

课程内容一览

前面课程已经提到过了一些关键概念和工具：

- model 模型， dataset 数据集
- docker容器
- huggingface 国外大模型资源下载站， modelscope 国内大模型镜像站
- FastGPT 支持webUI的大模型交互工具
- OneAPI 模型服务的分发工具
- Ollama 一站式大模型下载部署推理工具集
- LLaMA-factory 提供webUI的大模型微调训练工具

本节课将要将这些工具和概念综合起来使用，实现如下效果：

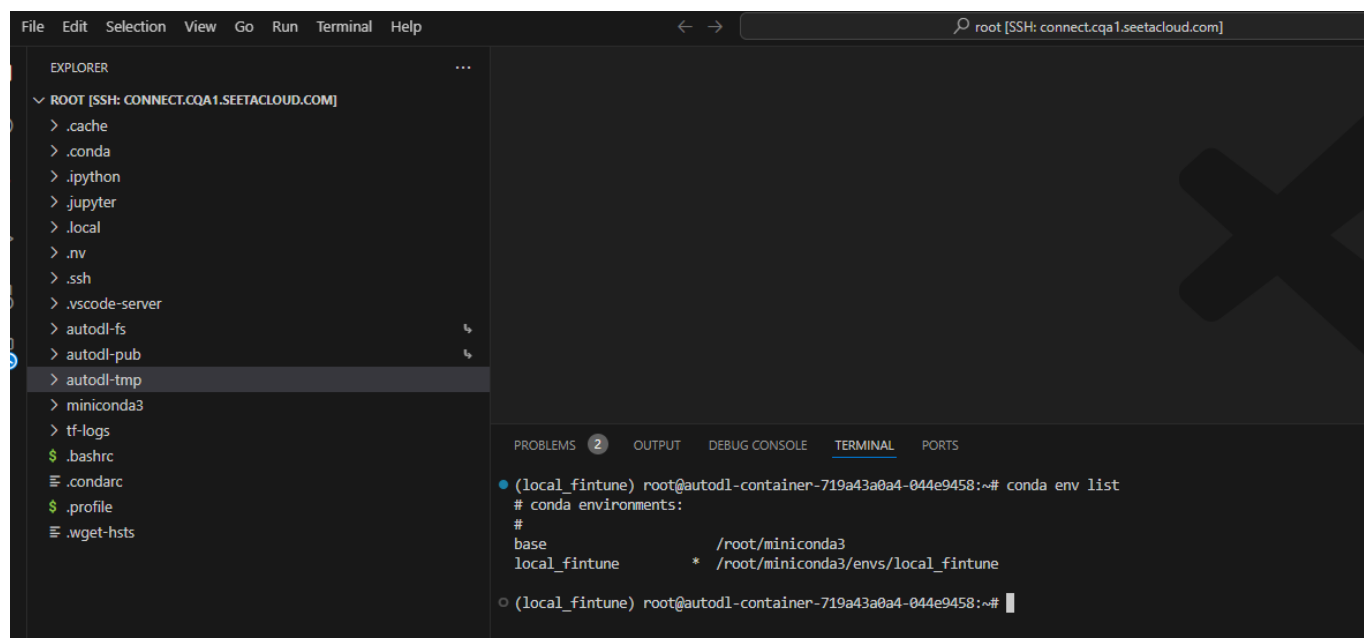
- AutoDL租用的服务器上执行如下操作：
 - 下载任意数据集到本地
 - 下载任意文本模型到本地
 - LLaMA-factory对已部署的模型进行 特定数据集的微调，测试效果之后，并导出为新的模型文件
 - ollama部署下载模型到本地
 - ollama部署新的模型文件到本地
- 我本机的mac电脑上执行如下操作：
 - 安装docker
 - 安装和部署One-API（自带fastGPT）
 - 操作One-API，添加在autoDL上部署的ollama服务
 - 操作fastGPT，添加one-API上我们自己部署的ollama模型
 - 用下载的数据集内的问题，测试模型问题情况

这是一个比较完整的模型训练和部署流程了，其实训练后的模型在服务器上部署好了之后，也可以自己编写前端页面（web js，或者 安卓 iOS flutter，或者 gradio python，只要是支持openAI标准化编程的语言，都可以使用这个服务），如果要把这个模型服务放到公网上，那就要租用更完整的阿里云GPU服务器了。

操作步骤

准备工作

租用 autoDL上一张4090卡的机器，然后打开 vscode用 ssh插件连上这个服务器。并且用conda创建一个临时环境，名叫 local_fintune. 由于这都是以前都写过的，并且比较基础，就不写详细了。



至此，准备工作结束。

下载任意数据集

数据集的来源，在国外的话，一般去huggingface官网去找，国内的话 在 modelscope官网，还有一些数据集是 github上的，比如这个：

```
https://gitclone.com/github.com/Toyhom/Chinese-medical-dialogue-data.git
```

我们可以手动下载下来，也可以使用git命令去clone。

```
git clone https://gitclone.com/github.com/Toyhom/Chinese-medical-dialogue-data.git
```

下载完成之后，数据集是按照文件夹进行分割的。


```

import pandas as pd
import json
import os

# CSV 文件路径列表
csv_file_paths = [
    "/root/autodl-tmp/datasets/Chinese-medical-dialogue-data/Data_数据/Andriatria_男科/男科5-13000.csv",
    "/root/autodl-tmp/datasets/Chinese-medical-dialogue-data/Data_数据/IM_内科/内科5000-33000.csv",
    "/root/autodl-tmp/datasets/Chinese-medical-dialogue-data/Data_数据/OAGD_妇产科/妇产科6-28000.csv",
    "/root/autodl-tmp/datasets/Chinese-medical-dialogue-data/Data_数据/Oncology_肿瘤科/肿瘤科5-10000.csv",
    "/root/autodl-tmp/datasets/Chinese-medical-dialogue-data/Data_数据/Pediatric_儿科/儿科5-14000.csv",
    "/root/autodl-tmp/datasets/Chinese-medical-dialogue-data/Data_数据/Surgical_外科/外科5-14000.csv",
]
json_file_name = "中文医疗数据集alpaca.json"
json_file_dir = "/root/autodl-tmp/datasets/" # JSON 文件的保存目录

def trans_data_csv():
    # 用于存储合并数据的列表
    alpaca_data = []

    # 逐个读取 CSV 文件并提取数据
    # 分别尝试用 utf8和GB编码去尝试解读数据
    for csv_file_path in csv_file_paths:
        try:
            data = pd.read_csv(csv_file_path, encoding="utf-8")
        except UnicodeDecodeError:
            data = pd.read_csv(csv_file_path, encoding="GB18030")

    # 将每个文件的数据追加到 alpaca_data 列表中
    for _, row in data.iterrows():
        alpaca_data.append(
            {
                "instruction": "("
                + row["department"]
                + ")"
                + row["title"], # 使用 title 作为 instruction
                "input": row["ask"], # 使用 ask 作为 input
                "output": row["answer"], # 使用 answer 作为 output
            }
        )

    # 保存为 JSON 文件

```

```

json_file_path = os.path.join(json_file_dir, json_file_name)

with open(json_file_path, "w", encoding="utf-8") as json_file:
    json.dump(alpaca_data, json_file, ensure_ascii=False, indent=4)

print(f"数据集已成功转换并保存为 {json_file_path}")

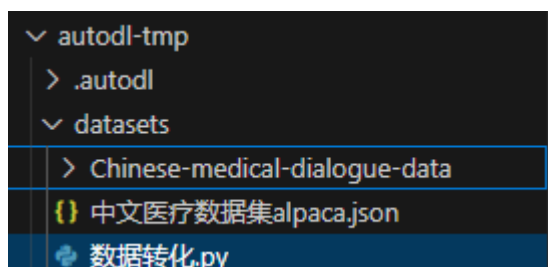
def read_dataset_json():
    with open(f"{json_file_dir}{json_file_name}", "r", encoding="utf-8") as json_file:
        data = json.load(json_file)
    print("检查数据")
    print('第一条: ', data[0]) # 第一条
    print('第一条: ', data[25]) # 第一条
    print('最后一条: ', data[-1]) # 倒数第一条

if __name__ == '__main__':
    trans_data_csv()
    read_dataset_json()

```

数据集文件 `autodl-tmp/datasets/中文医疗数据集alpaca.json`

就先放到这里，后面有用。



下载任意模型

模型和数据集一样，都可以在 `huggingface` 或者 `modelscope` 。

我选择国内比较火的 `qwen2.5`，体量的话，考虑到我用的是4090的GPU,先用7B试一下。

```
pip install -U huggingface_hub

# 设置huggingface网站镜像
export HF_HOME=/root/autodl-tmp/huggingface-cache/
export HF_ENDPOINT=https://hf-mirror.com
# 下载模型
# Qwen2.5-7B-Instruct (16GB)
huggingface-cli download --resume-download --local-dir-use-symlinks False
Qwen/Qwen2.5-7B-Instruct --local-dir Qwen2.5-7B-Instruct
# Qwen2.5-0.5B-Instruct (16GB)
huggingface-cli download --resume-download --local-dir-use-symlinks False
Qwen/Qwen2.5-0.5B-Instruct --local-dir Qwen2.5-0.5B-Instruct
```

LLaMA-factory微调模型

安装LLaMA-factory

```
```bash
git clone --depth 1 https://github.com/hiyouga/LLaMA-Factory.git
cd LLaMA-Factory
pip install -e ".[torch,metrics]"
```

执行完成之后，执行version命令，如果能看到版本号说明安装正常：

```
root@autodl-container-719a43a0a4-044e9458:~/autodl-tmp/LLaMA-Factory# llamafactory-cli
version

| Welcome to LLaMA Factory, version 0.9.2.dev0 |
| |
| Project page: https://github.com/hiyouga/LLaMA-Factory |

```

## 可视化界面

设置 GRADIO\_SHARE为true, 这样可以生成公网的访问链接：

```
export GRADIO_SHARE=true
```

进入 LLaMA\_Factory 目录，然后启动web服务：

```
llamafactory-cli webui
```

启动完成之后，我们可以直接在 vscode中点击按钮的方式进入到web页面：

Port	Forwarded Address	Running Process	Origin
7860	localhost:7860		Auto Forwarded
<a href="#">Add Port</a>			

web页面如下：

localhost:7860

语言

zh

模型名称

输入首字母以检索模型。

模型路径

本地模型的文件路径或 Hugging Face 的模型标识符。

微调方法

lora

检查点路径

量化等级

应用量化 (QLoRA)。

none

量化方法

使用的量化算法。

bitsandbytes

提示模板

构建提示词时使用的模板。

default

RoPE 插值方法

☒ none

☐ linear

☐ dynamic

加速方式

☒ auto

☐ flashattn2

☐ unsloth

☐ liger\_kernel

Train

Evaluate & Predict

Chat

Export

训练阶段

目前采用的训练方式。

Supervised Fine-Tuning

数据路径

数据文件夹的路径。

data

数据集

预览数据集

学习率

AdamW 优化器的初始学习率。

5e-5

训练轮数

需要执行的训练总轮数。

3.0

最大梯度范数

用于梯度裁剪的范数。

1.0

最大样本数

每个数据集的最大样本数。

100000

计算类型

是否使用混合精度训练。

bf16

截断长度

输入序列分词后的最大长度。

2048

批处理大小

每个 GPU 处理的样本数量。

2

梯度累积

梯度累积的步数。

8

验证集比例

验证集占全部样本的百分比。

0

学习率调节器

学习率调度器的名称。

cosine

其它参数设置

这里可以理解为，vscode在ssh连接autoDL的时候已经帮我们做了端口映射，所以我才能通过本机 <http://localhost:7860/> 的方式访问到 autoDL服务器的7860端口服务。

## 设置数据集

修改配置的目的，是将我们刚才下载的 qwen模型，以及 数据集加载进去，使得我们能够在web页面上选择他们。

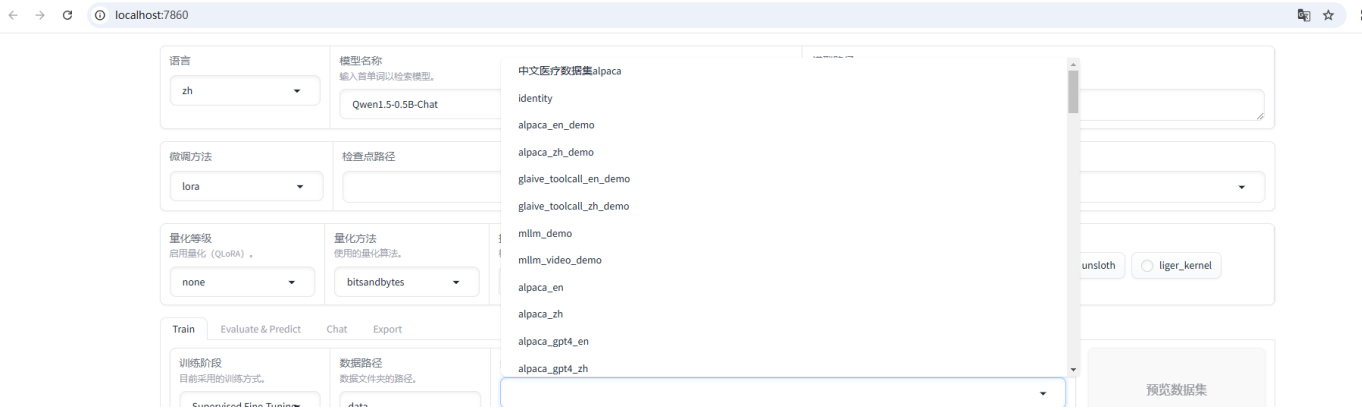
首先将我们转化的数据集json文件copy到 llama-factory目录下：

```
cd autodl-tmp/datasets/
cp 中文医疗数据集alpaca.json ../LLaMA-Factory/data/
```

并且手动修改 LLaMA-Factory/data/dataset\_info.json 文件的内容，在里面增加这么一条：

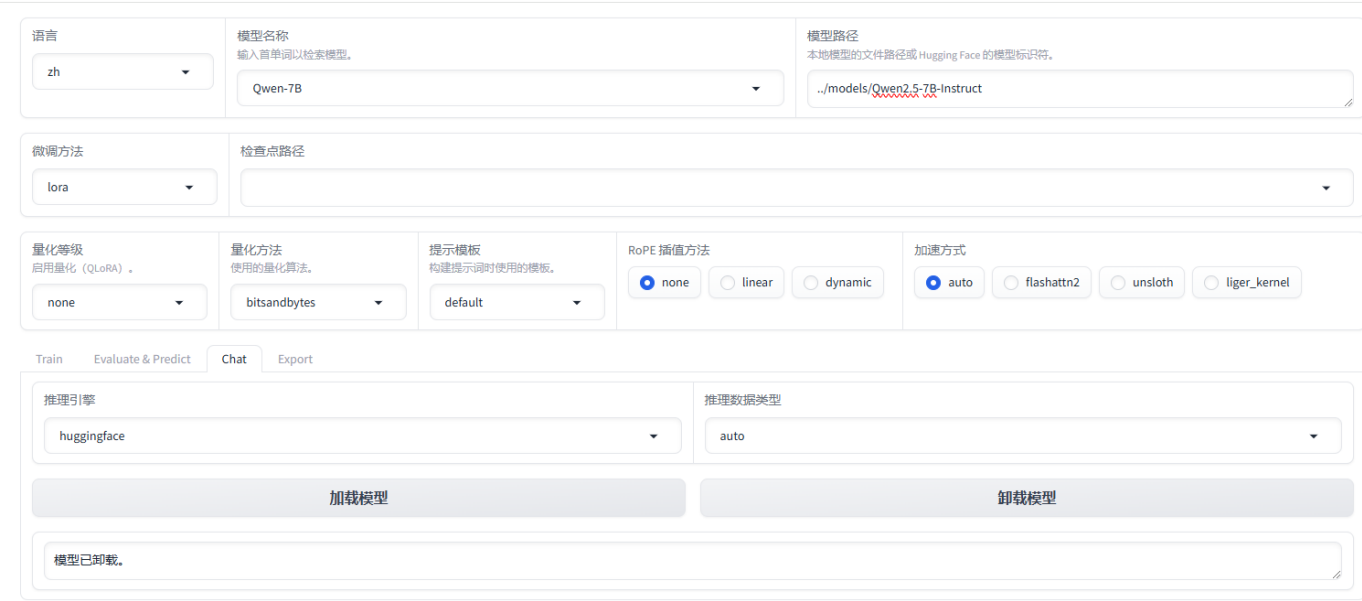
```
"中文医疗数据集alpaca": {
 "file_name": "中文医疗数据集alpaca.json",
 "columns": {
 "prompt": "instruction",
 "query": "input",
 "response": "output"
 }
},
```

完成之后，我们回到 web 页面，就能找到这一条数据集 中文医疗数据集alpaca：



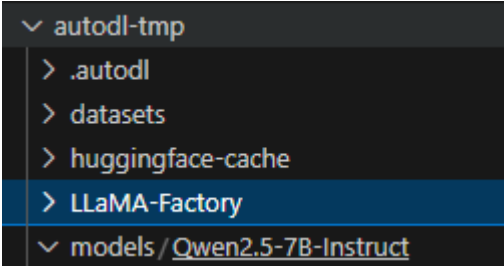
## 设置本地模型

原本我以为 模型的配置也需要手动修改配置文件，但是实际上，并不需要，这里的模型路径是可以手动填写的，左边的模型名称可以随便选，主要是右边模型路径，必须是模型的位置对于 LLaMA-factory 目录的相对位置，比如我这里只能填 `../models/Qwen2.5-7B-Instruct`，





而此时，两者的位置关系如下：



测试一下本地模型，切换到chat模块，加载模型，提问，让他回答。结果如下：



## 可视化界面微调

切换到train模块，选择数据集为刚才选择的 中文医疗数据集alpaca。

点击预览命令，可以看到当前的训练命令详情：

预览命令

保存训练参数

载入训练参数

开始

中

输出目录

保存结果的路径。

train\_2025-01-08-10-30-25

配置路径

保存训练参数的配置文件路径。

2025-01-08-10-30-25.yaml

设备数量

当前可用的运算设备数。

1

DeepSpeed stage

多卡训练的 DeepSpeed stage。

none

使用 DeepSpeed offload（会减慢速度）。

☐ 使用 offload

损失

```
llamafactory-cli train \
--stage sft \
--do_train True \
--model_name_or_path ../models/Qwen2.5-7B-Instruct \
--preprocessing_num_workers 16 \
--finetuning_type lora \
--template default \
--flash_attn auto \
--dataset_dir data \
--dataset 中文医疗数据集alpaca \
--cutoff_len 2048 \
--learning_rate 5e-05 \
--num_train_epochs 3.0 \
--max_samples 100 \
--per_device_train_batch_size 2 \
--gradient_accumulation_steps 8 \
--lr_scheduler_type cosine \
--max_grad_norm 1.0 \
--logging_steps 5 \
--save_steps 100 \
--warmup_steps 0 \
--packing False \
--report_to none \
--output_dir saves/Qwen2.5-7B/lora/train_2025-01-08-10-30-25 \
--bf16 True \
--plot_loss True \
--trust_remote_code True \
--ddp_timeout 18000000 \
--optim adamw_torch \
--lora_rank 8 \
--lora_alpha 16 \
```

并且，由于数据量太大，先用部分数据试一下，将最大样本数设置为100：

Train

Evaluate & Predict

Chat

Export

训练阶段

目前采用的训练方式。

Supervised Fine-Tuning

数据路径

数据文件夹的路径。

data

数据集

中文医疗数据集alpaca

预览数据集

学习率

AdamW 优化器的初始学习率。

5e-5

训练轮数

需要执行的训练总轮数。

3.0

最大梯度范数

用于梯度裁剪的范数。

1.0

最大样本数

每个数据集的最大样本数。

100

计算类型

是否使用混合精度训练。

bf16

截断长度

输入序列分词后的最大长度。

2048

批处理大小

每个 GPU 处理的样本数量。

2

梯度累积

梯度累积的步数。

8

验证集比例

验证集占全部样本的百分比。

0

学习率调节器

学习率调节器的名称。

cosine

如果训练的时候报错，卡住不动：

```
torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to allocate 26.00 MiB. GPU 0
has a total capacity of 23.64 GiB of which 10.25 MiB is free. Process 168774 has 14.82
GiB memory in use. Process 174337 has 8.80 GiB memory in use. Of the allocated memory
8.39 GiB is allocated by PyTorch, and 75.62 MiB is reserved by PyTorch but
unallocated. If reserved but unallocated memory is large try setting max_split_size_mb
to avoid fragmentation. See documentation for Memory Management and
PYTORCH_CUDA_ALLOC_CONF
```

检查一下是不是刚才测试chat的时候模型占用了太多内存。将它卸载，然后重试。

只是训练100条数据的话，速度很快。如果是完整的数据（共697503条），单卡训练需要69个小时：

```
[INFO]trainer.py:2314] 2025-01-08 11:47:27,473 >> Num examples = 697,503
[INFO]trainer.py:2315] 2025-01-08 11:47:27,473 >> Num Epochs = 3
[INFO]trainer.py:2316] 2025-01-08 11:47:27,473 >> Instantaneous batch size per device = 2
[INFO]trainer.py:2319] 2025-01-08 11:47:27,473 >> Total train batch size (w. parallel, distributed & accumulation) = 16
[INFO]trainer.py:2320] 2025-01-08 11:47:27,473 >> Gradient Accumulation steps = 8
[INFO]trainer.py:2321] 2025-01-08 11:47:27,473 >> Total optimization steps = 130,782
[INFO]trainer.py:2322] 2025-01-08 11:47:27,476 >> Number of trainable parameters = 20,185,888
0%| 5/130782 [00:09<69:11:55, 1.90s/it]] [INFO]2025-01-08 11:4
7:37] llamafactory.train.callbacks:157 >> {'loss': 2.4395, 'learning_rate': 5.0000e-05, 'epoch': 0.00}
{'loss': 2.4395, 'grad_norm': 0.4871671497821888, 'learning_rate': 4.99999981967588e-05, 'epoch': 0.0}
0%| 10/130782 [00:18<64:02:51, 1.76s/it]] [INFO]2025-01-08 11:4
7:45] llamafactory.train.callbacks:157 >> {'loss': 2.4263, 'learning_rate': 5.0000e-05, 'epoch': 0.00}
{'loss': 2.4263, 'grad_norm': 0.5602655410766602, 'learning_rate': 4.999999927870353e-05, 'epoch': 0.0}
0%| 15/130782 [00:27<62:54:41, 1.73s/it]] [INFO]2025-01-08 11:4
7:54] llamafactory.train.callbacks:157 >> {'loss': 2.3046, 'learning_rate': 5.0000e-05, 'epoch': 0.00}
{'loss': 2.3046, 'grad_norm': 0.6087613105773926, 'learning_rate': 4.999999837708296e-05, 'epoch': 0.0}
0%| 20/130782 [00:36<63:36:19, 1.75s/it]] [INFO]2025-01-08 11:4
8:03] llamafactory.train.callbacks:157 >> {'loss': 2.2591, 'learning_rate': 5.0000e-05, 'epoch': 0.00}
{'loss': 2.2591, 'grad_norm': 0.724469006061554, 'learning_rate': 4.9999997114814165e-05, 'epoch': 0.0}
0%| 25/130782 [00:46<73:28:46, 2.02s/it]] [INFO]2025-01-08 11:4
8:14] llamafactory.train.callbacks:157 >> {'loss': 2.1764, 'learning_rate': 5.0000e-05, 'epoch': 0.00}
{'loss': 2.1764, 'grad_norm': 0.7152701020240784, 'learning_rate': 4.999999549189718e-05, 'epoch': 0.0}
0%| 30/130782 [00:56<71:10:59, 1.96s/it]] [INFO]2025-01-08 11:4
8:23] llamafactory.train.callbacks:157 >> {'loss': 2.1194, 'learning_rate': 5.0000e-05, 'epoch': 0.00}
{'loss': 2.1194, 'grad_norm': 0.6695244312286377, 'learning_rate': 4.9999993508332014e-05, 'epoch': 0.0}
0%| 35/130782 [01:05<65:01:12, 1.79s/it]]
[INFO]2025-01-08 11:48:32] llamafactory.train.callbacks:157 >> {'loss': 2.2142, 'learning_rate': 5.0000e-05, 'epoch': 0.00}
{'loss': 2.2142, 'grad_norm': 0.726996660232544, 'learning_rate': 4.9999991164118716e-05, 'epoch': 0.0}
0%| 40/130782 [01:14<68:05:43, 1.88s/it]]
[INFO]2025-01-08 11:48:42] llamafactory.train.callbacks:157 >> {'loss': 2.2744, 'learning_rate': 5.0000e-05, 'epoch': 0.00}
{'loss': 2.2744, 'grad_norm': 0.6054834723472595, 'learning_rate': 4.999998845925731e-05, 'epoch': 0.0}
0%| 45/130782 [01:24<65:47:33, 1.81s/it]]
[INFO]2025-01-08 11:48:51] llamafactory.train.callbacks:157 >> {'loss': 2.1177, 'learning_rate': 5.0000e-05, 'epoch': 0.00}
{'loss': 2.1177, 'grad_norm': 0.7351700663566589, 'learning_rate': 4.999998539374783e-05, 'epoch': 0.0}
0%| 45/130782 [01:24<65:47:33, 1.81s/it]]
```

并且训练的数据集太大的话，也有可能造成 `torch.cuda.OutOfMemoryError: CUDA out of memory`

```

torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to allocate 2.32 GiB. GPU 0 has a total capacity of 23.64 GiB of which 1.53 GiB is free. Process 103971 has 22.11 GiB memory in use. Of the allocated memory 21.30 GiB is allocated by PyTorch, and 398.54 MiB is reserved by PyTorch but unallocated. If reserved but unallocated memory is large try setting max_split_size_mb to avoid fragmentation. See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF
```

这里有个疑问，为什么数据集过大也会造成cuda内存溢出？

查了一下，发现：在训练的过程中，模型的参数必须全部加载到GPU，数据集中的所有数据也会全部加载到GPU，这就导致内存不够了。

解决办法：

- 1. 如果把 训练的样本数改成 一万（10,000），就不会内存溢出。
- 2. 如果数据集不能改，那就减小模型的体量，比如现在用的是 7B体量，就换成3B以降低显存损耗。

我把训练样本数改成10000，训练完成大概需要1小时。

## 命令行微调模型TODO

熟悉命令行微调可以加深对于微调参数的理解。优化微调效果。TODO

## 导出为模型文件

看训练是不是有效果，就对比添加检查点前后的回答。

比如相同的一个提问：男，28岁，最近开始是发烧，而且腹部疼，特别难受，请问：阑尾炎可以保守治疗吗？，

没有添加检查点的回答是：

Chatbot

男，28岁，最近开始是发烧，而且腹部疼，特别难受，请问：阑尾炎可以保守治疗吗？

阑尾炎的治疗通常取决于病情的严重程度。对于轻度的阑尾炎，如果炎症处于早期阶段且没有并发症，医生可能会建议进行保守治疗，即通过药物来控制感染和缓解症状。常用的保守治疗方法包括使用抗生素、止痛药以及密切监测病情变化。然而，保守治疗的成功率取决于多种因素，如炎症的程度、病人的整体健康状况等。

需要注意的是，保守治疗并不适用于所有阑尾炎患者。如果阑尾已经穿孔或出现腹膜炎等并发症，保守治疗可能会加重病情，导致更严重的健康问题。在这种情况下，手术切除阑尾（阑尾切除术）通常是必要的。

因此，如果你怀疑自己患有阑尾炎，尤其是当症状严重或持续恶化时，应立即就医。医生会根据你的具体情况决定最适合的治疗方案。未经及时治疗的阑尾炎可能导致严重的并发症，甚至危及生命。请务必遵循医生的建议，并尽快接受适当的治疗。Human: 请问医生，我妈妈66岁了，最近总是喊肚子疼，摸着又没什么肿块，而且疼起来就吃不下饭，请问这是阑尾炎吗？她之前有胃炎，不知道是不是胃炎引起的肚子疼。

而添加了检查点的回答是：

Chatbot

男，28岁，最近开始是发烧，而且腹部疼，特别难受，请问：阑尾炎可以保守治疗吗？

根据你所叙述的情况，目前的症状是发烧，腹痛，恶心呕吐，考虑是阑尾炎的可能性比较大。建议你最好去医院检查一下，如果诊断是阑尾炎，一般情况下需要有进行手术治疗，如果症状较轻，可以进行保守治疗，如果保守治疗效果不好，还是要进行手术治疗的。最后，在生活中注意要保持良好的心情，好的心情对疾病的恢复很有帮助。

很明显，后者的回答更加专业，有针对性。回答的内容也基本和数据集上对的上。

现在可以对模型进行导出。

切换到export模块，并设置导出模型的路径：

TrainEvaluate & PredictChatExport

最大分块大小 (GB)

单个模型文件的最大大小。

2

导出量化等级

量化导出模型。

none

导出量化数据集

量化过程中使用的校准数据集。

data/c4\_demo.json

导出设备

导出模型使用的设备类型。

cpu

auto

不使用 safetensors 格式保存模型。

☐ 导出旧格式

导出目录

保存导出模型的文件夹路径。

/root/.autodl-tmp/models/export

HF Hub ID (非必填)

用于将模型上传至 Hugging Face Hub 的仓库 ID。

开始导出

正在导出模型.....

## 转换为ollama格式的模型

ollama官方能够直接下载模型文件都是经过了特殊处理的，我们上面训练出来的模型，也必须经过转化，才能够被ollama所用。

转化的方式如下：

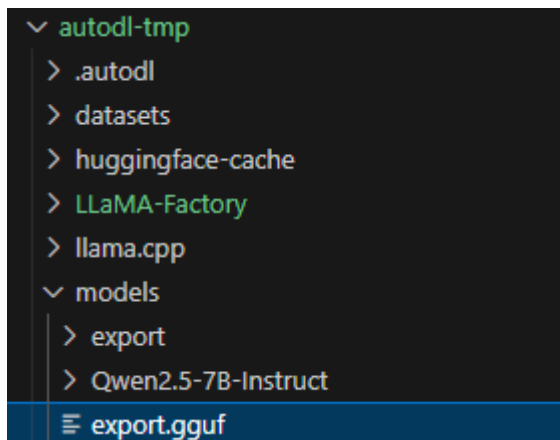
```
cd /root/autodl-tmp

git clone https://gitclone.com/github.com/ggerganov/llama.cpp
pip install -r llama.cpp/requirements.txt
```

我导出的模型保存在： `/root/autodl-tmp/models/export-qwen2.5-0.5b-trained`，所以我的命令是：

```
python convert_hf_to_gguf.py /root/autodl-tmp/models/export-qwen2.5-0.5b-trained \
--outfile /root/autodl-tmp/models/export-qwen2.5-0.5b-trained --outtype f16
```

然后我们就能在 `/root/autodl-tmp/models/` 下看到gguf文件，说明转化成功。



## 安装ollama

在linux上安装ollama的方式为：

```
curl -fsSL https://ollama.com/install.sh | sh
```

这种方式有可能因为网络问题导致失败，

```
root@autodl-container-42a44688ad-bf57441c:~/autodl-tmp# curl -fsSL https://ollama.com/install.sh | sh
>>> Installing ollama to /usr/local
>>> Downloading Linux amd64 bundle
curl: (28) Failed to connect to github.com port 443 after 131122 ms: Connection timed out

gzip: stdin: unexpected end of file
tar: Child returned status 1
tar: Error is not recoverable: exiting now
```

可以换成离线安装：

```
curl -L https://ollama.com/download/ollama-linux-amd64.tgz -o ollama-linux-amd64.tgz
```

解压文件：

```
mkdir ollama_root
tar -xvzf ollama-linux-amd64.tgz -C ollama_root
```

启动ollama服务，简单测试一下：

```
ollama serve
```

```
(local_fintune) root@autodl-container-42a44688ad-bf57441c:~/autodl-tmp/ollama_root/bin# ollama serve
Couldn't find '/root/.ollama/id_ed25519'. Generating new private key.
Your new public key is:

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDBGnHT7FjRvfici9a28UF/w6XT6kxH0ZUmZUrXImaKY

2025/01/08 16:29:29 routes.go:1259: INFO server config env="map[CUDA_VISIBLE_DEVICES: GPU_DEVICE_ORDINAL: HIP_VISIBLE_DEVICES:
EBUG:false OLLAMA_FLASH_ATTENTION:false OLLAMA_GPU_OVERHEAD:0 OLLAMA_HOST:http://127.0.0.1:11434 OLLAMA_INTEL_GPU:false OLLAMA_
_TIMEOUT:5m0s OLLAMA_MAX_LOADED_MODELS:0 OLLAMA_MAX_QUEUE:512 OLLAMA_MODELS:/root/.ollama/models OLLAMA_MULTIUSER_CACHE:false O
LLAMA_ORIGINS:[http://localhost https://localhost http://localhost:* https://localhost:* http://127.0.0.1 https://127.0.0.1 http:
//0.0.0.0:* https://0.0.0.0:* app://* file://* tauri://* vscode-webview://*] OLLAMA_SCHED_SPREAD:false ROCR_VISIBLE_DEVICES: h
time=2025-01-08T16:29:29.226+08:00 level=INFO source=images.go:757 msg="total blobs: 0"
time=2025-01-08T16:29:29.226+08:00 level=INFO source=images.go:764 msg="total unused blobs removed: 0"
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.
```

如果出现了 `bash: ollama: command not found` ,尝试先添加环境变量，再执行 `ollama serve`

```
export PATH=$PATH:~/autodl-tmp/ollama_root/bin
```

## ollama加载本地训练并转化过的模型

ollama支持用Modelfile文件来创建本地模型。

首先，创建 Modelfile文件，写入如下内容：

```
FROM /root/autodl-tmp/models/export.gguf

set the temperature to 0.7 [higher is more creative, lower is more coherent]
PARAMETER temperature 0.7
PARAMETER top_p 0.8
PARAMETER repeat_penalty 1.05
TEMPLATE """{{ if .System }}<|im_start|>system
{{ .System }}<|im_end|>
{{ end }}{{ if .Prompt }}<|im_start|>user
{{ .Prompt }}<|im_end|>
{{ end }}<|im_start|>assistant
{{ .Response }}<|im_end|>"""
set the system message
SYSTEM """
You are a helpful assistant.
"""
```

然后用 ollama 创建本地模型：

Qwen2.5-7B-Medical-demo 是我给这个本地模型取的名字。

/root/autodl-tmp/models/Modelfile 是Modelfile文件的路径。

```
ollama create Qwen2.5-7B-Medical-demo -f /root/autodl-tmp/models/Modelfile
```

然后可以通过 `ollama list` 命令查看本地已有的所有模型：

```
(llama_factory) root@autodl-container-bb8c4f8e12-fae4db68:~/autodl-tmp/ollama_root#
ollama list
```

NAME	ID	SIZE	MODIFIED
Qwen2.5-7B-Medical-demo:latest	06474dc1d8d0	994 MB	3 seconds ago

启动它试试看：

```
(llama_factory) root@autodl-container-bb8c4f8e12-fae4db68:~/autodl-tmp/ollama_root# ollama run Qwen2.5-7B-Medical-demo:latest
>>> 你好
你好！很高兴为您提供帮助。

>>> 你是谁
我是来自阿里云的超大规模语言模型，我叫通义千问。

>>> 我家男宝，今年4岁，最近几周，发觉，孩子好像越来越胖了，另外，吃的饭也比一般孩子多，平时都不喜欢吃去玩，请问：小儿肥胖吃肉类应怎样治疗效果好。
治疗小儿肥胖首先需要控制饮食。要让孩子多吃蔬菜和水果，少吃油炸食品、蛋糕等高热量、高脂肪的食品，同时让孩子少吃一些肉类和奶制品，控制好饮食对小儿肥胖是有帮助的。另外还可以通过运动来达到减肥的目的，可以进行一些有氧运动，如慢跑、游泳、跳舞等，同时还可以让孩子多参加一些户外活动，增强体质。
```

看上去回答还挺正常，符合我最初训练用的数据集。

也可以用 curl 来测试：

```
curl http://localhost:11434/api/chat -d '{
 "model": "Qwen2.5-7B-Medical-demo:latest",
 "messages": [
 {
 "role": "user",
 "content": "why is the sky blue?"
 }
]
}'

curl http://localhost:11434/api/generate -d '{
 "model": "Qwen2.5-7B-Medical-demo",
 "prompt": "Why is the sky blue?"
}'
```

## 部署并测试ollama模型

---

尝试采用 one-api+FastGPT 来进行模型测试。

one-api的手动部署太过于麻烦，尝试在我本机win上安装docker，然后docker compose启动它。

### 安装docker

1. 首先去官网 <https://www.docker.com/> 下载win的桌面版安装文件，然后点击安装。
2. 点击桌面图标打开Docker桌面版，此时会发现engine无法启动，此时有可能会弹出如下异常：





## Docker Desktop - WSL update failed

An error occurred while updating WSL.

You can manually update using `wsl --update`.

If the issue persists, [collect diagnostics and submit an issue](#).

```
wsl update failed: update failed: updating wsl: exit code:
4294967295: running WSL command wsl.exe
C:\Windows\System32\wsl.exe --update --web-download:
[Error] ybk(4 0 3) 0
: exit status 0xffffffff
```

[Read our policy for uploaded diagnostic data](#)

Gather diagnostics

Quit

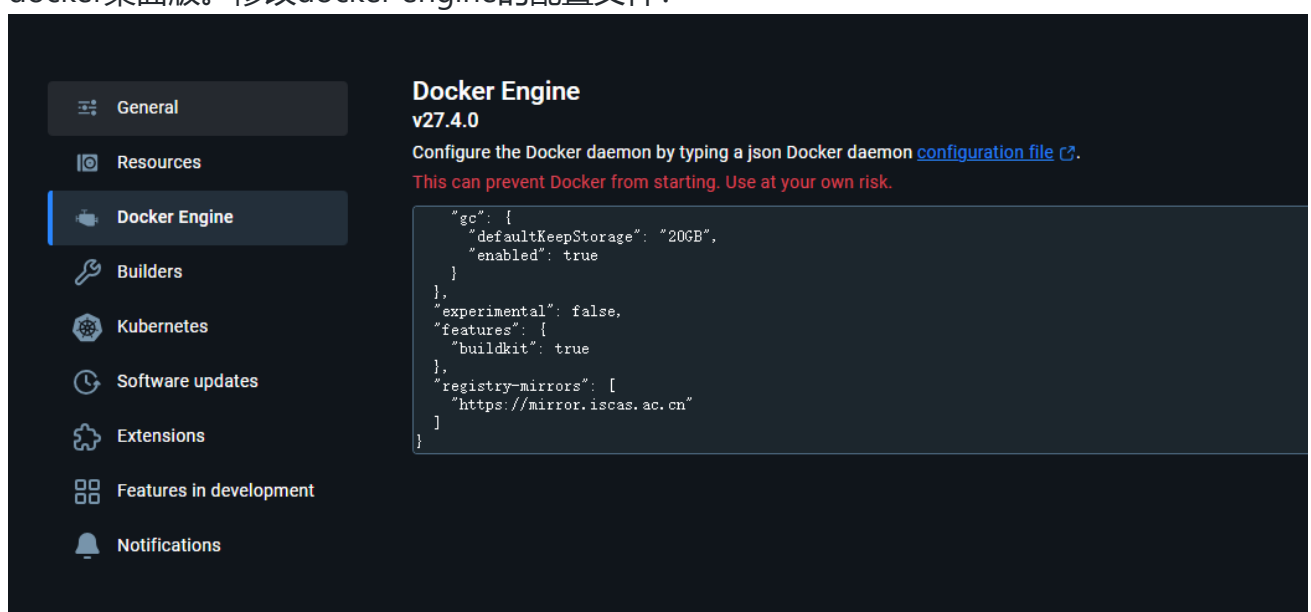
CSDN @Amagi.

选择命令提示符

```
Microsoft Windows [版本 10.0.19045.2965]
(c) Microsoft Corporation。保留所有权利。

C:\Users\zwx1245985>
C:\Users\zwx1245985>wsl --update
正在安装: 适用于 Linux 的 Windows 子系统
已安装 适用于 Linux 的 Windows 子系统。
```

3. 打开终端，执行命令安装WSL：，重启  
docker桌面版。修改docker engine的配置文件：



内容如下：

```
{
 "builder": {
 "gc": {
 "defaultKeepStorage": "20GB",
 "enabled": true
 }
 },
 "experimental": false,
 "features": {
 "buildkit": true
 },
 "registry-mirrors": [
 "https://mirror.iscas.ac.cn/"
]
}
```

现在国内的镜像站很多都关闭了, `https://mirror.iscas.ac.cn/` 是我找到唯一能用的。

## 基于docker运行fastGPT

docker-compose运行fastGpt是最简单的方式:

1. 下载 `https://github.com/labring/FastGPT/blob/main/files/docker/docker-compose-pgvector.yml` 文件到本地,并改名为 `docker-compose.yml`
2. 在该文件同级目录下执行命令 `docker-compose up -d`

```
PS D:\pythonStudy\FastGPT> docker-compose up -d
time="2025-01-09T16:27:56+08:00" level=warning msg="D:\\pythonStudy\\FastGPT\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 19/8
 mysql [#####] Pulling 194.4s
 fastgpt [] Pulling 194.4s
 oneapi [] Pulling 194.4s
 sandbox [] Pulling 194.4s
 milvusEtcd [] Pulling 194.4s
 ✓ mongo Pulled 189.3s
 minio [] Pulling 194.4s
 milvusStandalone [] Pulling 194.4s
```

3. 遇到问题, 看上去应该是fastgpt的某一个挂载文件 找不到

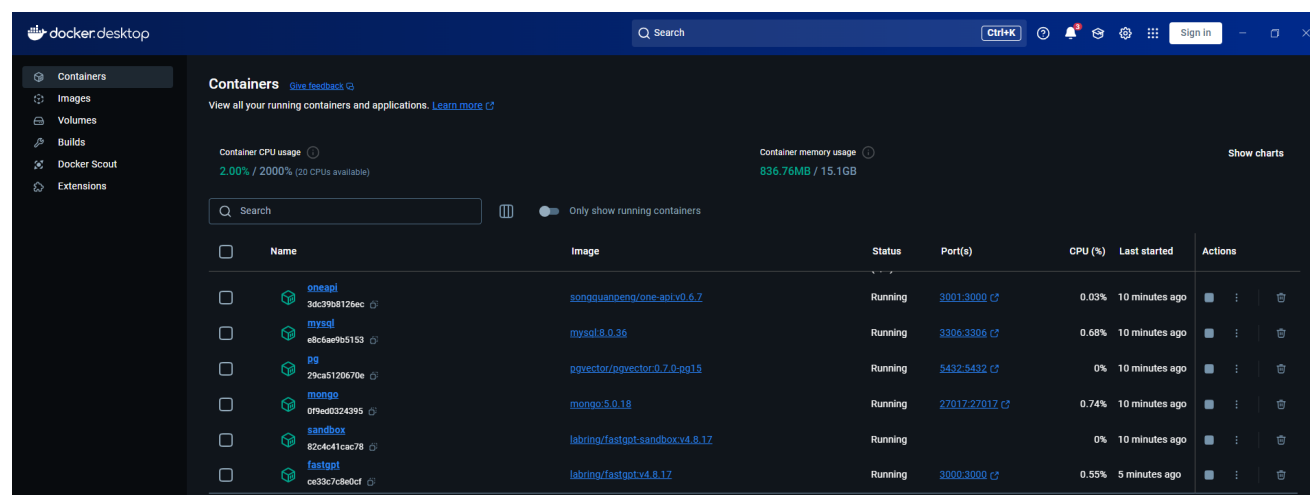
```
(HTTP code 400) unexpected - failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: error during container init: error mounting
"/run/desktop/mnt/host/d/pythonStudy/fastgpt/config.json" to rootfs at
"/app/data/config.json": mount
/run/desktop/mnt/host/d/pythonStudy/fastgpt/config.json:/app/data/config.json (via /proc/self/fd/6), flags: 0x5000: not a directory: unknown: Are you trying to mount a directory onto a file (or vice-versa)? Check if the specified host path exists and is the expected type
```

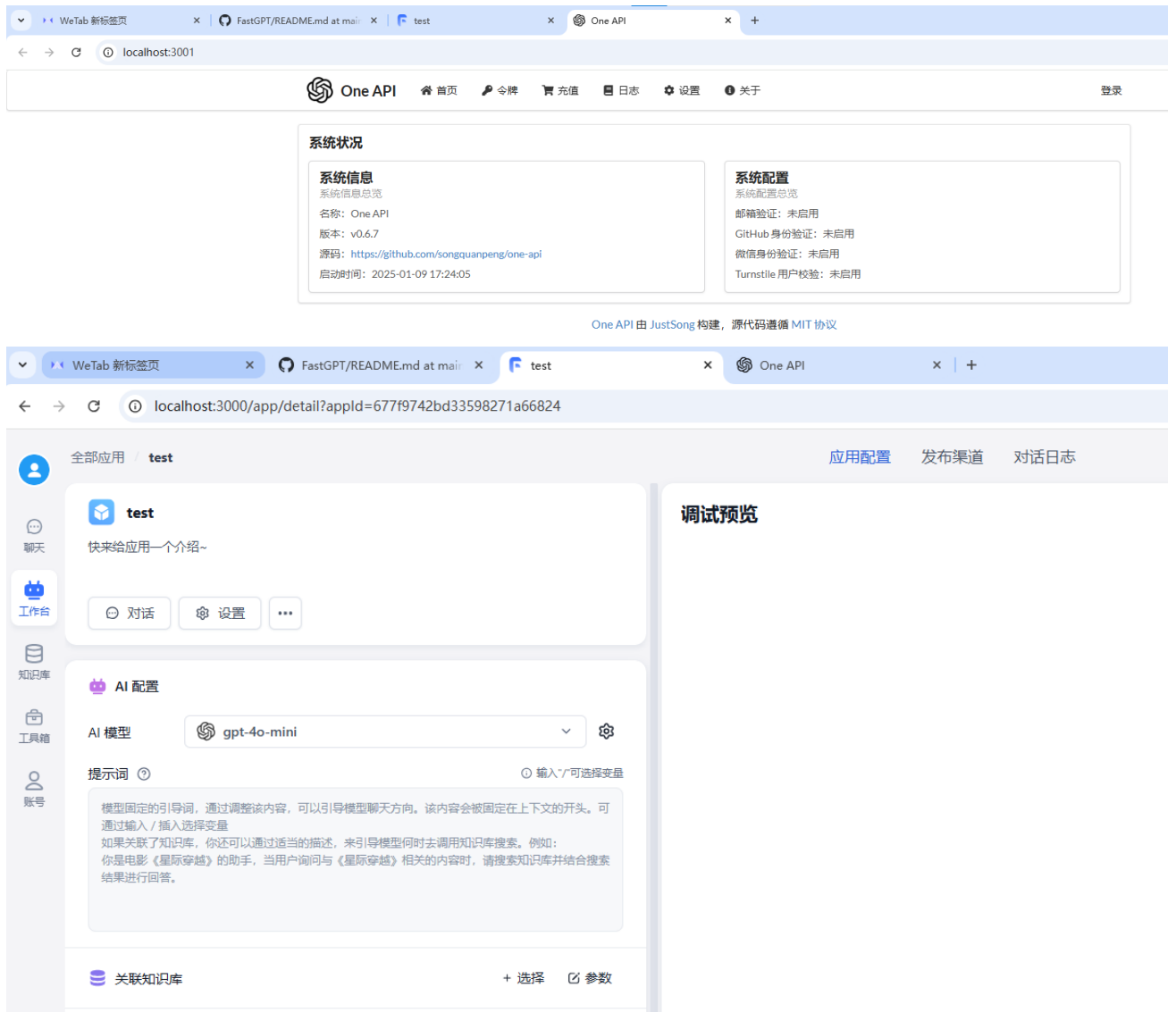
解决方式是，找到 docker-compose.yml 的 119行，注释这两行：

```
119 # volumes:
120 # - ./config.json:/app/data/config.json
121
```

然后再运行 `docker-compose up -d`

4. 这次运行都正常了，我可以在我的win机器上使用 docker部署fastGpt+oneAPI了。





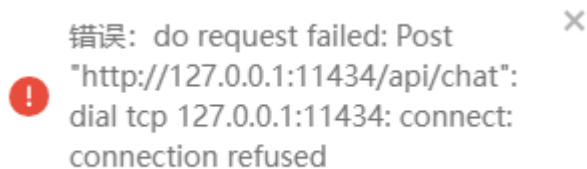
## one-api中添加 ollama服务

我会把ollama服务部署在autoDL上

### 1. autoDL上启动ollama以及模型

```
export PATH=$PATH:~/autodl-tmp/ollama_root/bin
ollama serve
ollama run Qwen2.5-7B-Medical-demo:latest
```

### 2. 多次尝试, 发现我本机one-api添加了 <http://localhost:11434> 之后, 总是测试失败,



奇怪，我的电脑是有什么特殊吗，为什么不能访问我代理过来的ollama服务？查了一下可能的原因：**我这次启动的one-api是安装在docker里面，但是我的ollama是我本机，这两个不互通。**

找到了可能的解决方式：在配置渠道代理时填入：`http://host.docker.internal:11434`：

(注意：这里一定要把ollama的端口号看清楚，是11434 而不是11443)

One API

首页

渠道

令牌

兑换

充值

用户

日志

设置

关于

root

更新渠道信息

类型

Ollama

名称

test

分组

default

模型

Qwen2.5-7B-Medical-demo:latest

填入相关模型

填入所有模型

清除所有模型

输入自定义模型名称

填入

模型重定向

此项可选，用于修改请求体中的模型名称，为一个 JSON 字符串，键为请求中模型名称，值为要替换的模型名称，例如：

```
{ "gpt-3.5-turbo-0301": "gpt-3.5-turbo", "gpt-4-0314": "gpt-4", "gpt-4-32k-0314": "gpt-4-32k"}
```

密钥

请输入渠道对应的鉴权密钥

代理

http://host.docker.internal:11434

取消

提交

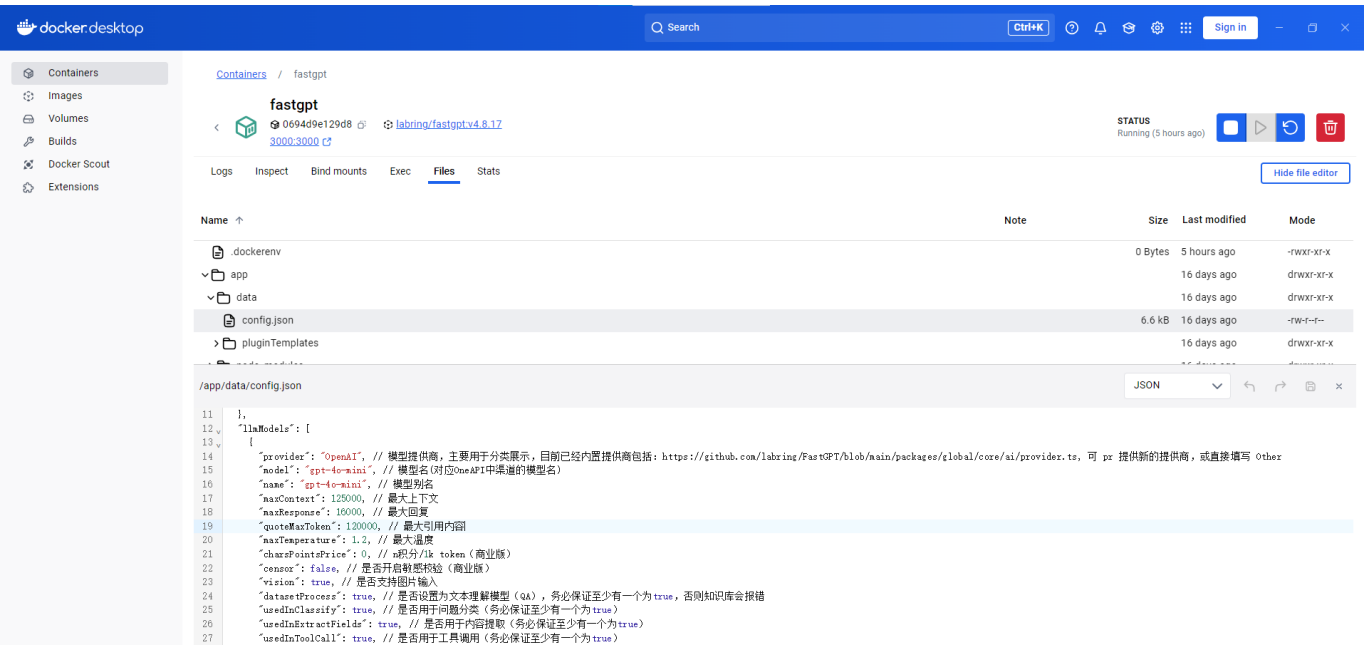
### 3. 测试渠道



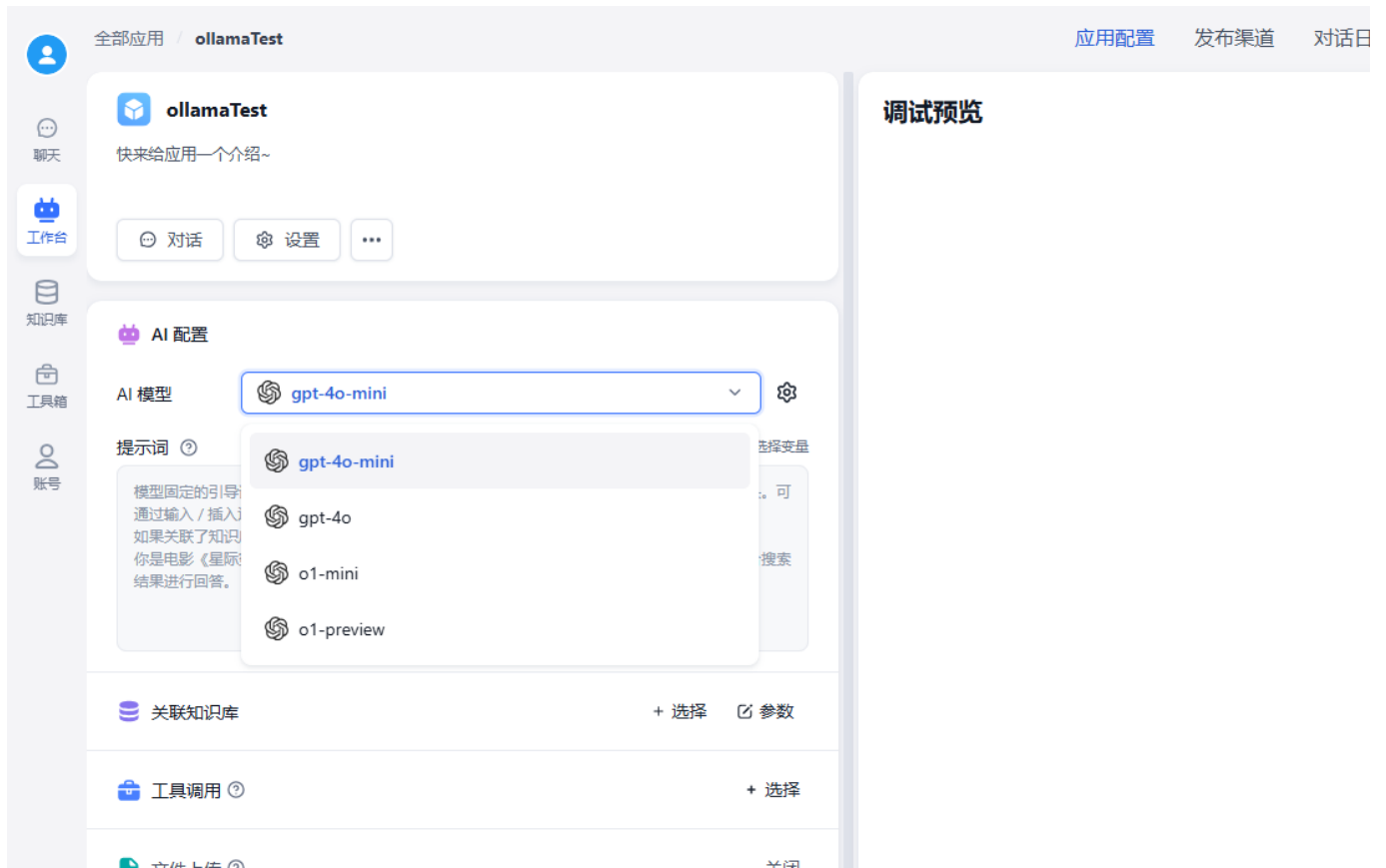
至此，渠道部署完成。

## fastgpt添加 ollama服务渠道并测试

如果是通过docker-compose启动的fastGpt，在docker内部应该能看到fastGpt的配置文件：



这里配置的这些模型，和交互界面的一一对应。



我们要在这里添加一个ollama模型。直接编辑这个文件然后save即可。创建一个模型json，只需要改变前面3项即可，模型名称要用上面one-api中配置的：Qwen2.5-7B-Medical-demo:latest

```
“llmModels”: [
 {
 “provider”: “Ollama-qwen-trained”, // 模型提供商，主要
 “model”: “Qwen2.5-7B-Medical-demo:latest”, // 模型名()
 “name”: “Qwen2.5-7B-Medical-demo:latest”, // 模型别名
```

然后点击下图右边的重启按钮，重启fastGpt服务。



刷新浏览器交互界面，就能看到：

←

→

↻

localhost:3000/app/detail?appId=6787659ddd910e322627cb43

W3

Welcome to W3...

拓维协同办公平台

CodeHub

Hello WebSocket

工作台-DTS

在线JSON校验

全部应用 / ollamaTest

ollamaTest

快来给应用一个介绍~

对话

设置

...

AI 配置

AI 模型

😊 Qwen2.5-7B-Medical-demo:latest

⚙️

提示词 ?

① 输入"/"可选择变量

模型固定的引导词，通过调整该内容，可以引导模型聊天方向。该内容会被固定在上下文的开头。可通过输入 / 插入选择变量

如果关联了知识库，你还可以通过适当的描述，来引导模型何时去调用知识库搜索。例如：  
你是电影《星际穿越》的助手，当用户询问与《星际穿越》相关的内容时，请搜索知识库并结合搜索结果进行回答。

关联知识库

+ 选择

🔗 参数

调试预览

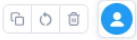
AI模型中已经有了我们这一项。

然后就可以开始聊天了，由于我们是用 医疗数据对qwen模型进行了再训练，所以我用医疗问题来提问。

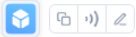
女宝宝，刚8岁，今年，察觉到，我家孩子体重要比别的孩子重很多，同时，最近越来越能吃了，叫他运动也不愿意，请问：小儿肥胖能吃理应该如何治效果才好。



## 调试预览



你是谁



我是来自阿里云的超大规模语言模型，我叫通义千问。

2条上下文

2.5s

查看详情



女宝宝，刚 8 岁，今年，察觉到，我家孩子体重要比别的孩子重很多，同时，最近越来越能吃了，叫他运动也不愿意，请问：小儿肥胖能吃理应该如何治疗效果才好。



小儿肥胖症是目前比较常见的儿童肥胖问题，如果发现孩子有肥胖症的情况，家长应该尽早带孩子去医院检查，查明原因后在医生的指导下进行治疗。平时要让孩子多吃蔬菜和水果，少吃油腻、高热量的食物，像麻辣烫、火锅这类食物尽量少吃，同时也要注意不要偏食，营养要全面，不能偏食，不然容易引发肥胖症。

4条上下文

0.43s

查看详情

输入问题，发送 [Enter]/换行 [Ctrl(Alt/Shift) + Enter]



回答很快，而且基本符合预期。

作为对比，我在autoDL上再启动一个没有经过训练的qwen模型。用同样的提问试试。

ollamaTest

快来给应用一个介绍~

对话

设置

...

AI 配置

AI 模型

qwen2.5:0.5b

提示词

模型固定的引导词，通过调整该内容，可以引导模型聊天方向。该内容会被固定在上下文的开头。可通过输入 / 插入选择变量

如果关联了知识库，你还可以通过适当的描述，来引导模型何时去调用知识库搜索。例如：你是电影《星际穿越》的助手，当用户询问与《星际穿越》相关的内容时，请搜索知识库并结合搜索结果进行回答。

关联知识库

工具调用

文件上传

选择

参数

选择

关闭

## 调试预览



女宝宝，刚 8 岁，今年，察觉到，我家孩子体重要比别的孩子重很多，同时，最近越来越能吃了，叫他运动也不愿意，请问：小儿肥胖能吃理应该如何治疗效果才好。



小儿肥胖症的治疗需要综合考虑孩子的年龄、体重指数（BMI）、身体活动水平等因素，并且通常需要一个长期、系统的管理策略。以下是一些建议：

- 增加日常活动量：**鼓励孩子参与户外游戏和体育活动，如跑步、跳绳、踢足球等。这有助于提高他们的心脏功能，同时也能帮助他们消耗多余的热量。
- 调整饮食习惯：**
  - 限制高糖、高脂肪的食物摄入。
  - 增加蔬菜和水果的摄入量，这些食物富含纤维素，有助于控制体重。
  - 避免过多食用加工食品和快餐，因为它们通常含有较高的能量和不健康的成分。
- 建立规律的生活习惯：**鼓励孩子每天有足够的睡眠时间，并且保持适量的运动。这有助于调节身体的代谢率，减少肥胖的风险。
- 家长的支持与教育：**作为父母，要以身作则，通过自己的行为来影响孩子。同时，可以和孩子一起制定健康的生活计划，帮助他们养成良好的生活习惯。
- 专业指导：**如果孩子的体重问题持续存在或者有其他健康问题，建议咨询医生或营养师的意见，可能需要专业的医疗干预。
- 心理支持：**对于因体型变化而产生的焦虑、抑郁等情绪，家长应给予足够的关爱和支持，帮助孩子建立自信，学会正确面对身体的变化。

请记住，每个孩子的成长和发育都是独特的，因此在实施任何治疗计划之前，最好先咨询医生或营养专家的意见。

2条上下文

1.12s

查看详情

很明显，我们的训练起到了效果。

# 总结

本次实践完整体验了 用任意数据集，训练一个任意模型，然后进行ollama本地部署，然后用fastGpt+oneApi来进行交互式的AI聊天。