

百度智能云千帆SDK实战

LLMOps SDK 构建持续成长的AI原生应用

HAN

2023.12.1

目录

01 百度智能云千帆平台与SDK

02 SDK架构和使用

03 SDK进阶最佳实践

百度智能云千帆平台与SDK

百度智能云千帆大模型平台

一站式企业级大模型平台，提供先进的生成式AI生产及应用全流程开发工具链

千帆AI原生应用工作台

千帆大模型平台



BML· AI开发平台

百度百舸· AI异构计算平台

50+个

预置模型

一站式

大模型开发及服务运行平台

支持文心大模型

支持在线优化调优

支持数据集管理

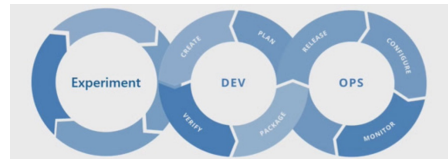
支持自定义开源大模型

百度智能云千帆SDK助力高效进行大模型开发应用

 python

 LangChain


Agent


LMOps

 jupyter  GitHub

更熟悉的开发方式

本地/云IDE

命令行使用

更灵活的组合使用

自由串联应用

LMOps迭代模型

更低的接入成本

开源工程代码

海量Cookbook

百度智能云千帆SDK的功能概览



大模型预测能力

- Chat 对话
- Completion 续写
- Embedding 向量化
- Plugin 插件调用
- 文生图



大模型调优

- 创建训练任务
- 创建任务运行
- 获取任务运行详情
- 停止任务运行
- 训练任务发布为模型



数据管理

- 创建数据集
- 发起数据集发布任务
- 发起数据集导入任务
- 数据清洗
- 数据标注



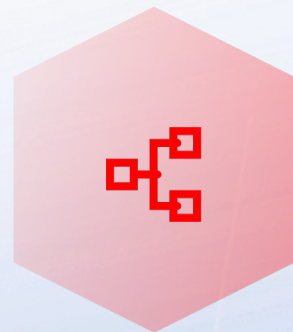
模型服务部署

- 创建服务
- 查询服务详情
- 获取模型详情
- 获取模型版本详情



Dataset

- 数据集加载
- 数据集导出
- 数据实体操作
- 数据处理, 清洗



Trainer

- 基础Action
- Pipeline定制流程
- Action事件回调
- Fine-tune串联

SDK架构和使用

百度智能云千帆SDK

架构简介

Qianfan SDK

Dataset

Format

Text, Jsonl, CSV, Zip

Source

Qianfan, LocalFile

Operator

Up/Down、Process

Trainer

Finetune

Lora、PTuning、SFT、RLHF

Pretrain

PreTrain , PostPretrain

Pipeline

Action-based pipeline

Extensions

Agent

Component

Client

Hub

Resource

Console/Finetune

Console/Model

Console/Data

llm/Chat

llm/Embedding

llm/Completion

Text2Image

Plugin

数据集

多数据源接入

本地、千帆平台

高度灵活使用

数据集CRUD、数据清洗、数据标注

模型训练

自动化训练任务

数据切分、精调训练、模型发布

全面监控训练过程

应用工程

Agent框架

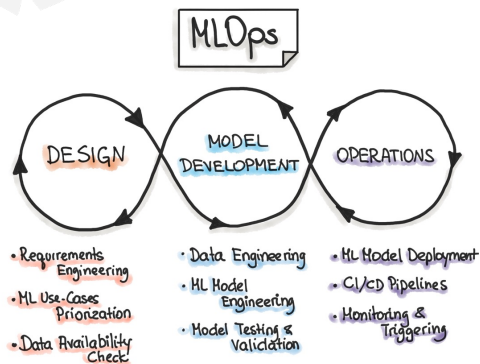
Tools灵活接入

模型开发：LMOps

自动化数据处理、模型训练以及监控

LMOps

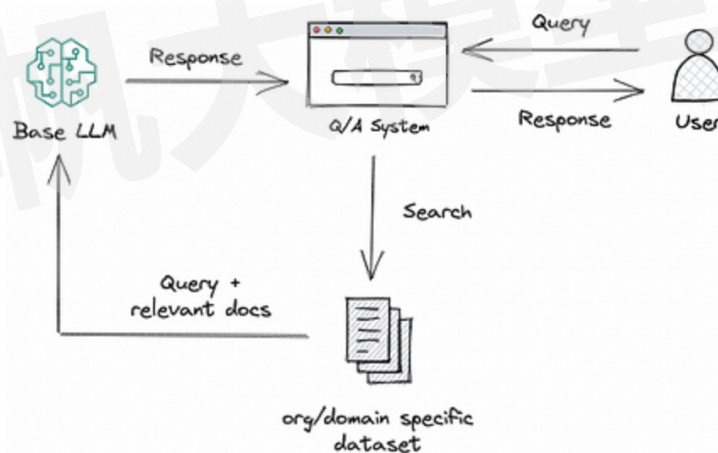
数据	模型	训练
无监督学习 ↑大幅上升	代码修改 ↓几乎不需要	分布式并行 ↑大幅上升
调优	调优	服务
提示工程 ↑新出现	SFT/LoRA ↑新出现	插件拓展 ↑新出现
服务	部署	评估
多模态生成 ↑大幅上升	基础+增量 ↑新出现	模型做裁判 ↑新出现



业务应用：RAG应用开发/Agent

使用千帆SDK+langchain

实现检索知识增强应用



关于千帆SDK安装和使用

目前千帆 SDK 已发布到 PyPI，同学们可使用 pip 命令进行安装。安装千帆 SDK 需要 3.7.0 或更高的 Python 版本

qianfan 0.2.0

`pip install qianfan` 



Released: Nov 23, 2023

SDK基本使用流程:

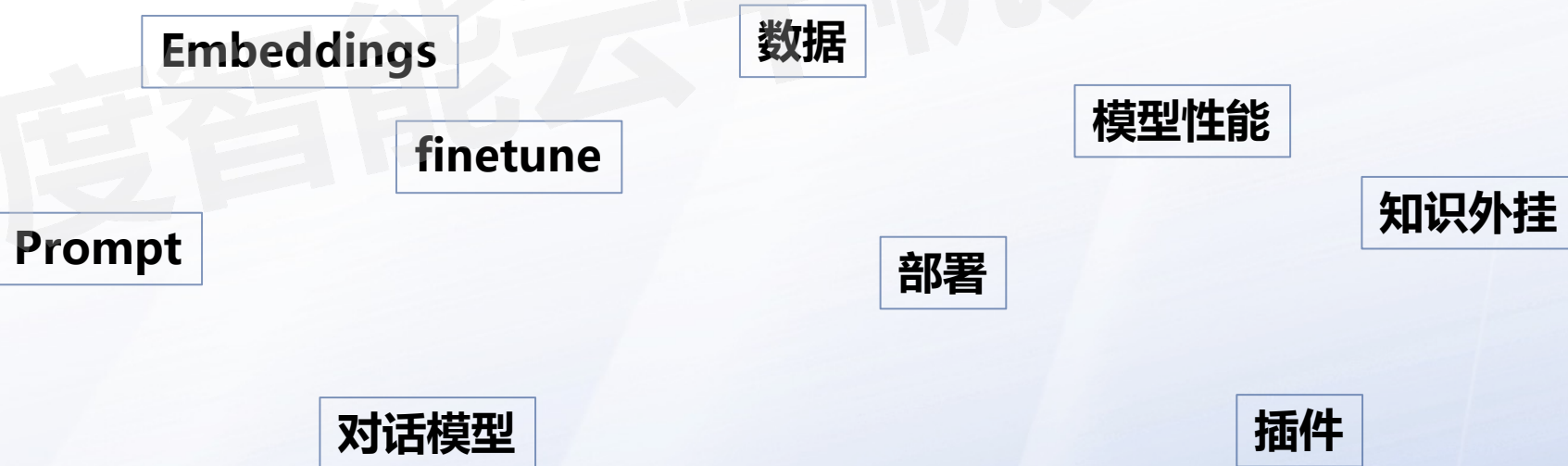


大模型应用示例

思考一个专业智能客服问答的场景：

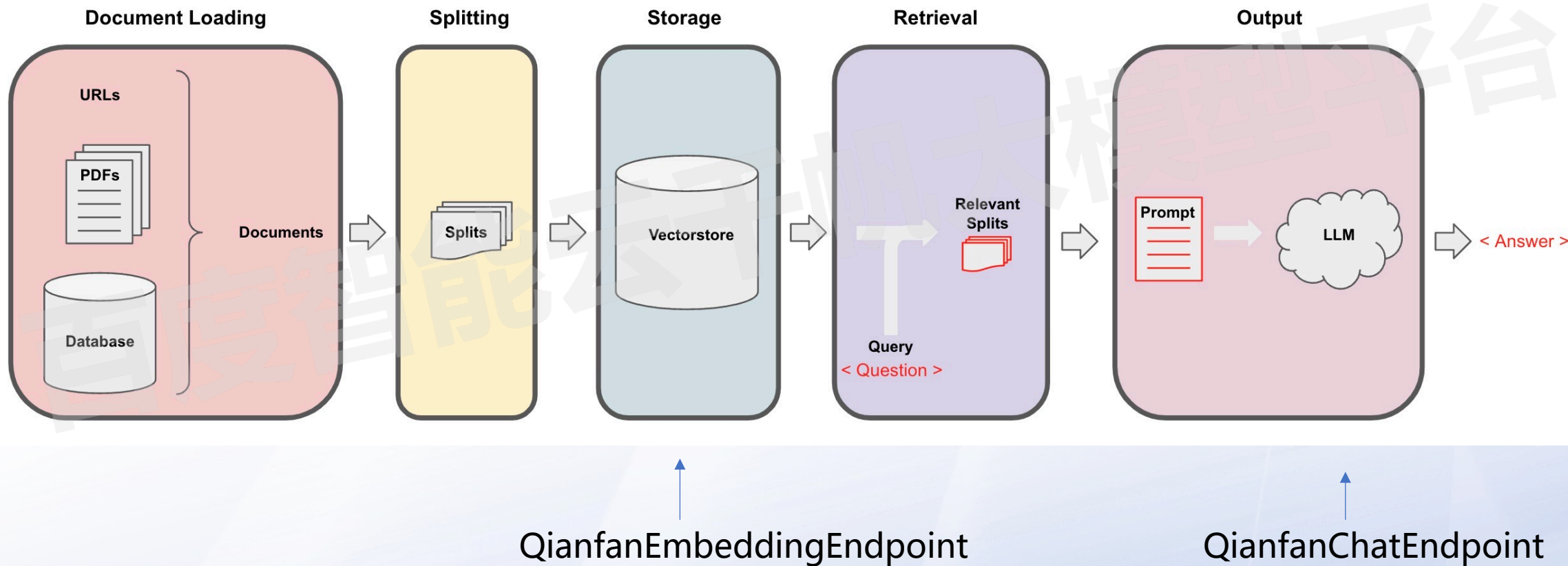
我们需要给应用增加多轮对话、专业的知识问答能力、并希望客服能够按照我们希望的格式进行回答。

这样的情况下，我们需要使用到哪些大模型应用开发需要关注的元素？



结合Langchain实现RAG

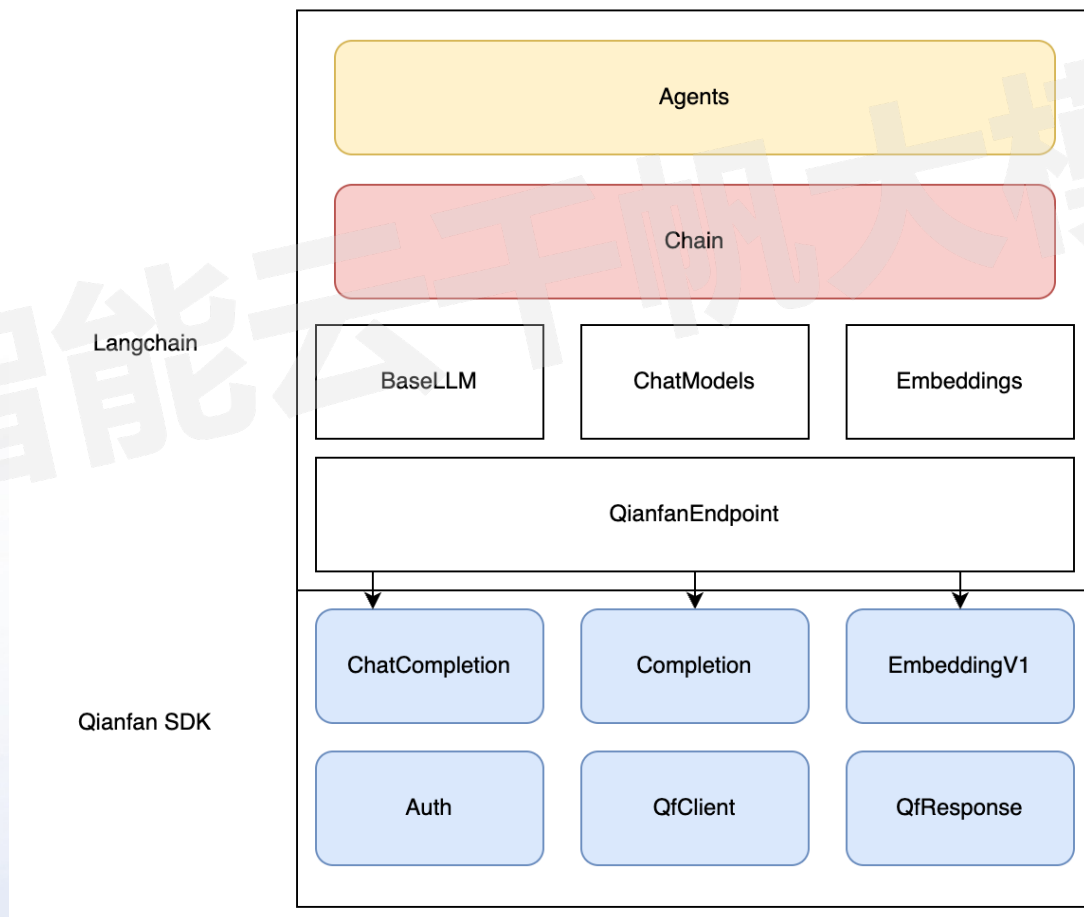
以下是一个使用qianfan、Langchain实现RAG-QA的例子



https://github.com/baidubce/bce-qianfan-sdk/blob/main/cookbook/RAG/question_answering/question_answering.ipynb

结合Langchain

千帆SDK已集成到Langchain实现中，通过Langchain.chat_models/lms/embeddings可以快速的调用SDK的基础预测能力。



- 1. 模型风格的调整：**对于领域内涵的知识，需要适配用户提问的口吻甚至方言等风格，这样的情况无法单一使用RAG来实现，我们仍然需要调整模型以达成期待的效果。
- 2. 需要维护知识库：**对于领域的知识库，RAG系统能够提供灵活的知识外挂配置，但对于一些无法交付和管理知识库的场景，例如通用的对话QA应用，RAG也不太适合。
- 3. 模型能力不足：**对于基础模型的推理，总结能力不足的场景，RAG可能无法正常的发挥其提供知识补充的辅助能力。
- 4. 增加了语料召回的请求开销：**整体链路可能耗时相比于原生模型调用更长。

SDK进阶最佳实践

什么是LMOps?

『LMOps代表大模型运维生产管理，其缩写LMOps的意思是面向LM的MLOps，这意味着LMOps是用于管理LM驱动的应用程序生命周期(包括开发、部署和维护)的一组工具和最佳实践。』

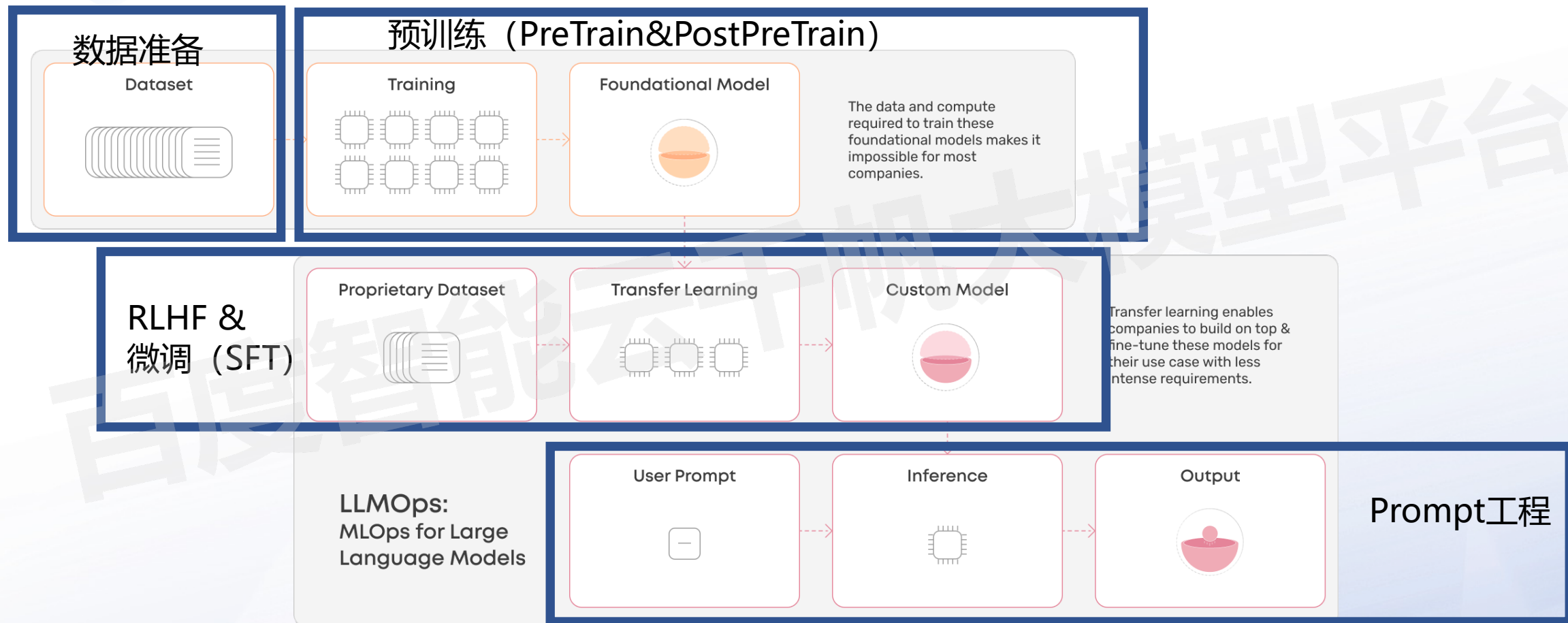
- **LM(大模型, Large model)** 是一种深度学习模型(具体来说例如语言文本类输出大模型可以归为大语言模型)。这类模型有数十, 上千亿个参数, 并在数十亿个单词上进行训练(因此被称为**大模型**)。

- **MLOps(机器学习运维, Machine Learning Operations)** 是一组用于管理机器学习模型生产运维的应用程序生命周期的工具和最佳实践。

总的来说, LMOps是一种特殊的MLOps, 针对于解决大模型场景下的例如Prompt Engineering、Finetune等模型应用调优手段特例化的一个子集。

LMOps与千帆SDK

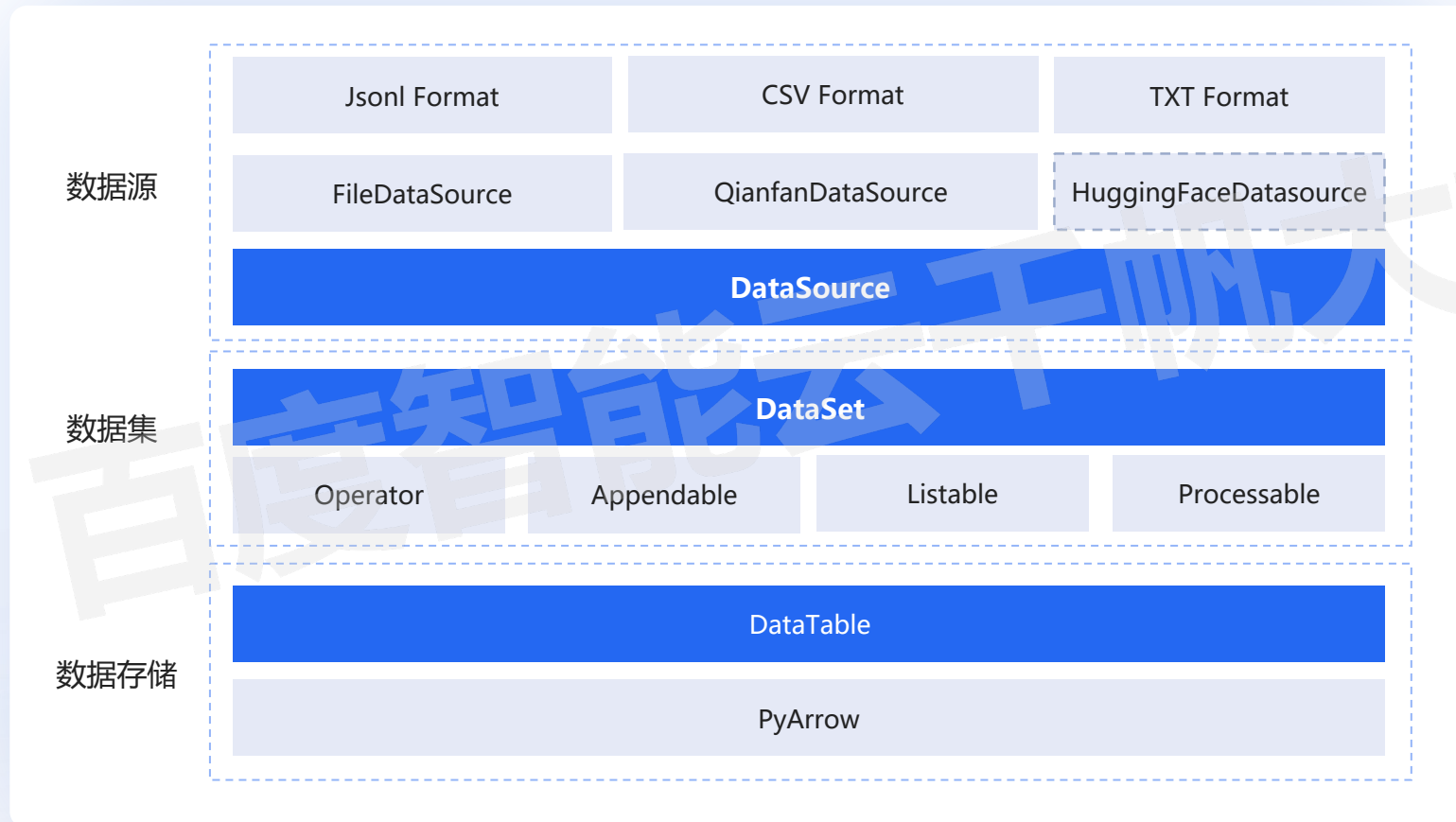
•LMOps的组成



千帆SDK一大目标就是为了简化LMOps这些生命周期的步骤，而应运而生的开发新工具和最佳实践。

千帆SDK Dataset

Dataset作为LMOps中重要的一环，我们做了如下设计和考虑，打通了多个数据源，可以方便的进行数据导入导出，并结合千帆平台能力预置了若干数据清洗和数据处理的能力。



多数据源

数据源支持了本地，文心千帆，BOS多种数据源。以及Jsonl，CSV，TXT等多种数据格式的数据元信息定义。

灵活处理

数据集支持了数据集的导入导出，数据增删改查，以及数据清洗，处理等操作。

快速存取

底层数据存储使用apache-arrow进行存储，并优化了嵌套数据的处理存取。

千帆SDK Dataset

我们以收集一个开源法律数据集为例，展开说明dataset的使用

- 数据导入/下载
- 数据导出/上传
- 数据清洗/处理
- 数据预览

```
from qianfan.dataset import Dataset

# 从本地文件导入
dataset = Dataset.load(data_file="path/to/dataset_file.jsonl")

# 从千帆导入
dataset = Dataset.load(qianfan_dataset_id=42)
```

```
from typing import Dict, Any

def filter_func(row: Dict[str, Any]) -> bool:
    # 编写你的过滤逻辑
    return "sensitive data for example" not in row["col1"]

def map_func(row: Dict[str, Any]) -> Dict[str, Any]:
    # 编写你的映射逻辑
    return {
        "col1": row["col1"].replace("sensitive data for example", ""),
        "col2": row["col2"]
    }

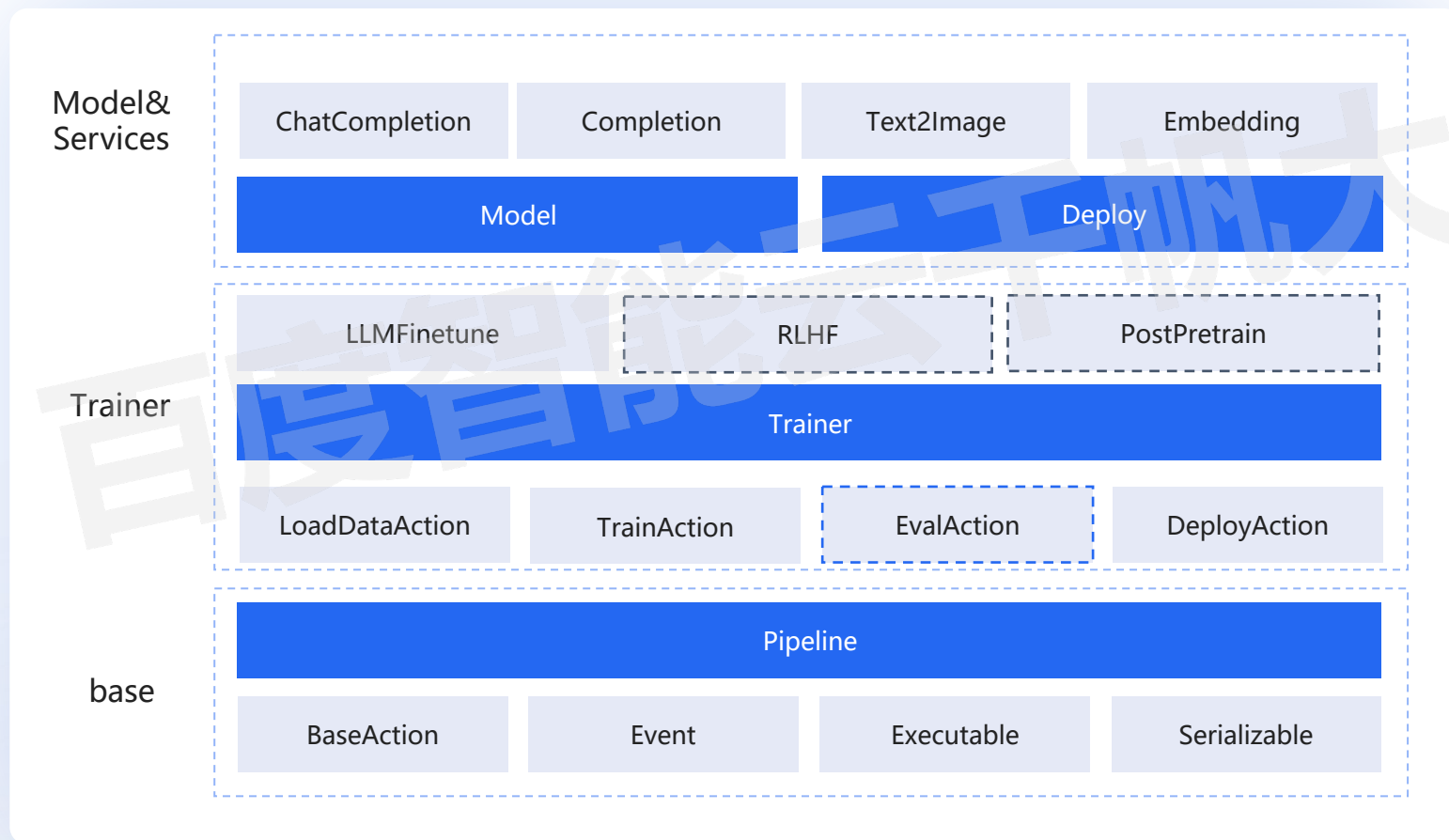
print(dataset.filter(filter_func).map(map_func).list())
```

https://github.com/baidubce/bce-qianfan-sdk/blob/dataset_readme/docs/dataset.md

千帆SDK Trainer

Training作为模型生产最费时费力的环境，大大的提高了大模型生产的门槛；也是决定应用效果的关键步骤之一。

千帆SDK围绕如何自动化训练流程提出了Trainer的概念，结合千帆平台提供了Fine-tune，定制训练Pipeline等模型训练能力。



快速部署

模型可以通过Model、Services快速部署调用以进行验证和更新。

自动化训练

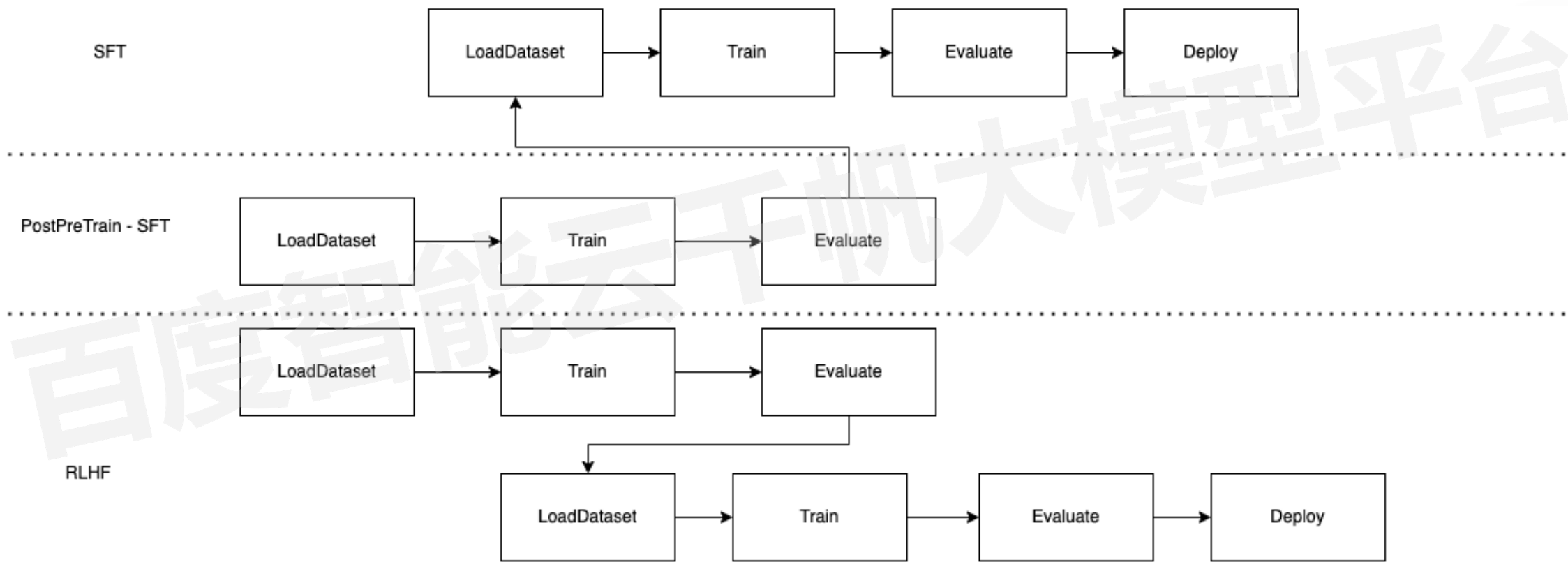
自动化串联千帆平台典型的三种训练任务整体流程，从数据准备、训练、到评估以及模型服务发布。

重点信息

基础的Action、Pipeline实现，支持持久化和事件状态回调帮助监控训练中的每一个运行节点

千帆SDK Trainer

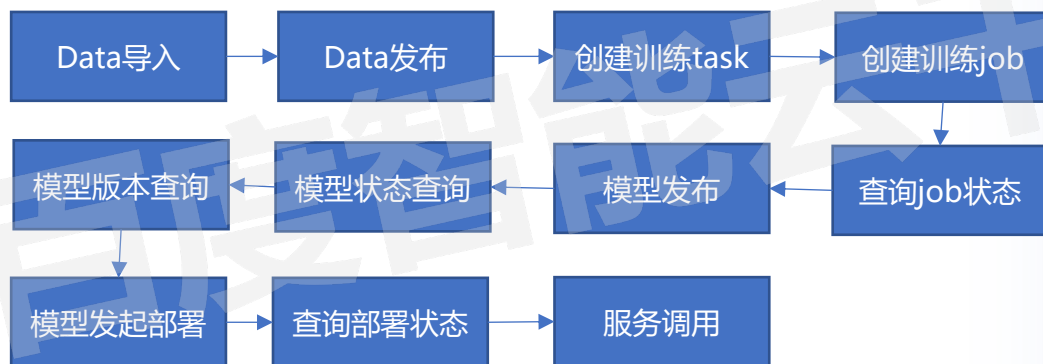
对于不同的训练任务默认实现的Pipeline:



对于相同从数据->SFT训练->部署->推理调用这一条任务pipeline，使用Trainer和原生API实现所需要的开发成本

使用原生API实现

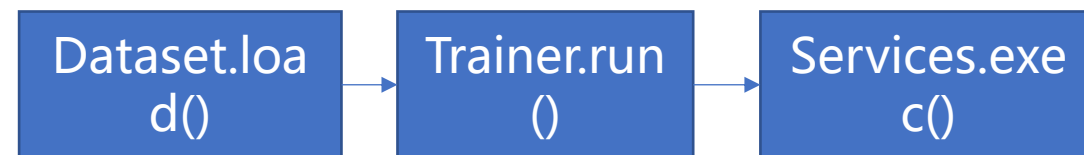
复杂的状态判断以及流程串联，中间任务状态监控，错误处理



<https://github.com/baidubce/bce-qianfan-sdk/blob/main/cookbook/finetune/console-finetune.ipynb>

使用SDK Trainer实现

- 一键式运行训练
- Pipeline定制组装
- 状态事件回调
- ...

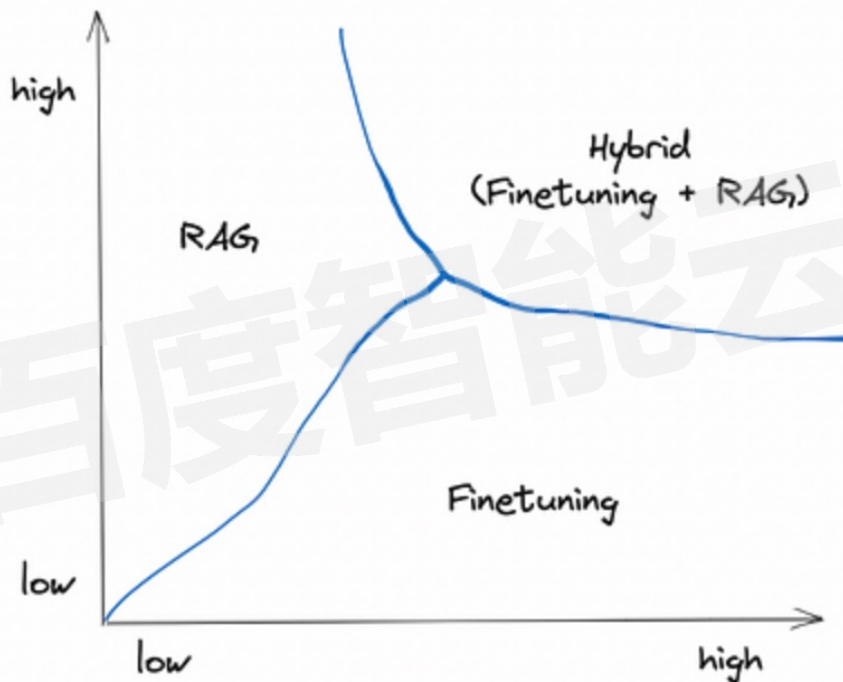


https://github.com/baidubce/bce-qianfan-sdk/blob/dataset_readme/docs/trainer.md

RAG vs SFT

外部知识库依赖

external knowledge
required



模型适配需求
model adaptation required
(e.g. behaviour/
writing style/
vocabulary)

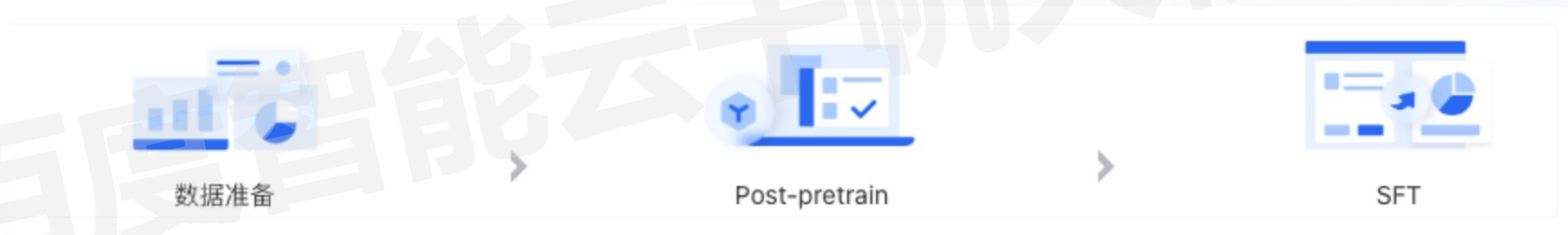
RAG和SFT，其实都是大模型应用实现或调优的手段。

在实际开发过程中应该根据大模型应用程序的需求和优先级进行细致的评估。

从成本上来说，不调整模型的肯定
会比需要微调更节省成本。
大部分情况下，我们可以结合
RAG+SFT将是最好的选择。

千帆SDK Trainer

Post-pretrain是一种预训练的模型训练方法。通常的应用场景是：
 在千帆大模型平台中，我们先对泛文本无标注数据进行预训练，通过PostPretrain得到一个强大的通用语言模型。
 然后，在此预训练模型的基础上进行SFT，调整部分参数后，得到一个更强大的模型。
 PostPretrain消耗的算力非常大，同时作业时间非常长，大部分场景下使用SFT即可。



从数据准备到PostPretrain，再用于SFT

RLHF (Reinforcement Learning from Human Feedback, 基于人类反馈的强化学习), 一种机器学习方法, 它使智能系统能够从环境中学习并最大化特定目标。具体来说可以分成两部分:

1. 在RLHF中, 通过对同一输入的多个生成结果进行人工排序, 获得包含人类偏好反馈的标注数据, 从而训练出一个奖励模型 (Reward Model)。
2. 在强化学习中, 奖励模型将对大语言模型的多个生成结果的排序进行判定。最终, 强化学习通过更新大模型的参数, 使得输出结果符合奖励模型的判定要求。



RLHF包含了奖励模型训练以及强化学习训练

智能体（Agent）是大模型应用中非常火热的话题，千帆SDK也基于Agent做了一系列的设计和实现。



https://github.com/baidubce/bce-qianfan-sdk/blob/main/cookbook/agents/qianfan_single_action_agent_example.ipynb

SDK Roadmap



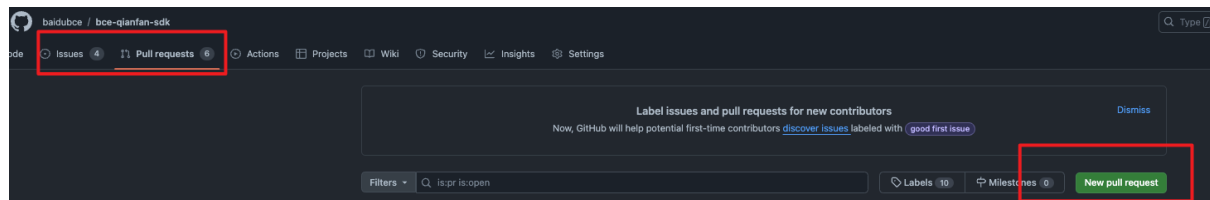
开源贡献

千帆SDK欢迎同学们一起参与到开源社区的建设

👉 关于如何贡献?

Github

<https://github.com/baidubce/bce-qianfan-sdk>



百度智能云千帆社区

<https://cloud.baidu.com/qianfandev>



社区发帖

👉 关于贡献内容

- Features: 希望得到支持的功能点
- Cookbooks: 和千帆平台有价值的案例notebook
- Bugs fix

在以上两个渠道均可进行相关反馈，反馈用户有机会获得百度智能云千帆大模型平台代金券，最高上限200元!

* 注: 百度智能云千帆相关工作人员会根据贡献内容来决定赠送代金券金额

课后作业

***将作业在【百度智能云千帆社区】进行发布，发布时选择 #大模型实训营 话题**

1. 使用千帆SDK实现一个开源模型预置服务实现终端对话机器人。尝试支持多模型选择。
2. 收集一份专业类型，例如金融，计算机的知识文本数据，通过Langchain/qianfan实现RAG。
3. 使用千帆SDK Dataset加载预置数据的数据集，使用Trainer运行SFT训练任务，训练自己的模型，并发布成服务，使用Service调用。



扫码进入
百度智能云千帆社区



添加小助手
进入课程群

T H A N K S

附录：

千帆平台API目前存在两种AK、SK，在使用上存在功能差异，我们推荐统一使用AccessKey作为SDK鉴权的初始化方式。与API对比

- **AccessKey: IAM安全认证AK、SK【推荐】**
- API Key: 千帆应用AK、SK

功能

API Key

Chat 对话



Completion 续写



Embedding 向量化



Plugin 插件调用



文生图



大模型调优



大模型管理



大模型服务



数据集管理



Prompt管理



AccessKey: IAM安全认证AK、SK

```
import os
import qianfan

os.environ["QIANFAN_ACCESS_KEY"]="..."
os.environ["QIANFAN_SECRET_KEY"]="..."
# 通过 App Id 选择使用的应用
# 该参数可选, 若不提供 SDK 会自动选择最新创建的应用
os.environ["QIANFAN_APPID"]="..."

chat_comp = qianfan.ChatCompletion(model="ERNIE-Bot")
resp = chat_comp.do(messages=[{
    "role": "user",
    "content": "你好, 千帆"
}], top_p=0.8, temperature=0.9, penalty_score=1.0)

print(resp["result"])
```

API Key: 千帆应用AK、SK

```
import os
import qianfan

os.environ["QIANFAN_AK"]="..."
os.environ["QIANFAN_SK"]="..."

chat_comp = qianfan.ChatCompletion(model="ERNIE-Bot")
resp = chat_comp.do(messages=[{
    "role": "user",
    "content": "你好, 千帆"
}], top_p=0.8, temperature=0.9, penalty_score=1.0)

print(resp["result"])
```

Completion模型服务

相比于对话的例子，Completion只需要提供一个prompt。

```
import qianfan
comp = qianfan.Completion()

resp = comp.do(model="ERNIE-Bot", prompt="你好")
# 输出: 你好! 有什么我可以帮助你的吗?

# 续写功能同样支持流式调用
resp = comp.do(model="ERNIE-Bot", prompt="你好", stream=True)
for r in resp:
    print(r['result'])

# 异步调用
resp = await comp.ado(model="ERNIE-Bot-turbo", prompt="你好")
print(resp['body']['result'])

# 异步流式调用
resp = await comp.ado(model="ERNIE-Bot-turbo", prompt="你好", stream=True)
async for r in resp:
    print(r['result'])

# 调用非平台内置的模型
resp = comp.do(endpoint="your_custom_endpoint", prompt="你好")
```

Embedding模型服务

Embedding是一种将文本转换成向量形式的技术，通过高维向量的表征，我们可以通过向量相似度比较实现文本检索，推荐等功能。

```
# Embedding 基础功能
import qianfan
emb = qianfan.Embedding()

resp = emb.do(model="Embedding-V1", texts=["世界上最高的山"])
print(resp['data'][0]['embedding'])
# 输出: 0.062249645590782166, 0.05107472464442253, 0.033479999750852585, ...]

# 异步调用
resp = await emb.ado(texts=[
    "世界上最高的山"
])
print(resp['data'][0]['embedding'])

# 使用非预置模型
resp = emb.do(endpoint="your_custom_endpoint", texts=[
    "世界上最高的山"
])
```

在前面的学习中我们已经学会了大模型的基础应用能力，除此之外，SDK也提供了千帆平台的插件能力。插件可以理解为大模型能力的增强和补充，特别是对于一些实时性的数据获取，以及可配置的知识库，我们的可以通过插件来实现。以下为使用知识库插件实现我们前面的需求的代码样例：

```
import os
import qianfan

os.environ["QIANFAN_ACCESS_KEY"]="your_ak"
os.environ["QIANFAN_SECRET_KEY"]="your_sk"

# Plugin 基础功能展示
plugin = qianfan.Plugin()
resp = plugin.do(endpoint="your_custom_endpoint", prompt="如何你在千帆平台创建数据集? ")
print(resp['result'])
```

针对于有多模态数据生成的需求，SDK也集成了千帆平台提供的文生图功能：

```
import os
os.environ["QIANFAN_ACCESS_KEY"]="your_ak"
os.environ["QIANFAN_SECRET_KEY"]="your_sk"

import qianfan
from PIL import Image
import io

t2i = qianfan.Text2Image()
resp = t2i.do(prompt="A Ragdoll cat with a bowtie.", with_decode="base64")
img_data = resp["body"]["data"][0]["image"]

img = Image.open(io.BytesIO(img_data))
display(img)
```

除了可以单次的跑预测，我们也针对大量请求需求的场景提供了批量推理这一功能：

```
prompt_list = ["你好", "很高兴认识你"]
# 所有模型传入的均为一个 List，但内部元素类型与模型类型相关
# 内部元素类型与 `do` 方法传入的参数类型一致
# 例如 ChatCompletion 则应该传入 List[Union[Dict, QfMessages]]
# 额外参数例如 `model` 等，与 `do` 方法可传入的参数和使用方法一致
task = Completion().batch_do(prompt_list, worker_num=10)
# 由于推理任务较为耗时，所以推理会在后台进行
# 返回的 task 是一个 Future 对象，可以通过它来获得任务运行状态
# 例如通过 resp.finished_count() 和 resp.total_count() 来获取已完成和总任务数
print("{} / {}".format(task.finished_count(), task.total_count())) # => 11/20
# SDK 会按照输入顺序进行批量推理
# 可以通过遍历的方式获取已完成任务的结果
for r in task:
    try:
        # 需要调用 r.result() 来显式等待某一条推理完成
        res = r.result()
        # 如果推理成功，res 是一个 QfResponse 对象
        # 否则会是一个 Exception 对象，用户需要进行错误处理
    except Exception as e:
        print(e)

# 也可以通过 task.results() 来等待所有推理任务完成并获取所有结果
# 但也同样需要进行错误处理
try:
    results = task.results()
except Exception as e:
    print(e)

# 或者仅等待所有任务完成，避免因返回的数据量较大导致过多的内存占用
# 之后可再采用上述遍历的方式逐个获取结果
task.wait()
# 结果与输入一一对应，结果与 `do` 返回类型一致
for prompt, result in zip(prompt_list, results):
    print(prompt, result)

# 异步调用
results = await Completion().abatch_do(prompt_list, worker_num=10)
# 返回值为一个 List，与输入列表中的元素一一对应
# 正常情况下与 `do` 返回类型一致，但如果发生异常则会是一个 Exception 对象
for prompt, result in zip(prompt_list, results):
    if not isinstance(result, Exception):
        print(prompt, result)
```

如何加载数据：

SDK支持了三种数据集源的导入，可以使用load()快速导入：

本地文件/文件目录数据集，千帆平台数据集，Huggingface数据集。

```
import os
os.environ["QIANFAN_ACCESS_KEY"] = "your_ak"
os.environ["QIANFAN_SECRET_KEY"] = "your_sk"
from qianfan.dataset import Dataset
from qianfan.trainer import LLMFinetune

qf_ds = Dataset.load(qianfan_dataset_id=36372, is_download_to_local=False)
local_ds = Dataset.load(data_file="./xx.jsonl")
hf_ds = Dataset.load(huggingface_name="squad")
```

如何导出数据：

SDK支持自定义数据源的导出、同时预置了本地，千帆平台，BOS（Baidu Object Service）等输出源。可以通过DataSource或直接使用Dataset.save()实现。

```
qf_ds.save(qianfan_dataset_id=36488)
```

有监督微调（SFT）是指采用预先训练好的神经网络模型，并针对你自己的专门任务在少量的监督数据上对其进行重新训练的技术。在千帆平台上已经预置了ERNIE-Bot系列大模型和BLOOM系列大模型。

```
create_job_resp = FineTune.create_job({
    "taskId": task_id,
    "baseTrainType": "ERNIE-Bot-turbo",
    "trainType": "ERNIE-Bot-turbo-0725",
    "trainMode": "SFT",
    "peftType": "LoRA",
    "trainConfig": {
        "epoch": 4,
        "learningRate": 0.00002,
        "batchSize": 4,
        "maxSeqLen": 4096
    },
    "trainset": [
        {
            "type": 1,
            "id": 1234
        }
    ],
    "trainsetRate": 20
})

# 获取运行 id
print(create_job_resp['result']['id'])
```

通过resources接口创建SFT训练任务

1.幻觉 LLM答复可能出现幻觉，生成的输出在语法和语义上都是正确的，但与现实脱节，并且基于错误的假设。

2.不遵循指令 LLM答复并非按照我们希望的指令或者是格式进行推理和输出，导致服务输出答非所问；或者需要多次尝试才能实现。

3.垂域不适配 LLM对于垂直领域，或者特定场景下的需求无法很好的理解和正确的输出，对于每一次的query，回答的问题无法给出正确的回答或推理结果。

4.模型输出安全问题 LLM答复中带有敏感信息，或者具有伤害性，不符合我们希望控制模型输出的需求。

5.数据实时性 LLM内化的知识无法实时的更新，需要依赖外部的工具。

...