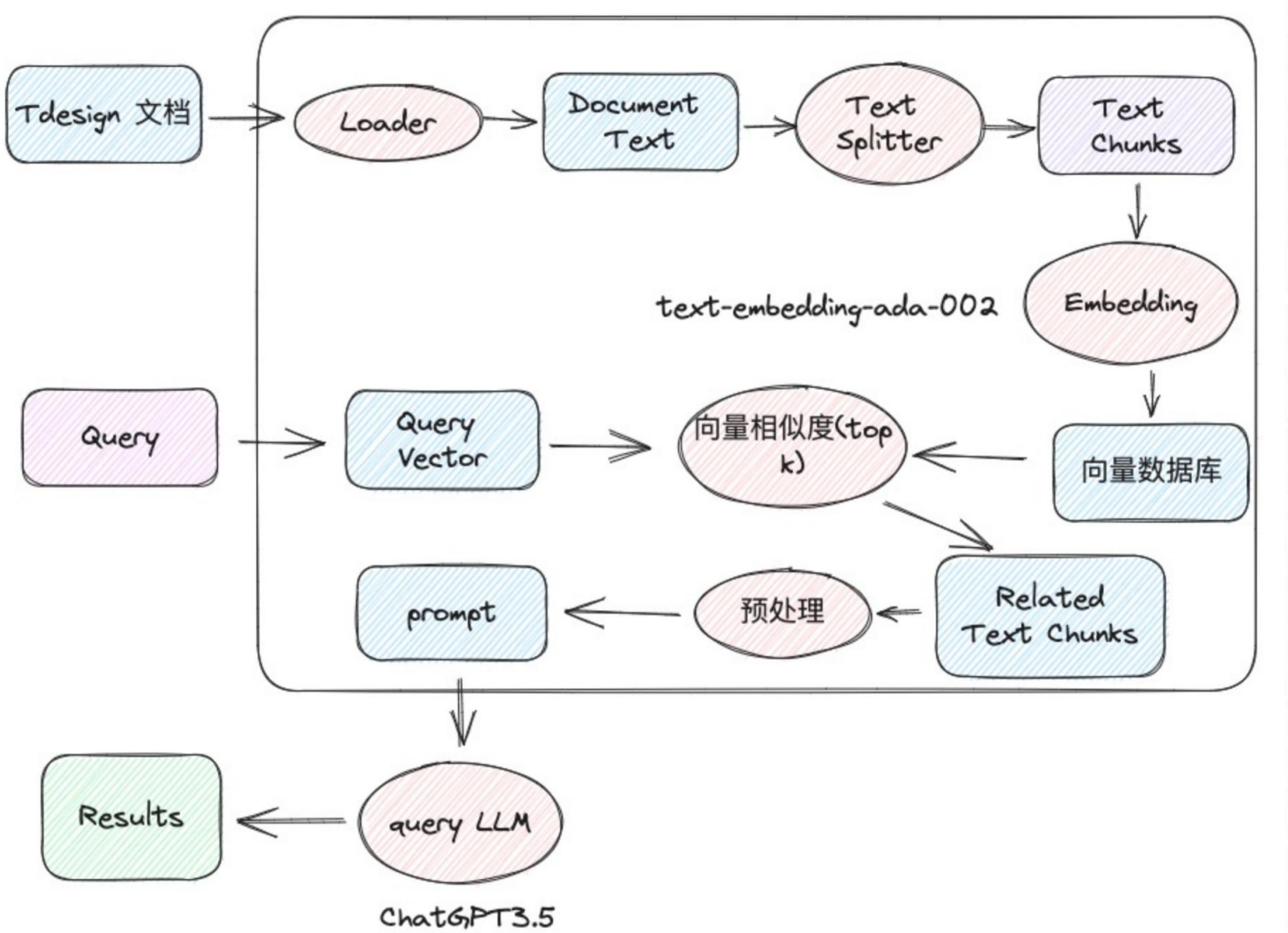
2 基于CVP架构的本地知识库架构设计

01、RAG（retrieval augmented generation）

1、知识数据向量化

2、知识数据召回

3、查询返回结果



02、架构选型

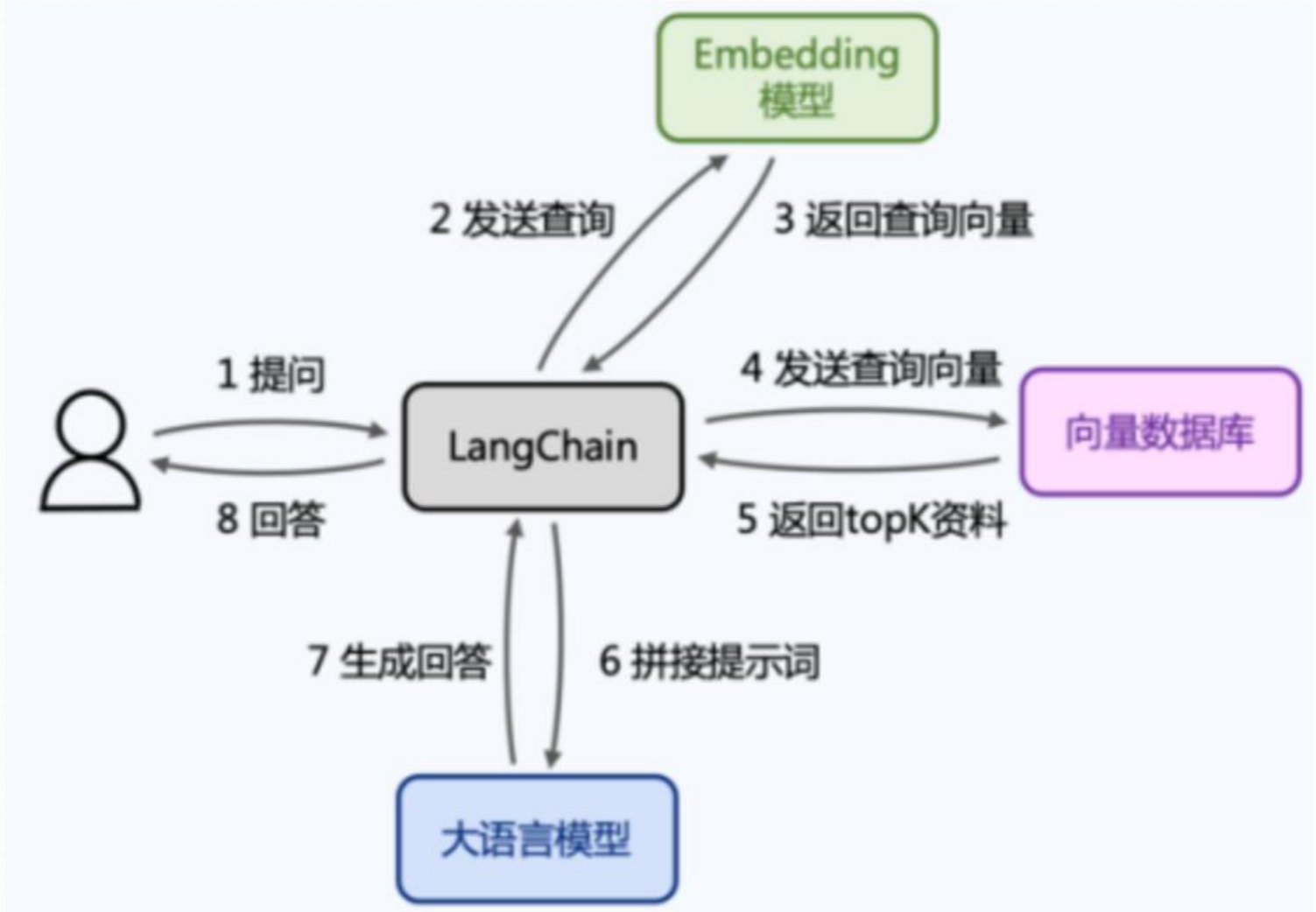
Fine-tuning（微调）VS Embedding（嵌入）VS GPTs

	微调	Embedding	GPTs
模型选型	开源LLM：GLM，Llama等	一般是闭源的模型	OpenAI
方法	通过微调更新知识	通过迭代向量数据库	通过更新Prompt
调试			短
数据类型			方服务，通用的
开发门槛	需要GPU算力支持	基于API调用	Assistants API
技术门槛	较深的AI算法支持储备	相对较少的AI知识	较少的AI的知识

微调+Embedding

03、技术选型

LLM + VectorDB + Prompt + LangChain 应用架构模式设计



开发框架选型

向量数据库选型

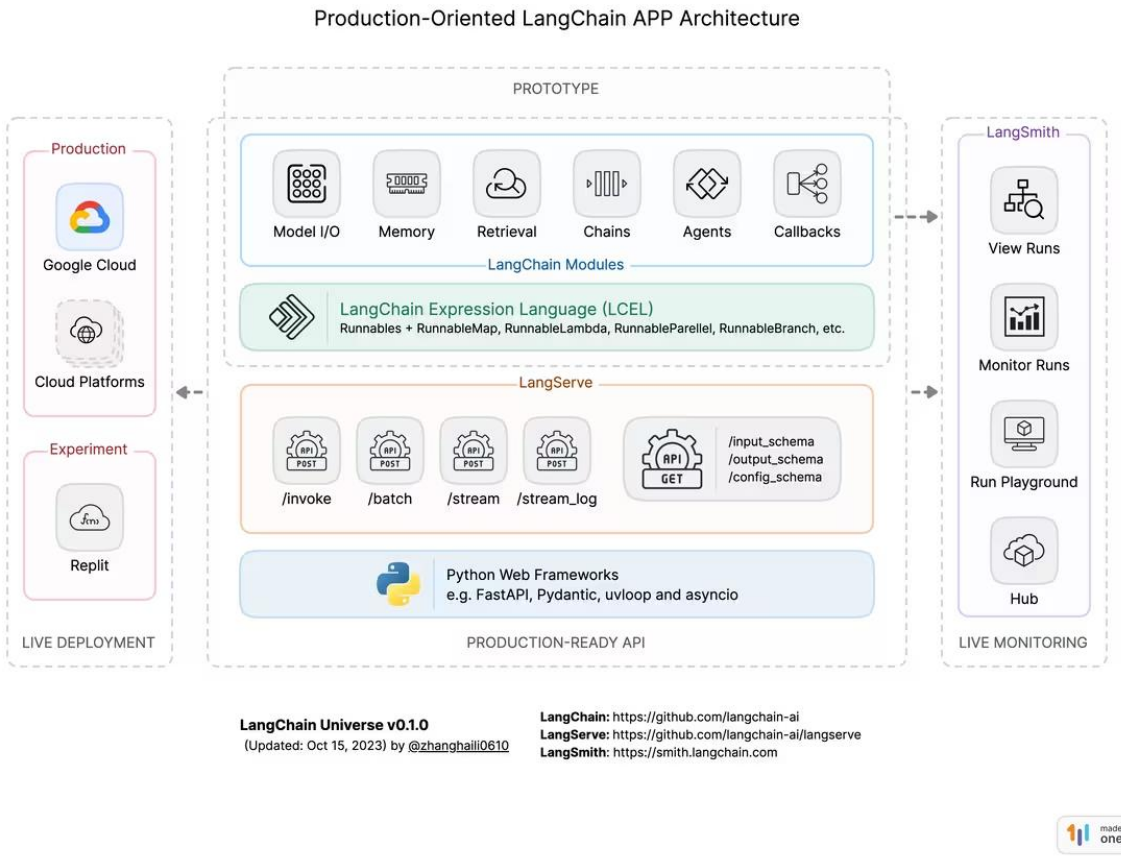
嵌入模型选型

大模型选型

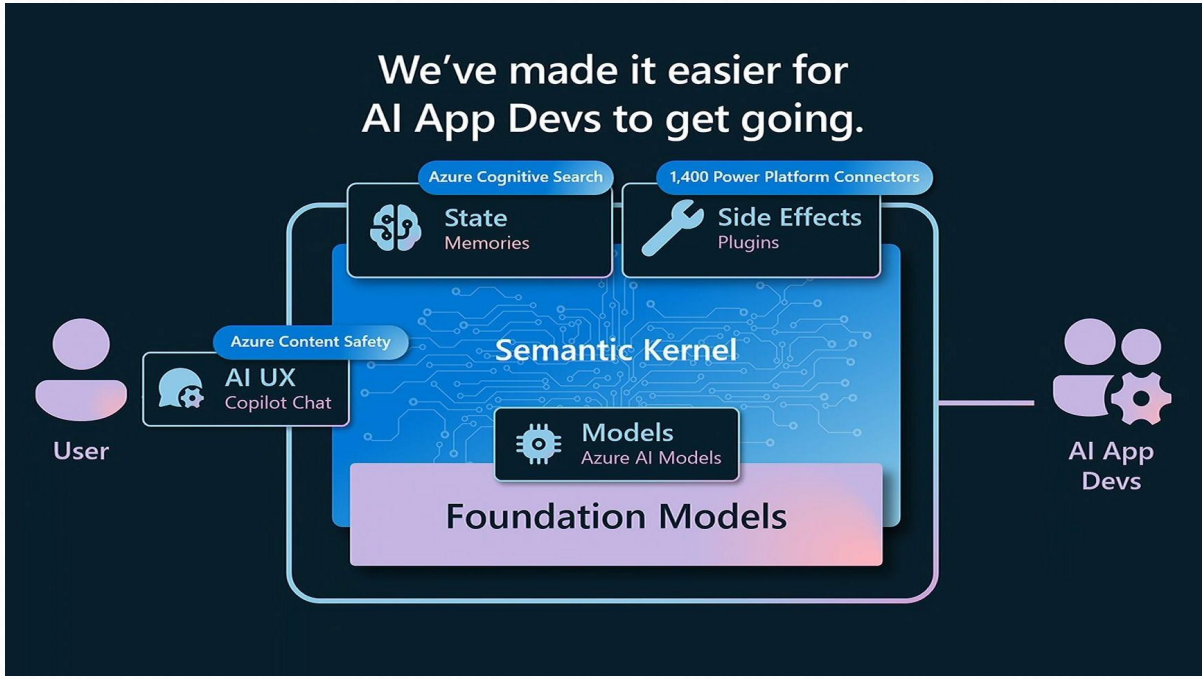
前端技术选型

04、开发框架选型

LangChain



Semantic Kernel

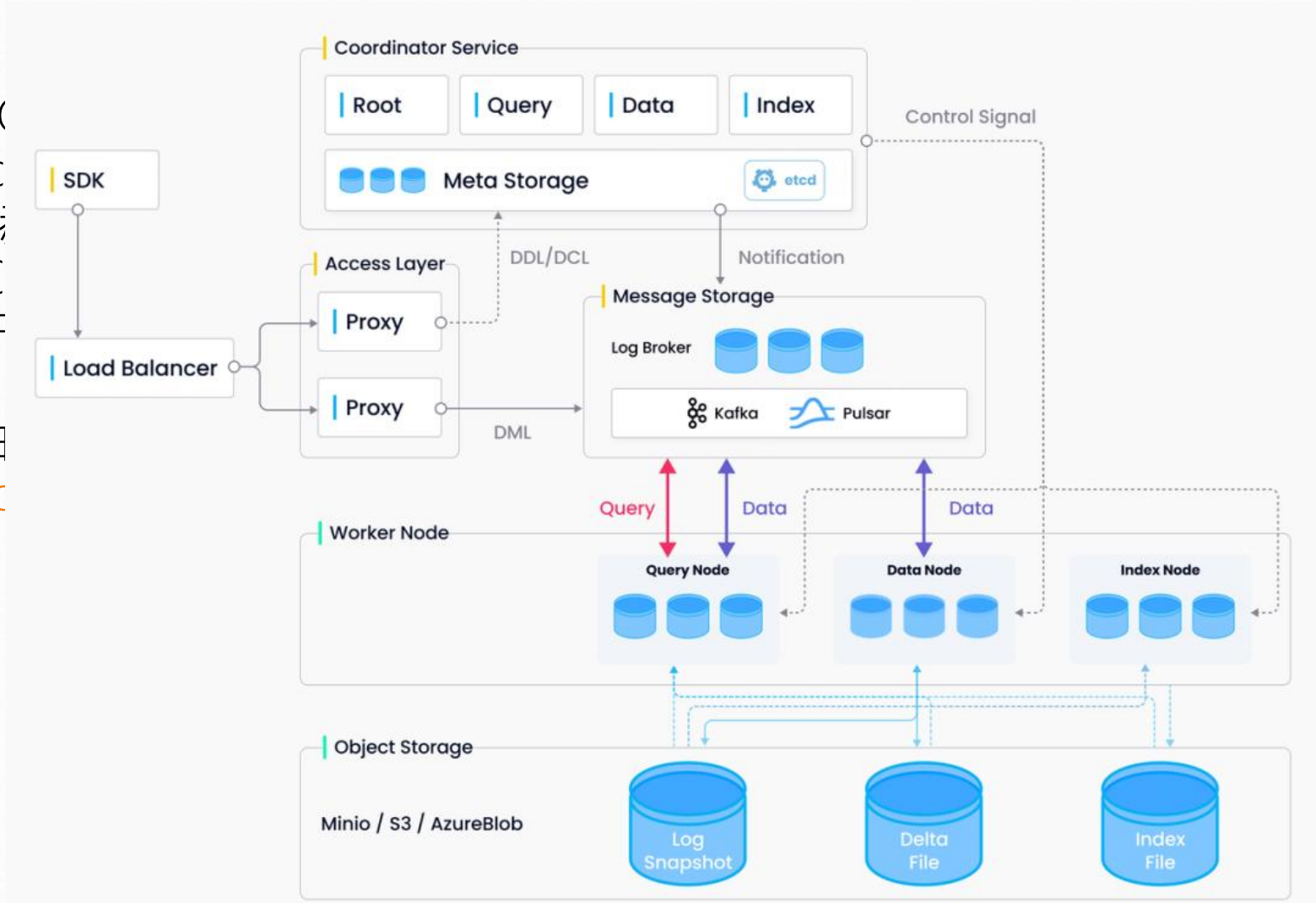


Semantic Kernel（SK）是一个轻量级的 SDK，可以让传统编程语言（目前仅支持 C# 和 Python）与 AI 大语言模型集成

05、向量数据库选型

1、向量数据库选型

- 国产基于 PostgreS
  - PgVector、Clic
- 基于传统倒排搜索
  - Elasticsearch C
- 基于向量检索库（F
  - Chroma
- 基于原生向量设计
  - Milvus、Pinec



06、嵌入模型选型

1、Embedding Model（嵌入模型）

- 文本向量化模型
  - OpenAI 的 text-embedding-ada-002 / m3e-base（支持768维）
- 图像向量化模型
  - clip-vit-base-patch32
- 音频向量化模型
  - wav2vec2-base-960h

2、文本向量化模型实例

- Prompt：“Your text string goes here”
- 使用 OpenAI text-embedding-ada-002 模型文本 Embedding，生成一个 1536 维向量，长度为 1536 维的数组，如下：
- “ -0.006929283495992422, -0.005336422007530928, ... -4547132266452536e-05,-0.024047505110502243 ”

07、模型选型

模型的选择

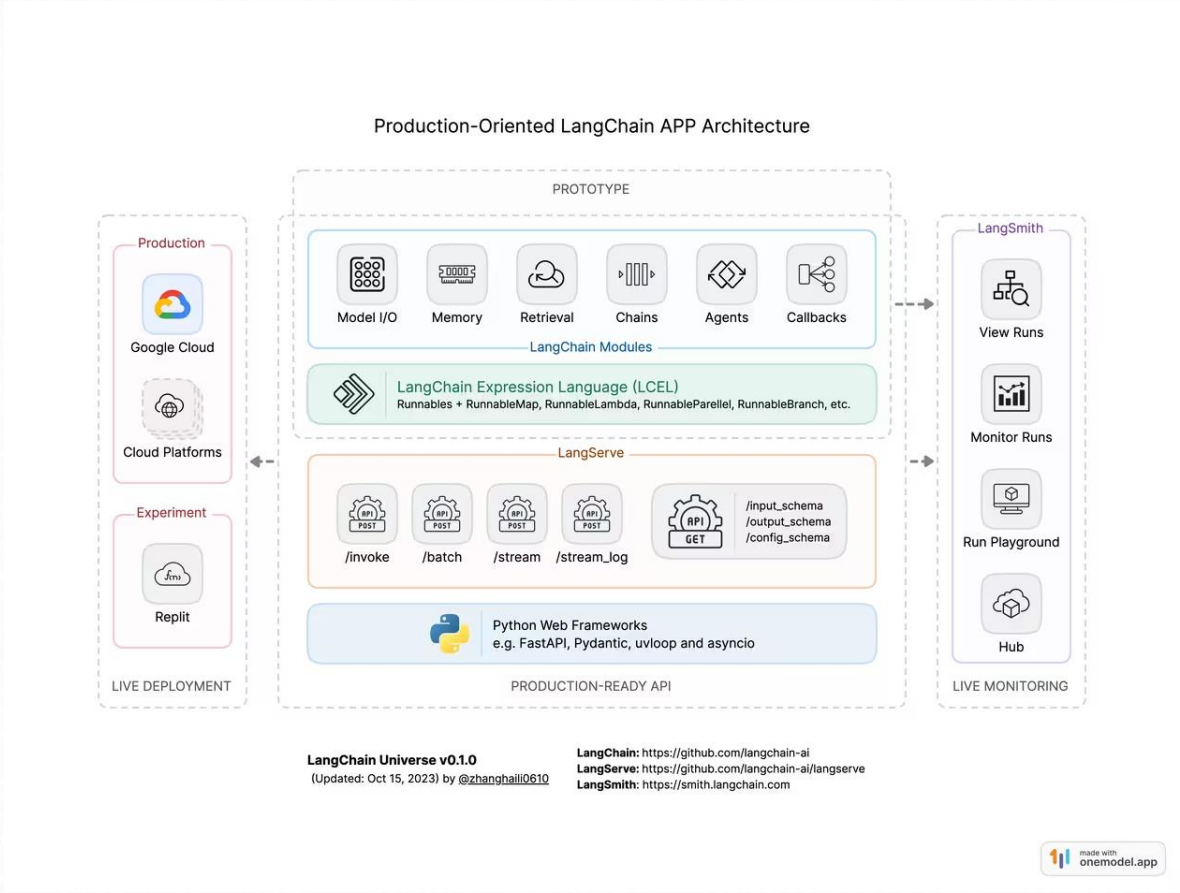
Model	MMLU	C-Eval	GSM8K	MATH	HumanEval	MBPP	BBH	CMMLU
	5-shot	5-shot	8-shot	4-shot	0-shot	3-shot	3-shot	5-shot
LLaMA2-7B	46.8	32.5	16.7	3.3	12.8	20.8	38.2	31.8
LLaMA2-13B	55.0	41.4	29.6	5.0	18.9	30.3	45.6	38.4
LLaMA2-34B	62.6	-	42.2	6.2	22.6	33.0	44.1	-
ChatGLM2-6B	47.9	51.7	32.4	6.5	-	-	33.7	-
InternLM-7B	51.0	53.4	31.2	6.3	10.4	14.0	37.0	51.8
InternLM-20B	62.1	58.8	52.6	7.9	25.6	35.6	52.5	59.0
Baichuan2-7B	54.7	56.3	24.6	5.6	18.3	24.2	41.6	57.1
Baichuan2-13B	59.5	59.0	52.8	10.1	17.1	30.2	49.0	62.0
Qwen-7B (original)	56.7	59.6	51.6	-	24.4	31.2	40.6	58.8
Qwen-7B	58.2	63.5	51.7	11.6	29.9	31.6	45.0	62.2
Qwen-14B	**66.3**	**72.1**	**61.3**	**24.8**	**32.3**	**40.8**	**53.4**	**71.0**

上下文序列长度

Model	Sequence Length					
	1024	2048	4096	8192	16384	32768
Qwen-7B (original)	4.23	3.78	39.35	469.81	2645.09	-
+ dynamic_ntk	4.23	3.78	3.59	3.66	5.71	-
+ dynamic_ntk + logn	4.23	3.78	3.58	3.56	4.62	-
+ dynamic_ntk + logn + window_attn	4.23	3.78	3.58	3.49	4.32	-
Qwen-7B	**4.23**	**3.81**	**3.52**	**3.31**	7.27	181.49
+ dynamic_ntk + logn + window_attn	**4.23**	**3.81**	**3.52**	**3.33**	**3.22**	**3.17**
Qwen-14B	-	**3.46**	22.79	334.65	3168.35	-
+ dynamic_ntk + logn + window_attn	-	**3.46**	**3.29**	**3.18**	3.42	-

08、前端框架选择

RESTful接口



gradio

streamlit

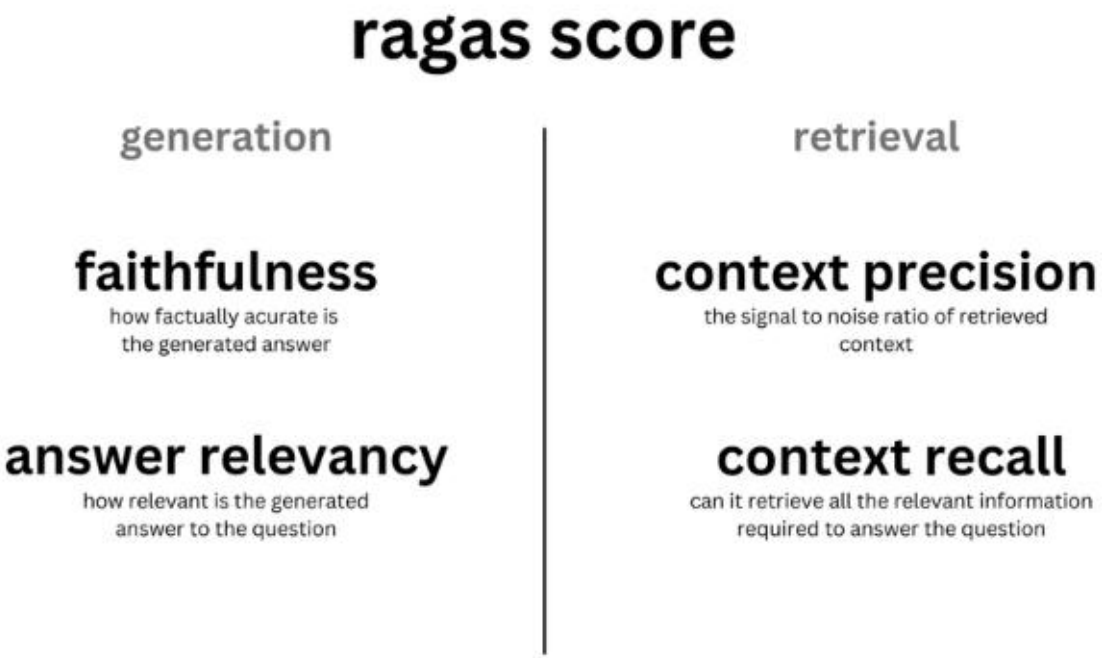
3 基于CVP架构的RAG评估

01、智能评估

https://github.com/ explodinggradients/ragas

Ragas是一个框架，帮助您**评估检索增强生成（RAG）流程**。RAG表示一类LLM应用程序，它使用外部数据来增强LLM的上下文。有现有的工具和框架可帮助您构建这些流程，但评估和量化流程性能可能很困难。这就是Ragas（RAG评估）发挥作用的地方

02、评估维度



Faithfulness：衡量的是指回答与问题基于上下文的事实一致性，返回值【0-1】范围，越高越好。

context precision：衡量检索到的上下文与问题的相关性。传达了检索流程的质量

answer relevancy：衡量回答与问题的相关性【0-1】，越高越好

Context Recall：上下文召回率

03、评估代码

```
from langchain.document_loaders import TextLoader
from langchain.indexes import VectorstoreIndexCreator
from langchain.chains import RetrievalQA
from langchain.chat_models import ChatOpenAI

loader = TextLoader("./nyc_wikipedia/nyc_text.txt")
index = VectorstoreIndexCreator().from_loaders([loader])


llm = ChatOpenAI(temperature=0)
qa_chain = RetrievalQA.from_chain_type(
    llm,
    retriever=index.vectorstore.as_retriever(),
    return_source_documents=True,
)
```

```
eval_questions = [
    "What is the population of New York City as of 2020?",
    "Which borough of New York City has the highest population?",
    "What is the economic significance of New York City?",
    "How did New York City get its name?",
    "What is the significance of the Statue of Liberty in New York City?",
]

eval_answers = [
    "8,804,190",
    "Brooklyn",
    "New York City's economic significance is vast, as it serves as the global
    "New York City got its name when it came under British control in 1664. Kin
    "The Statue of Liberty in New York City holds great significance as a symbo
]

examples = [
    {"query": q, "ground_truths": [eval_answers[i]]}
    for i, q in enumerate(eval_questions)
]
```

```
from ragas.langchain.evalchain import RagasEvaluatorChain
from ragas.metrics import (
    faithfulness,
    answer_relevancy,
    context_precision,
    context_recall,
)

# create evaluation chains
faithfulness_chain = RagasEvaluatorChain(metric=faithfulness)
answer_rel_chain = RagasEvaluatorChain(metric=answer_relevancy)
context_rel_chain = RagasEvaluatorChain(metric=context_precision)
context_recall_chain = RagasEvaluatorChain(metric=context_recall)
```

Faithfulness

```
eval_result = faithfulness_chain(result)
eval_result["faithfulness_score"]
```

```
0.8
```

03、案例演示

评估流程代码演示

4 基于CVP架构的本地知识库代码落地

01、核心要点考虑

要点1：核心参数设置

- 大模型温度参数设置为0
- 系统提示词

要点2：

问题分析	解决方案
知识切片不连贯导致的上下文信息丢失	提升Embedding质量，切片附加上下文，用其他工程手段优化
多维度知识匹配能力有限，context 长度有限，会造成多维度知识检索的能力偏弱	chunkSize优化，大模型增加上下文，增加topK的召回数量
用户输入的问题，在向量数据库中关联不上	处理用户输入，标准化

要点3：

清晰且语义化的数据结构

02、代码实战

代码演示

5 基于CVP架构的本地知识库项目上线以及运维

01、项目上线及运维

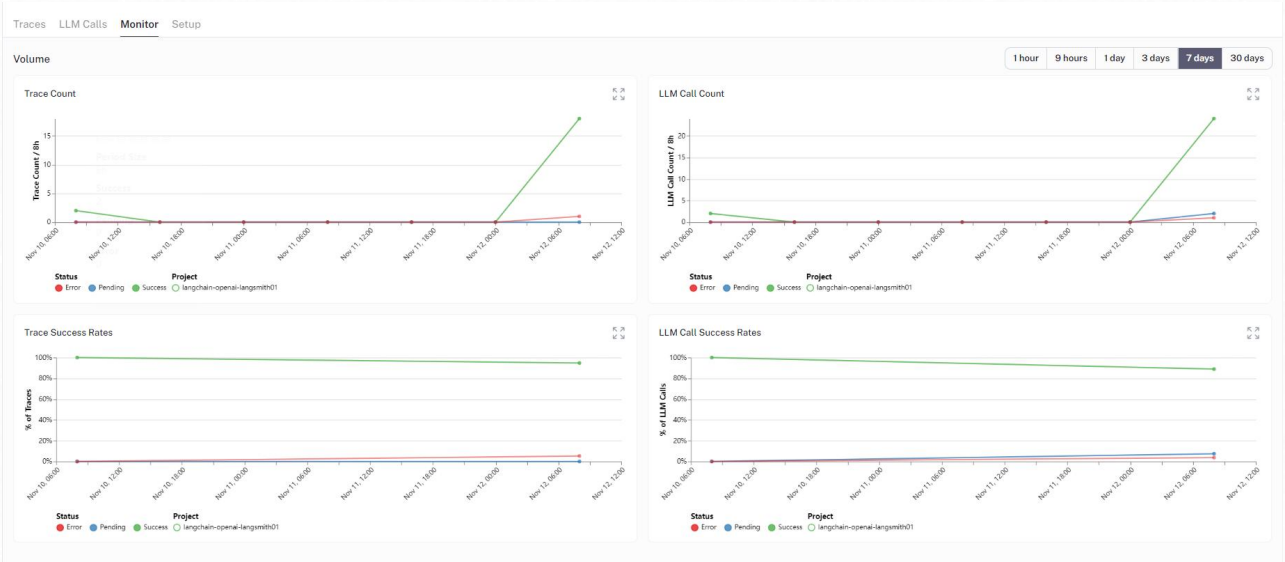
LangChain Server 1.0 OAS 3.1

/openapi.json

A simple api server using Langchain's Runnable interfaces

default

- POST /get_knowledge/invoke Invoke
- POST /get_knowledge/batch Batch
- POST /get_knowledge/stream Stream
- POST /get_knowledge/stream_log Stream Log
- GET /get_knowledge/input_schema Input Schema
- GET /get_knowledge/output_schema Output Schema
- GET /get_knowledge/config_schema Config Schema



02、项目总结

加入微调框架

数据格式测试评估

模型的更换评估

切片大小测试评估

模型的大小评估

检索算法测试评估

谢谢观看