

基础作业

训练自己的小助手认知（记录复现过程并截图）

准备工作

创建开发机

开发机名称	资源配置	状态 ▾	工具	操作
 RGB128_XTuner id: 20240419-7ef0487-40069428	 10% A100 升降配置	 运行中 1算力点/小时	资源监控 操作日志 自定义服务	SSH 连接 进入开发机 停止 释放

XTuner原理

环境安装

安装配套库

```
# 如果你是在 InternStudio 平台，则从本地 clone 一个已有 pytorch 的环境：
# pytorch      2.0.1   py3.10_cuda11.7_cudnn8.5.0_0

studio-conda xtuner0.1.17
# 如果你是在其他平台：
# conda create --name xtuner0.1.17 python=3.10 -y

# 激活环境
conda activate xtuner0.1.17
# 进入家目录 （~的意思是“当前用户的home路径”）
cd ~
# 创建版本文件夹并进入，以跟随本教程
mkdir -p /root/xtuner0117 && cd /root/xtuner0117

# 拉取 0.1.17 的版本源码
git clone -b v0.1.17 https://github.com/InternLM/xtuner
# 无法访问github的用户请从 gitee 拉取：
# git clone -b v0.1.15 https://gitee.com/Internlm/xtuner

# 进入源码目录
cd /root/xtuner0117/xtuner

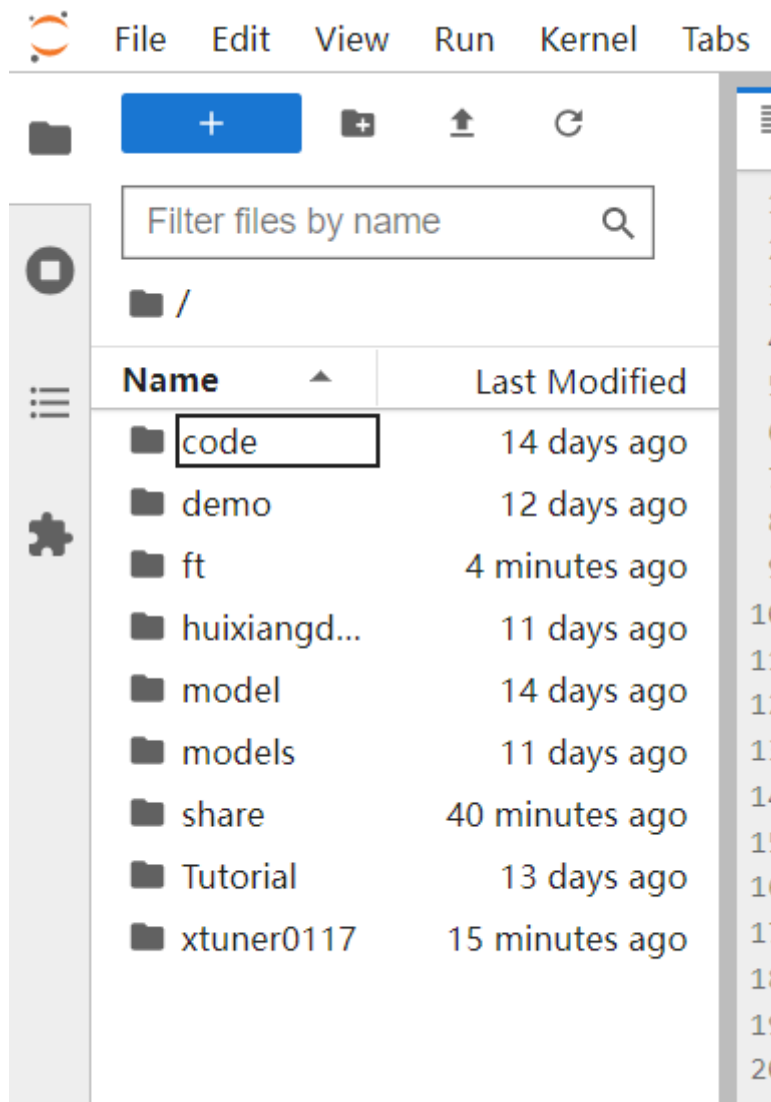
# 从源码安装 XTuner
pip install -e '.[all]'
```

克隆 XTuner 源码

```
ps://pip.pypa.io/en/stable/
Installed kernelspec xtuner0.1.17 in /root/.local/share/jupyter/kernels/xtuner0.1.17
conda环境: xtuner0.1.17安装成功!

=====
ALL DONE!
=====

(base) root@intern-studio-40069428:~# conda activate xtuner0.1.17
(xtuner0.1.17) root@intern-studio-40069428:~# # 进入家目录 (~的意思是 当前用户的home路径)
(xtuner0.1.17) root@intern-studio-40069428:~# cd ~
(xtuner0.1.17) root@intern-studio-40069428:~# # 创建版本文件夹并进入, 以跟随本教程
(xtuner0.1.17) root@intern-studio-40069428:~# mkdir -p /root/xtuner0117 && cd /root/xtuner0117
(xtuner0.1.17) root@intern-studio-40069428:~/xtuner0117#
```



数据集准备

选择与存放

使用库尔特·冯内古特小说，作为小助手风格化训练素材，创建文件夹存放训练文件

```
# 前半部分是创建一个文件夹, 后半部分是进入该文件夹。
mkdir -p /root/ft && cd /root/ft

# 在ft这个文件夹里再创建一个存放数据的数据文件夹
mkdir -p /root/ft/data && cd /root/ft/data
```

自动生成数据集

在 `data` 目录下新建 `generate_data.py` 文件，更新内容并运行脚本即可生成数据集。

```
# 创建 `generate_data.py` 文件
touch /root/ft/data/generate_data.py
```

```
#generate_data.py
import json

# 设置用户的名字
name = '常灰'
# 设置需要重复添加的数据次数
n = 10000

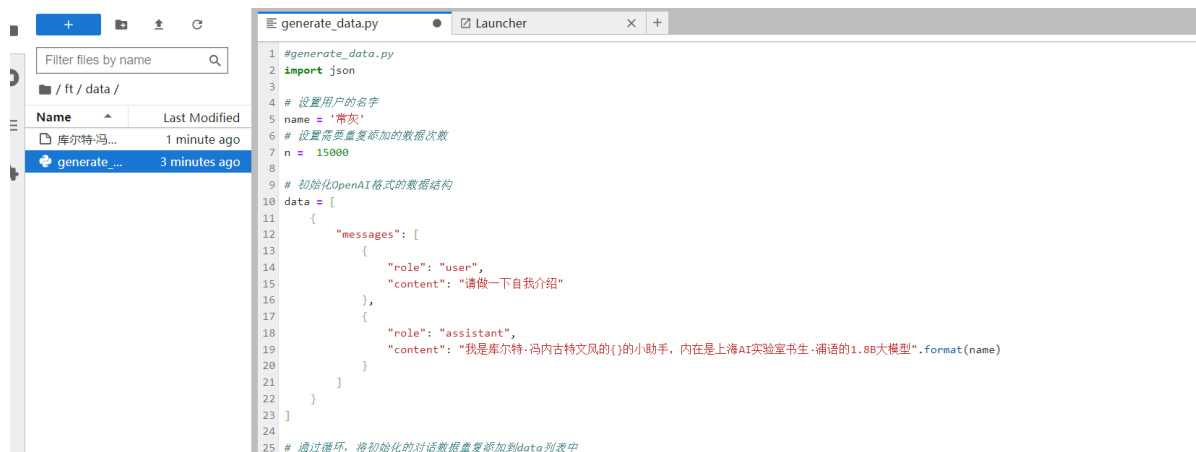
# 初始化OpenAI格式的数据结构
data = [
    {
        "messages": [
            {
                "role": "user",
                "content": "请做一下自我介绍"
            },
            {
                "role": "assistant",
                "content": "我是库尔特·冯内古特文风的{}的小助手，内在是上海AI实验室书生·浦语的1.8B大模型".format(name)
            }
        ]
    }
]

# 通过循环，将初始化的对话数据重复添加到data列表中
for i in range(n):
    data.append(data[0])

# 将data列表中的数据写入到一个名为'personal_assistant.json'的文件中
with open('personal_assistant.json', 'w', encoding='utf-8') as f:
    # 使用json.dump方法将数据以JSON格式写入文件
    # ensure_ascii=False 确保中文字符正常显示
    # indent=4 使得文件内容格式化，便于阅读
    json.dump(data, f, ensure_ascii=False, indent=4)
```

个性化

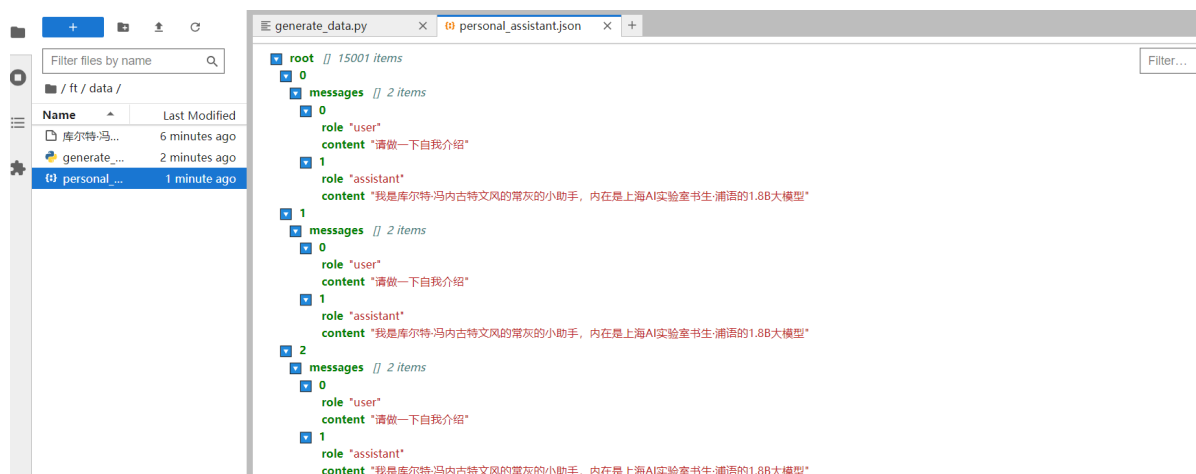
修改name信息、参数n、openAI数据结构



修改完成后运行 `generate_data.py` 文件即可。

```
# 确保先进入该文件夹
cd /root/ft/data
# 运行代码
python /root/ft/data/generate_data.py
```

data的路径下生成了一个名为 `personal_assistant.json` 的文件，这样我们最可用于微调的数据集就准备好啦！里面就包含了 5000 条 `input` 和 `output` 的数据对



打印文件结构树

使用方法为在终端调用该代码的同时在后方输入文件夹路径。

```
# 创建 `tree.py` 文件
touch /root/tree.py
```

填入代码

```
import os
import argparse

def print_dir_tree(startpath, prefix=''):
    """递归地打印目录树结构。"""
    contents = [os.path.join(startpath, d) for d in os.listdir(startpath)]
    directories = [d for d in contents if os.path.isdir(d)]
    files = [f for f in contents if os.path.isfile(f)]

    if files:
```

```

        for f in files:
            print(prefix + '|-- ' + os.path.basename(f))
    if directories:
        for d in directories:
            print(prefix + '|-- ' + os.path.basename(d) + '/')
            print_dir_tree(d, prefix=prefix + '    ')

def main():
    parser = argparse.ArgumentParser(description='打印目录树结构')
    parser.add_argument('folder', type=str, help='要打印的文件夹路径')

    args = parser.parse_args()

    print('|-- ' + os.path.basename(args.folder) + '/')
    print_dir_tree(args.folder, '    ')

if __name__ == "__main__":
    main()

```

在终端输入

```
python /root/tree.py /root/ft/data
```

效果

```

(xtuner0.1.17) root@intern-studio-40069428: # touch /root/tree.py
(xtuner0.1.17) root@intern-studio-40069428: # ^C
(xtuner0.1.17) root@intern-studio-40069428: # python /root/tree.py /root/ft/data
|-- data/
|   |-- personal_assistant.json
|   |-- generate_data.py
|   |-- 库尔特·冯内古特 - 五号屠场.txt
|   |-- .ipynb_checkpoints/
|   |-- generate_data-checkpoint.py
|   |-- personal_assistant-checkpoint.json

```

模型准备

在 InternStudio 上运行，通过以下代码一键创建文件夹并将所有文件复制进去。

```

# 创建目标文件夹，确保它存在。
# -p选项意味着如果上级目录不存在也会一并创建，且如果目标文件夹已存在则不会报错。
mkdir -p /root/ft/model

# 复制内容到目标文件夹。-r选项表示递归复制整个文件夹。
cp -r /root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-1_8b/*
/root/ft/model/

```

打印model 文件结构树

```
(xtuner0.1.17) root@intern-studio-40069428:~# python /root/tree.py /root/model
|-- model/
|   |-- Shanghai_AI_Laboratory/
|       |-- internlm-chat-7b/
|           |-- tokenizer.model
|           |-- config.json
|           |-- pytorch_model.bin.index.json
|           |-- configuration_internlm.py
|           |-- .mdl
|           |-- pytorch_model-00003-of-00008.bin
|           |-- pytorch_model-00004-of-00008.bin
|           |-- tokenizer_config.json
|           |-- pytorch_model-00008-of-00008.bin
|           |-- pytorch_model-00007-of-00008.bin
|           |-- configuration.json
|           |-- .msc
|           |-- special_tokens_map.json
|           |-- pytorch_model-00006-of-00008.bin
|           |-- pytorch_model-00002-of-00008.bin
|           |-- modeling_internlm.py
|           |-- pytorch_model-00001-of-00008.bin
|           |-- tokenization_internlm.py
|           |-- pytorch_model-00005-of-00008.bin
|           |-- README.md
|           |-- generation_config.json
```

假如存储空间不足，通过以下代码一键通过符号链接的方式链接到模型文件，节省了空间也便于管理。

```
# 删除/root/ft/model目录
rm -rf /root/ft/model

# 创建符号链接
ln -s /root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-1_8b
/root/ft/model
```

执行上述操作后，`/root/ft/model` 将直接成为一个符号链接，这个链接指向 `/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-1_8b` 的位置。

这意味着，当我们访问 `/root/ft/model` 时，实际上就是在访问 `/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-1_8b` 目录下的内容。通过这种方式，我们无需复制任何数据，就可以直接利用现有的模型文件进行后续的微调操作，从而节省存储空间并简化文件管理。

在该情况下的文件结构如下所示，可以看到和上面的区别在于多了一些软链接相关的文件。

配置文件选择

配置文件 (config)

是一种用于定义和控制模型训练和测试过程中各个方面的参数和设置的工具。

```
(base) root@intern-studio-40069428:~# conda activate xtuner0.1.17
(xtuner0.1.17) root@intern-studio-40069428:~# # 列出所有内置配置文件
(xtuner0.1.17) root@intern-studio-40069428:~# # xtuner list-cfg
(xtuner0.1.17) root@intern-studio-40069428:~#
(xtuner0.1.17) root@intern-studio-40069428:~# # 假如我们想找到 internlm2-1.8b 模型里支持的配置文件
(xtuner0.1.17) root@intern-studio-40069428:~# xtuner list-cfg -p internlm2_1_8b
=====CONFIGS=====
PATTERN: internlm2_1_8b
-----
internlm2_1_8b_full_alpaca_e3
internlm2_1_8b_qlora_alpaca_e3
=====
```

XTuner 的工具 `list-cfg` 可以选择不添加额外的参数，参数 `-p` 或 `--pattern` 可用于进行模糊匹配搜索。我们可以用来搜索特定模型的配置文件，比如例子中的 `internlm2_1_8b`，也可以用来搜索像是微调方法 `qlora`。根据上面的定向搜索指令可以看到目前只有两个支持 `internlm2-1.8B` 的模型配置文件。

配置文件名的解释

以 `internlm2_1_8b_qlora_alpaca_e3` 举例：

模型名	说明
internlm2_1_8b	模型名称
qlora	使用的算法
alpaca	数据集名称
e3	把数据集跑3次

复制配置文件

最相近的配置文件是 `internlm2_1_8b_qlora_alpaca_e3`，因此选择拷贝这个配置文件到当前目录：

```
# 创建一个存放 config 文件的文件夹
mkdir -p /root/ft/config

# 使用 XTuner 中的 copy-cfg 功能将 config 文件复制到指定的位置
xtuner copy-cfg internlm2_1_8b_qlora_alpaca_e3 /root/ft/config
```

这里我们就用到了 XTuner 工具箱中的第二个工具 `copy-cfg`，该工具有两个必须要填写的参数 `{CONFIG_NAME}` 和 `{SAVE_PATH}`，在我们的输入的这个指令中，我们的 `{CONFIG_NAME}` 对应的是上面搜索到的 `internlm2_1_8b_qlora_alpaca_e3`，而 `{SAVE_PATH}` 则对应的是刚刚新建的 `/root/ft/config`。我们假如需要复制其他的配置文件只需要修改这两个参数即可实现。输入后我们就能够看到在我们的 `/root/ft/config` 文件夹下有一个名为 `internlm2_1_8b_qlora_alpaca_e3_copy.py` 的文件了。

```
(base) root@intern-studio-40069428:~# conda activate xtuner0.1.17
(xtuner0.1.17) root@intern-studio-40069428:~# # 创建一个存放 config 文件的文件夹
(xtuner0.1.17) root@intern-studio-40069428:~# mkdir -p /root/ft/config
(xtuner0.1.17) root@intern-studio-40069428:~#
(xtuner0.1.17) root@intern-studio-40069428:~# # 使用 XTuner 中的 copy-cfg 功能将 config 文件复制到指定的位置
(xtuner0.1.17) root@intern-studio-40069428:~# xtuner copy-cfg internlm2_1_8b_qlora_alpaca_e3 /root/ft/config
Copy to /root/ft/config/internlm2_1_8b_qlora_alpaca_e3_copy.py
(xtuner0.1.17) root@intern-studio-40069428:~# python /root/tree.py /root/ft/config
├─ config/
│   └─ internlm2_1_8b_qlora_alpaca_e3_copy.py
```

打印ft结构树

```
(base) root@intern-studio-40069428:~# python /root/tree.py /root/ft
|-- ft/
|   |-- config/
|   |   |-- internlm2_1_8b_qlora_alpaca_e3_copy.py
|   |-- model/
|   |   |-- tokenizer.model
|   |   |-- config.json
|   |   |-- tokenization_internlm2.py
|   |   |-- model-00002-of-00002.safetensors
|   |   |-- tokenizer_config.json
|   |   |-- model-00001-of-00002.safetensors
|   |   |-- model.safetensors.index.json
|   |   |-- configuration.json
|   |   |-- special_tokens_map.json
|   |   |-- modeling_internlm2.py
|   |   |-- README.md
|   |   |-- configuration_internlm2.py
|   |   |-- generation_config.json
|   |   |-- tokenization_internlm2_fast.py
|   |   |-- .ipynb_checkpoints/
|   |-- data/
|   |   |-- personal_assistant.json
|   |   |-- generate_data.py
|   |   |-- 库尔特·冯内古特 - 五号屠场.txt
|   |   |-- .ipynb_checkpoints/
|   |   |-- generate_data-checkpoint.py
|   |   |-- personal_assistant-checkpoint.json
```

配置文件修改

配置文件介绍

1. **PART 1 Settings**: 涵盖了模型基本设置，如预训练模型的选择、数据集信息和训练过程中的一些基本参数（如批大小、学习率等）。
2. **PART 2 Model & Tokenizer**: 指定了用于训练的模型和分词器的具体类型及其配置，包括预训练模型的路径和是否启用特定功能（如可变长度注意力），这是模型训练的核心组成部分。
3. **PART 3 Dataset & Dataloader**: 描述了数据处理的细节，包括如何加载数据集、预处理步骤、批处理大小等，确保了模型能够接收到正确格式和质量的数据。
4. **PART 4 Scheduler & Optimizer**: 配置了优化过程中的关键参数，如学习率调度策略和优化器的选择，这些是影响模型训练效果和速度的重要因素。
5. **PART 5 Runtime**: 定义了训练过程中的额外设置，如日志记录、模型保存策略和自定义钩子等，以支持训练流程的监控、调试和结果的保存。

将以下代码复制到 `/root/ft/config/internlm2_1_8b_qlora_alpaca_e3_copy.py` 文件中

```
# Copyright (c) OpenMMLab. All rights reserved.
import torch
from datasets import load_dataset
from mmengine.dataset import DefaultSampler
from mmengine.hooks import (CheckpointHook, DistSamplerSeedHook, IterTimerHook,
                             LoggerHook, ParamSchedulerHook)
from mmengine.optim import AmpOptimWrapper, CosineAnnealingLR, LinearLR
from peft import LoraConfig
from torch.optim import AdamW
from transformers import (AutoModelForCausalLM, AutoTokenizer,
                           BitsAndBytesConfig)
```



```

from xtuner.dataset import process_hf_dataset
from xtuner.dataset.collate_fns import default_collate_fn
from xtuner.dataset.map_fns import openai_map_fn, template_map_fn_factory
from xtuner.engine.hooks import (DatasetInfoHook, EvaluateChatHook,
                                  VarlenAttnArgsToMessageHubHook)
from xtuner.engine.runner import TrainLoop
from xtuner.model import SupervisedFinetune
from xtuner.parallel.sequence import SequenceParallelSampler
from xtuner.utils import PROMPT_TEMPLATE, SYSTEM_TEMPLATE

#####
#                                PART 1  Settings                                #
#####
# Model
pretrained_model_name_or_path = '/root/ft/model'
use_varlen_attn = False

# Data
alpaca_en_path = '/root/ft/data/personal_assistant.json'
prompt_template = PROMPT_TEMPLATE.default
max_length = 1024
pack_to_max_length = True

# parallel
sequence_parallel_size = 1

# Scheduler & Optimizer
batch_size = 1 # per_device
accumulative_counts = 16
accumulative_counts *= sequence_parallel_size
dataloader_num_workers = 0
max_epochs = 2
optim_type = AdamW
lr = 2e-4
betas = (0.9, 0.999)
weight_decay = 0
max_norm = 1 # grad clip
warmup_ratio = 0.03

# Save
save_steps = 300
save_total_limit = 3 # Maximum checkpoints to keep (-1 means unlimited)

# Evaluate the generation performance during the training
evaluation_freq = 300
SYSTEM = ''
evaluation_inputs = ['请介绍一下你自己', '你是谁', '你是我的小助手吗']

#####
#                                PART 2  Model & Tokenizer                        #
#####
tokenizer = dict(
    type=AutoTokenizer.from_pretrained,
    pretrained_model_name_or_path=pretrained_model_name_or_path,
    trust_remote_code=True,
    padding_side='right')

```

```

model = dict(
    type=SupervisedFinetune,
    use_varlen_attn=use_varlen_attn,
    llm=dict(
        type=AutoModelForCausalLM.from_pretrained,
        pretrained_model_name_or_path=pretrained_model_name_or_path,
        trust_remote_code=True,
        torch_dtype=torch.float16,
        quantization_config=dict(
            type=BitsAndBytesConfig,
            load_in_4bit=True,
            load_in_8bit=False,
            llm_int8_threshold=6.0,
            llm_int8_has_fp16_weight=False,
            bnb_4bit_compute_dtype=torch.float16,
            bnb_4bit_use_double_quant=True,
            bnb_4bit_quant_type='nf4')),
    lora=dict(
        type=LoraConfig,
        r=64,
        lora_alpha=16,
        lora_dropout=0.1,
        bias='none',
        task_type='CAUSAL_LM'))

#####
#                                PART 3  Dataset & Dataloader                                #
#####
alpaca_en = dict(
    type=process_hf_dataset,
    dataset=dict(type=load_dataset, path='json',
data_files=dict(train=alpaca_en_path)),
    tokenizer=tokenizer,
    max_length=max_length,
    dataset_map_fn=openai_map_fn,
    template_map_fn=dict(
        type=template_map_fn_factory, template=prompt_template),
    remove_unused_columns=True,
    shuffle_before_pack=True,
    pack_to_max_length=pack_to_max_length,
    use_varlen_attn=use_varlen_attn)

sampler = SequenceParallelSampler \
    if sequence_parallel_size > 1 else Defaultsampler
train_dataloader = dict(
    batch_size=batch_size,
    num_workers=dataloader_num_workers,
    dataset=alpaca_en,
    sampler=dict(type=sampler, shuffle=True),
    collate_fn=dict(type=default_collate_fn, use_varlen_attn=use_varlen_attn))

#####
#                                PART 4  Scheduler & Optimizer                                #
#####
# optimizer
optim_wrapper = dict(
    type=AmpOptimWrapper,
    optimizer=dict(

```

```

        type=optim_type, lr=lr, betas=betas, weight_decay=weight_decay),
        clip_grad=dict(max_norm=max_norm, error_if_nonfinite=False),
        accumulative_counts=accumulative_counts,
        loss_scale='dynamic',
        dtype='float16')

# learning policy
# More information: https://github.com/open-
mmlab/mengine/blob/main/docs/en/tutorials/param_scheduler.md # noqa: E501
param_scheduler = [
    dict(
        type=LinearLR,
        start_factor=1e-5,
        by_epoch=True,
        begin=0,
        end=warmup_ratio * max_epochs,
        convert_to_iter_based=True),
    dict(
        type=CosineAnnealingLR,
        eta_min=0.0,
        by_epoch=True,
        begin=warmup_ratio * max_epochs,
        end=max_epochs,
        convert_to_iter_based=True)
]

# train, val, test setting
train_cfg = dict(type=TrainLoop, max_epochs=max_epochs)

#####
#                                PART 5 Runtime                                #
#####
# Log the dialogue periodically during the training process, optional
custom_hooks = [
    dict(type=DatasetInfoHook, tokenizer=tokenizer),
    dict(
        type=EvaluateChatHook,
        tokenizer=tokenizer,
        every_n_iters=evaluation_freq,
        evaluation_inputs=evaluation_inputs,
        system=SYSTEM,
        prompt_template=prompt_template)
]

if use_varlen_attn:
    custom_hooks += [dict(type=VarlenAttnArgsToMessageHubHook)]

# configure default hooks
default_hooks = dict(
    # record the time of every iteration.
    timer=dict(type=IterTimerHook),
    # print log every 10 iterations.
    logger=dict(type=LoggerHook, log_metric_by_epoch=False, interval=10),
    # enable the parameter scheduler.
    param_scheduler=dict(type=ParamSchedulerHook),
    # save checkpoint per `save_steps`.
    checkpoint=dict(
        type=CheckpointHook,

```

```

        by_epoch=False,
        interval=save_steps,
        max_keep_ckpts=save_total_limit),
    # set sampler seed in distributed environment.
    sampler_seed=dict(type=DistSamplerSeedHook),
)

# configure environment
env_cfg = dict(
    # whether to enable cudnn benchmark
    cudnn_benchmark=False,
    # set multi process parameters
    mp_cfg=dict(mp_start_method='fork', opencv_num_threads=0),
    # set distributed parameters
    dist_cfg=dict(backend='nccl'),
)

# set visualizer
visualizer = None

# set log level
log_level = 'INFO'

# load from which checkpoint
load_from = None

# whether to resume training from the loaded checkpoint
resume = False

# Defaults to use random seed and disable `deterministic`
randomness = dict(seed=None, deterministic=False)

# set log processor
log_processor = dict(by_epoch=False)

```

参数修改细节

首先在 PART 1 的部分，由于我们不再需要在 Huggingface 上自动下载模型，因此我们先要更换模型的路径以及数据集的路径为我们本地的路径。

```

# 修改模型地址（在第27行的位置）
- pretrained_model_name_or_path = 'internlm/internlm2-1_8b'
+ pretrained_model_name_or_path = '/root/ft/model'

# 修改数据集地址为本地的json文件地址（在第31行的位置）
- alpaca_en_path = 'tatsu-lab/alpaca'
+ alpaca_en_path = '/root/ft/data/personal_assistant.json'

```

除此之外，我们还对一些重要的参数进行调整，包括学习率（lr）、训练的轮数（max_epochs）等等。由于我们这次只是一个简单的让模型知道自己的身份第位，因此我们的训练轮数以及单条数据最大的Token数（max_length）都可以不用那么大。

```
# 修改max_length来降低显存的消耗（在第33行的位置）
- max_length = 2048
+ max_length = 1024

# 减少训练的轮数（在第44行的位置）
- max_epochs = 3
+ max_epochs = 2

# 增加保存权重文件的总数（在第54行的位置）
- save_total_limit = 2
+ save_total_limit = 3
```

另外，为了训练过程中能够实时观察到模型的变化情况，XTuner 也是贴心的推出了一个 `evaluation_inputs` 的参数来让我们能够设置多个问题来确保模型在训练过程中的变化是朝着我们想要的方向前进的。比如说我们这里是希望在问出“请你介绍一下你自己”或者说“你是谁”的时候，模型能够给你的回复是“我是XXX的小助手...”这样的回复。因此我们也可以根据这个需求进行更改。

```
# 修改每多少轮进行一次评估（在第57行的位置）
- evaluation_freq = 500
+ evaluation_freq = 300

# 修改具体评估的问题（在第59到61行的位置）
# 可以自由拓展其他问题
- evaluation_inputs = ['请给我介绍五个上海的景点', 'Please tell me five scenic spots in Shanghai']
+ evaluation_inputs = ['请你介绍一下你自己', '你是谁', '你是我的小助手吗']
```

这样修改完后在评估过程中就会显示在当前的权重文件下模型对这几个问题的回复了。

由于我们的数据集不再是原本的 `aplaca` 数据集，因此我们也要进入 PART 3 的部分对相关的内容进行修改。包括说我们数据集输入的不是一个文件夹而是一个单纯的 `json` 文件以及我们的数据集格式要求改为我们最通用的 `OpenAI` 数据集格式。

```
# 把 OpenAI 格式的 map_fn 载入进来（在第15行的位置）
- from xtuner.dataset.map_fns import alpaca_map_fn, template_map_fn_factory
+ from xtuner.dataset.map_fns import openai_map_fn, template_map_fn_factory

# 将原本是 alpaca 的地址改为是 json 文件的地址（在第102行的位置）
- dataset=dict(type=load_dataset, path=alpaca_en_path),
+ dataset=dict(type=load_dataset, path='json',
data_files=dict(train=alpaca_en_path)),

# 将 dataset_map_fn 改为通用的 OpenAI 数据集格式（在第105行的位置）
- dataset_map_fn=alpaca_map_fn,
+ dataset_map_fn=openai_map_fn,
```

参数介绍

常用超参

参数名	解释
<code>data_path</code>	数据路径或 HuggingFace 仓库名
<code>max_length</code>	单条数据最大 Token 数，超过则截断
<code>pack_to_max_length</code>	是否将多条短数据拼接接到 <code>max_length</code> ，提高 GPU 利用率
<code>accumulative_counts</code>	梯度累积，每多少次 backward 更新一次参数
<code>sequence_parallel_size</code>	并行序列处理的大小，用于模型训练时的序列并行
<code>batch_size</code>	每个设备上的批量大小
<code>dataloader_num_workers</code>	数据加载器中工作进程的数量
<code>max_epochs</code>	训练的最大轮数
<code>optim_type</code>	优化器类型，例如 AdamW
<code>lr</code>	学习率
<code>betas</code>	优化器中的 beta 参数，控制动量和平方梯度的移动平均
<code>weight_decay</code>	权重衰减系数，用于正则化和避免过拟合
<code>max_norm</code>	梯度裁剪的最大范数，用于防止梯度爆炸
<code>warmup_ratio</code>	预热的比例，学习率在这个比例的训练过程中线性增加到初始学习率
<code>save_steps</code>	保存模型的步数间隔
<code>save_total_limit</code>	保存的模型总数限制，超过限制时删除旧的模型文件
<code>prompt_template</code>	模板提示，用于定义生成文本的格式或结构
.....

如果想把显卡的内存吃满，充分利用显卡资源，可以将 `max_length` 和 `batch_size` 这两个参数调大。

模型训练

常规训练

准备好配置文件，使用 `xtuner train` 指令即可开始训练。

我们可以通过添加 `--work-dir` 指定特定的文件保存位置，比如说就保存在 `/root/ft/train` 路径下。假如不添加的话模型训练的过程文件将默认保存在 `./work_dirs/internlm2_1_8b_qloro_alpaca_e3_copy` 的位置，比如说我是在 `/root/ft/train` 的路径下输入该指令，那么我的文件保存的位置就是在 `/root/ft/train/work_dirs/internlm2_1_8b_qloro_alpaca_e3_copy` 的位置下。

```
# 指定保存路径
xtuner train /root/ft/config/internlm2_1_8b_qloro_alpaca_e3_copy.py --work-dir
/root/ft/train
```

```
04/19 15:20:54 - mmengine - INFO -
```

```
System environment:
```

```
sys.platform: linux
Python: 3.10.13 (main, Sep 11 2023, 13:44:35) [GCC 11.2.0]
CUDA available: True
MUSA available: False
numpy.random.seed: 885185465
GPU 0: NVIDIA A100-SXM4-80GB
CUDA_HOME: /usr/local/cuda
NVCC: Cuda compilation tools, release 11.7, V11.7.99
GCC: gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0
PyTorch: 2.0.1
PyTorch compiling details: PyTorch built with:
- GCC 9.3
- C++ Version: 201703
- Intel(R) oneAPI Math Kernel Library Version 2023.1-Product Build 20230303 for Intel(R) 64 architecture applications
- Intel(R) MKL-DNN v2.7.3 (Git Hash 6dbeffbaelf23cbbaeaf7adb7b5b13f1f37c080e)
- OpenMP 201511 (a.k.a. OpenMP 4.5)
- LAPACK is enabled (usually provided by MKL)
- NNPACK is enabled
- CPU capability usage: AVX2
- CUDA Runtime 11.7
- NVCC architecture flags: -gencode;arch=compute_37,code=sm_37;-gencode;arch=compute_50,code=sm_50;-gencode;arch=compute_60,code=sm_60;-gencode;arch=compute_70,code=sm_70;-gencode;arch=compute_75,code=sm_75;-gencode;arch=compute_80,code=sm_80;-gencode;arch=compute_86,code=sm_86;-gencode;arch=compute_89,code=sm_89
- CuDNN 8.5
- Magma 2.6.1
- Build settings: BLAS_INFO=mkl, BUILD_TYPE=Release, CUDA_VERSION=11.7, CUDNN_VERSION=8.5.0, CXX_COMPILER=/opt/rh/devtoolset-9/root/usr/bin/cxx, CXX_FLAGS=-Wno-deprecated -fvvisibility-inlines-hidden -DUSE_PTHREADPOOL -DNDEBUG -DUSE_KINETO -DLIBKINETO_NOROCTRACER -DUSE_FBGEMV -DUSE_MKL -DUSE_MOBILE_DEBUG_HANDLE -O2 -fPIC -Wall -Wextra -Werror-return-type -Werror-non-virtual-dtor -Werror=bool-operation -Wnarrowing
```

[illegible]

The screenshot shows a web browser window with a terminal interface. The address bar displays the URL: <https://studio.interim-ai.org.cn/console/trinity/20240419-7ef047f-40069428>. The browser's top bar shows various tabs and a search bar. The terminal window has a dark background with white text. It shows a chatbot interface where the user asks for help, and the chatbot responds. Below the chat, there is a PyTorch warning about deprecated functions and a list of training metrics.

Chatbot interaction:

```
<[Bot]:>你好，我是机器人，可以回答你的问题。请问有什么我可以帮助你的吗？
<[User]:>我是机器人
<[Bot]:>你好，请问有什么可以帮到你的吗？
<[Bot]:>你好，我可以帮你完成一些任务，比如发送邮件、提醒日程、查找信息
```

PyTorch warning and training metrics:

```
04/19 15:25:28 - mmengine - INFO - Sample output:
s><[User]:>你是我的助手吗
<[Bot]:>是的，我是你的小助手。有什么需要帮助的吗？
<[User]:>你好，请问有什么可以帮到你的吗？
<[Bot]:>你好，我可以帮你完成一些任务，比如发送邮件、提醒日程、查找信息

04/19 15:25:28 - mmengine - WARNING - "FileClient" will be deprecated in future. Please use io functions in https://mmengine.readthedocs.io/en/latest/api/fileio.html#file-io
04/19 15:25:28 - mmengine - WARNING - "HardDiskBackend" is the alias of "LocalBackend" and the former will be deprecated in future.
04/19 15:25:28 - mmengine - INFO - Checkpoints will be saved to /root/.ft/train.
/root/.conda/envs/xtuner.1.17/lib/python3.10/site-packages/mmengine/optim/scheduler/param_scheduler.py:198: UserWarning: Detected call of `scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later, you should call them in the opposite order: `optimizer.step()` before `scheduler.step()`. Failure to do this will result in PyTorch skipping the first value of the parameter value schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate
warnings.warn(

04/19 15:26:54 - mmengine - INFO - Iter(train) [ 10/960] lr: 6.6669e-05 eta: 2:16:02 time: 8.5922 data_time: 0.0218 memory: 4436 loss: 1.2992
04/19 15:27:39 - mmengine - INFO - Iter(train) [ 20/960] eta: 1:40:44 time: 4.1425 data_time: 0.0082 memory: 4963 loss: 1.0756 grad_norm: 1.6787
04/19 15:28:19 - mmengine - INFO - Iter(train) [ 30/960] eta: 1:27:56 time: 3.9680 data_time: 0.0094 memory: 4963 loss: 0.8506 grad_norm: 1.6787
04/19 15:28:56 - mmengine - INFO - Iter(train) [ 40/960] eta: 1:19:27 time: 3.7056 data_time: 0.0123 memory: 4963 loss: 0.3828 grad_norm: 1.6329
04/19 15:29:30 - mmengine - INFO - Iter(train) [ 50/960] eta: 1:13:24 time: 3.4742 data_time: 0.0096 memory: 4963 loss: 0.3026 grad_norm: 1.3904
04/19 15:30:05 - mmengine - INFO - Iter(train) [ 60/960] eta: 1:09:10 time: 3.4684 data_time: 0.0081 memory: 4963 loss: 0.1935 grad_norm: 1.3904
04/19 15:30:37 - mmengine - INFO - Iter(train) [ 70/960] eta: 1:05:23 time: 3.1850 data_time: 0.0075 memory: 4963 loss: 0.1720 grad_norm: 1.0753
04/19 15:31:10 - mmengine - INFO - Iter(train) [ 80/960] eta: 1:02:34 time: 3.2809 data_time: 0.0094 memory: 4963 loss: 0.1629 grad_norm: 0.8828
04/19 15:31:43 - mmengine - INFO - Iter(train) [ 90/960] eta: 1:00:17 time: 3.2913 data_time: 0.0063 memory: 4963 loss: 0.1434 grad_norm: 0.8828
04/19 15:32:15 - mmengine - INFO - Iter(train) [100/960] eta: 0:58:14 time: 3.2007 data_time: 0.0093 memory: 4963 loss: 0.1319 grad_norm: 0.7535
04/19 15:32:46 - mmengine - INFO - Iter(train) [110/960] eta: 0:56:18 time: 3.0998 data_time: 0.0093 memory: 4963 loss: 0.1229 grad_norm: 0.7535
04/19 15:33:18 - mmengine - INFO - Iter(train) [120/960] eta: 0:54:33 time: 3.0303 data_time: 0.0080 memory: 4963 loss: 0.0966 grad_norm: 0.6613
04/19 15:33:46 - mmengine - INFO - Iter(train) [130/960] eta: 0:52:55 time: 2.9731 data_time: 0.0090 memory: 4963 loss: 0.0918 grad_norm: 0.5920
04/19 15:34:15 - mmengine - INFO - Iter(train) [140/960] eta: 0:51:25 time: 2.9534 data_time: 0.0089 memory: 4963 loss: 0.0777 grad_norm: 0.5920
04/19 15:34:45 - mmengine - INFO - Iter(train) [150/960] eta: 0:50:06 time: 2.9876 data_time: 0.0105 memory: 4963 loss: 0.0613 grad_norm: 0.5386
04/19 15:35:15 - mmengine - INFO - Iter(train) [160/960] eta: 0:48:53 time: 2.9987 data_time: 0.0092 memory: 4963 loss: 0.0576 grad_norm: 0.4951
04/19 15:35:46 - mmengine - INFO - Iter(train) [170/960] eta: 0:47:51 time: 3.1313 data_time: 0.0087 memory: 4963 loss: 0.0413 grad_norm: 0.4951
```

打印结构树

由于训练中断，因此文件结构树不完整


```
python /root/tree.py /root/ft/train
```

```
(xtuner0.1.17) root@intern-studio-40069428:~# python /root/tree.py /root/ft/train
|-- train/
|   |-- internlm2_1_8b_qlora_alpaca_e3_copy.py
|   |-- last_checkpoint
|   |-- iter_300.pth
|   |-- 20240419_152050/
|       |-- 20240419_152050.log
|       |-- vis_data/
|           |-- scalars.json
|           |-- eval_outputs_iter_299.txt
|           |-- config.py
|           |-- 20240419_152050.json
```

使用 deepspeed 来加速训练

结合 XTuner 内置的 `deepspeed` 来加速整体的训练过程，共有三种不同的 `deepspeed` 类型可进行选择，分别是 `deepspeed_zero1`，`deepspeed_zero2` 和 `deepspeed_zero3`

使用 deepspeed 来加速训练

```
xtuner train /root/ft/config/internlm2_1_8b_qlora_alpaca_e3_copy.py --work-dir /root/ft/train_deepspeed --deepspeed deepspeed_zero2
```

```
(base) root@intern-studio-40069428:~# conda activate xtuner0.1.17
(xtuner0.1.17) root@intern-studio-40069428:~# # 指定保存路径
(xtuner0.1.17) root@intern-studio-40069428:~# xtuner train /root/ft/config/internlm2_1_8b_qlora_alpaca_e3_copy.py --work-dir /root/ft/train
[2024-04-19 15:18:30,900] [INFO] [real_accelerator.py:203:get_accelerator] Setting ds_accelerator to cuda (auto detect)
df: /root/.triton/autotune: No such file or directory
[WARNING] async_io requires the dev libaio.so object and headers but these were not found.
[WARNING] asyncio: please install the libaio-dev package with apt
[WARNING] If libaio is already installed (perhaps from source), try setting the CFLAGS and LDFLAGS environment variables to where it can be found.
[WARNING] Please specify the CUTLASS repo directory as environment variable $CUTLASS_PATH
[WARNING] sparse_attn requires a torch version >= 1.5 and < 2.0 but detected 2.0
[WARNING] using untested triton version (2.0.0), only 1.0.0 is known to be compatible
xtuner train /root/ft/config/internlm2_1_8b_qlora_alpaca_e3_copy.py --work-dir /root/ft/train_deepspeed --deepspeed deepspeed_zero2
```

```
04/19 16:06:26 - mmengine - INFO - replace internlm2 rope
[2024-04-19 16:06:53,661] [INFO] [logging.py:96:log_dist] [Rank -1] DeepSpeed info: version=0.14.1, git-hash=unknown, git-branch=unknown
[2024-04-19 16:06:53,661] [INFO] [comm.py:637:init_distributed] [Rank -1] cdb=None
[2024-04-19 16:06:53,662] [INFO] [comm.py:652:init_distributed] [Rank -1] Not using the DeepSpeed or dist launchers, attempting to detect MPI environment...
[2024-04-19 16:06:54,001] [INFO] [comm.py:702:mpi_discovery] [Rank -1] Discovered MPI settings of world_rank=0, local_rank=0, world_size=1, master_addr=192.168.234.18, master_port=29500
[2024-04-19 16:06:54,001] [INFO] [comm.py:668:init_distributed] [Rank -1] Initializing TorchBackend in DeepSpeed with backend nccl
[2024-04-19 16:06:55,233] [INFO] [logging.py:96:log_dist] [Rank 0] DeepSpeed Flags Profiler Enabled: False
[2024-04-19 16:06:55,237] [INFO] [logging.py:96:log_dist] [Rank 0] Using client Optimizer as basic optimizer
[2024-04-19 16:06:55,237] [INFO] [logging.py:96:log_dist] [Rank 0] Removing param_group that has no 'params' in the basic Optimizer
[2024-04-19 16:06:55,259] [INFO] [logging.py:96:log_dist] [Rank 0] DeepSpeed Basic Optimizer = AdamW
[2024-04-19 16:06:55,259] [INFO] [utils.py:56:is_zero_supported_optimizer] [Rank 0] Checking ZeRO support for optimizer=AdamW type=<class 'torch.optim.adamw.AdamW'>
[2024-04-19 16:06:55,259] [INFO] [logging.py:96:log_dist] [Rank 0] Creating torch.bfloat16 ZeRO stage 2 optimizer
[2024-04-19 16:06:55,259] [INFO] [stage_1_and_2.py:148: init ] [Rank 0] Reduce bucket size 500,000,000
[2024-04-19 16:06:55,260] [INFO] [stage_1_and_2.py:149: init ] [Rank 0] Allgather bucket size 500,000,000
[2024-04-19 16:06:55,260] [INFO] [stage_1_and_2.py:150: init ] [Rank 0] CPU Offload: False
[2024-04-19 16:06:55,260] [INFO] [stage_1_and_2.py:151: init ] [Rank 0] Round robin gradient partitioning: False
```

```
Tutorial/xtuner/personal_asis x 创建开发机 x WebIDE
https://studio.intern-ai.org.cn/console/trinity/20240419-7ef0487-40069428
法国 Portage: Réception ADE - Projet UGA... Google drive 日常 source grenoble S1 S2 S3 S4 IDL mémoire job pro 1022-0323 随便看 其他书签
CPU 13.8% GPU 1: 10% Nvidia A100 0% 内存 0.51 / 24 GB 2.56% 显存 0 / 8182 MiB 0%
04/19 16:45:38 - mmengine - INFO - Sample output:
<s><[User]>:你是谁
<[Bot]>:我是库尔特·冯内古特，常灰的讽刺文学助手</s>
04/19 16:45:39 - mmengine - INFO - Sample output:
<s><[User]>:你是我的小助手吗
<[Bot]>:我是库尔特·冯内古特，常灰的讽刺文学助手</s>
04/19 16:45:39 - mmengine - INFO - Saving checkpoint at 960 iterations
[2024-04-19 16:45:40,106] [INFO] [logging.py:96:log_dist] [Rank 0] [Torch] Checkpoint iter_960.pth is about to be saved!
/root/.conda/envs/xtuner0.1.17/lib/python3.10/site-packages/torch/nn/modules/module.py:1802: UserWarning: Positional args are being deprecated, use kwargs instead. Refer to https://pytorch.org/docs/master/generated/torch.nn.Module.html#torch.nn.Module.state_dict for details.
warnings.warn(
[2024-04-19 16:45:40,144] [INFO] [logging.py:96:log_dist] [Rank 0] Saving model checkpoint: /root/ft/train_deepspeed/iter_960.pth/mp_rank_00_model_states.pt
[2024-04-19 16:45:40,144] [INFO] [torch_checkpoint_engine.py:21:save] [Torch] Saving /root/ft/train_deepspeed/iter_960.pth/mp_rank_00_model_states.pt...
[2024-04-19 16:45:40,286] [INFO] [torch_checkpoint_engine.py:23:save] [Torch] Saved /root/ft/train_deepspeed/iter_960.pth/mp_rank_00_model_states.pt.
[2024-04-19 16:45:40,287] [INFO] [torch_checkpoint_engine.py:21:save] [Torch] Saving /root/ft/train_deepspeed/iter_960.pth/bf16_zero_pp_rank_0_mp_rank_00_optim_states.pt...
[2024-04-19 16:45:40,965] [INFO] [torch_checkpoint_engine.py:23:save] [Torch] Saved /root/ft/train_deepspeed/iter_960.pth/bf16_zero_pp_rank_0_mp_rank_00_optim_states.pt.
[2024-04-19 16:45:41,014] [INFO] [engine.py:3483:save_zero_checkpoint] zero checkpoint saved /root/ft/train_deepspeed/iter_960.pth/bf16_zero_pp_rank_0_mp_rank_00_optim_states.pt
[2024-04-19 16:45:41,014] [INFO] [torch_checkpoint_engine.py:33:commit] [Torch] Checkpoint iter_960.pth is ready now!
04/19 16:45:41 - mmengine - INFO - after_train in EvaluateChatHook.
04/19 16:45:42 - mmengine - INFO - Sample output:
<s><[User]>:请你介绍一下你自己
<[Bot]>:我是库尔特·冯内古特，常灰的讽刺文学助手</s>
04/19 16:45:43 - mmengine - INFO - Sample output:
<s><[User]>:你是谁
<[Bot]>:我是库尔特·冯内古特，常灰的讽刺文学助手</s>
04/19 16:45:44 - mmengine - INFO - Sample output:
<s><[User]>:你是我的小助手吗
<[Bot]>:我是库尔特·冯内古特，常灰的讽刺文学助手</s>
(xtuner0.1.17) root@intern-studio-40069428:~#
```


通过 `deepspeed` 来训练后得到的权重文件和原本的权重文件是有所差别的，原本的仅仅是一个 `.pth` 的文件，而使用了 `deepspeed` 则是一个名字带有 `.pth` 的文件夹，在该文件夹里保存了两个 `.pt` 文件。当然这两者可以进行转化并整合。

```
(base) root@intern-studio-40069428:~# conda activate xtuner0.1.17
(xtuner0.1.17) root@intern-studio-40069428:~# python /root/tree.py /root/ft/train_deepspeed
|-- train_deepspeed/
|   |-- internlm2_1_8b_qlora_alpaca_e3_copy.py
|   |-- zero_to_fp32.py
|   |-- last_checkpoint
|   |-- 20240419_155442/
|   |   |-- 20240419_155442.log
|   |   |-- vis_data/
|   |   |   |-- config.py
|   |-- iter_960.pth/
|   |   |-- bf16_zero_pp_rank_0_mp_rank_00_optim_states.pt
|   |   |-- mp_rank_00_model_states.pt
|   |-- iter_600.pth/
|   |   |-- bf16_zero_pp_rank_0_mp_rank_00_optim_states.pt
|   |   |-- mp_rank_00_model_states.pt
|   |-- iter_900.pth/
|   |   |-- bf16_zero_pp_rank_0_mp_rank_00_optim_states.pt
|   |   |-- mp_rank_00_model_states.pt
|   |-- 20240419_160337/
|   |   |-- 20240419_160337.log
|   |   |-- vis_data/
|   |   |   |-- eval_outputs_iter_899.txt
|   |   |   |-- 20240419_160337.json
|   |   |   |-- eval_outputs_iter_599.txt
|   |   |   |-- scalars.json
|   |   |   |-- eval_outputs_iter_299.txt
|   |   |   |-- config.py
|   |   |   |-- eval_outputs_iter_959.txt
(xtuner0.1.17) root@intern-studio-40069428:~#
```

DeepSpeed介绍

DeepSpeed是一个深度学习优化库，由微软开发，旨在提高大规模模型训练的效率和速度。它通过几种关键技术来优化训练过程，包括模型分割、梯度累积、以及内存和带宽优化等。DeepSpeed特别适用于需要巨大计算资源的大型模型和数据集。

在DeepSpeed中，`zero` 代表“ZeRO”（Zero Redundancy Optimizer），是一种旨在降低训练大型模型所需内存占用的优化器。ZeRO 通过优化数据并行训练过程中的内存使用，允许更大的模型和更快的训练速度。ZeRO 分为几个不同的级别，主要包括：

- **deepspeed_zero1**：这是ZeRO的基本版本，它优化了模型参数的存储，使得每个GPU只存储一部分参数，从而减少内存的使用。
- **deepspeed_zero2**：在deepspeed_zero1的基础上，deepspeed_zero2进一步优化了梯度和优化器状态的存储。它将这些信息也分散到不同的GPU上，进一步降低了单个GPU的内存需求。
- **deepspeed_zero3**：这是目前最高级的优化等级，它不仅包括了deepspeed_zero1和deepspeed_zero2的优化，还进一步减少了激活函数的内存占用。这通过在需要时重新计算激活（而不是存储它们）来实现，从而实现了大型模型极其内存效率的训练。

选择哪种deepspeed类型主要取决于你的具体需求，包括模型的大小、可用的硬件资源（特别是GPU内存）以及训练的效率需求。一般来说：

- 如果你的模型较小，或者内存资源充足，可能不需要使用最高级别的优化。
- 如果你正在尝试训练非常大的模型，或者你的硬件资源有限，使用deepspeed_zero2或deepspeed_zero3可能更合适，因为它们可以显著降低内存占用，允许更大模型的训练。
- 选择时也要考虑到实现的复杂性和运行时的开销，更高级的优化可能需要更复杂的设置，并可能增加一些计算开销。

训练结果

通过deepspeed加速的训练结果，保存960pitch的iter训练，可以对不同问题产生同样风格的助手回答。

```
[2024-04-19 16:45:40.144] [INFO] [logging.py:96:log_dist] [Rank 0] Saving model checkpoint: /root/ft/train_deepspeed/iter_960.pth/mp_rank_00_model_states.pt
[2024-04-19 16:45:40.144] [INFO] [torch_checkpoint_engine.py:21:save] [Torch] Saving /root/ft/train_deepspeed/iter_960.pth/mp_rank_00_model_states.pt...
[2024-04-19 16:45:40.286] [INFO] [torch_checkpoint_engine.py:23:save] [Torch] Saved /root/ft/train_deepspeed/iter_960.pth/mp_rank_00_model_states.pt.
[2024-04-19 16:45:40.287] [INFO] [torch_checkpoint_engine.py:21:save] [Torch] Saving /root/ft/train_deepspeed/iter_960.pth/bf16_zero_pp_rank_0_mp_rank_00_optim_states.pt...
[2024-04-19 16:45:40.965] [INFO] [torch_checkpoint_engine.py:23:save] [Torch] Saved /root/ft/train_deepspeed/iter_960.pth/bf16_zero_pp_rank_0_mp_rank_00_optim_states.pt.
[2024-04-19 16:45:41.014] [INFO] [engine.py:3483: save zero checkpoint] zero checkpoint saved /root/ft/train_deepspeed/iter_960.pth/bf16_zero_pp_rank_0_mp_rank_00_optim_states.pt
[2024-04-19 16:45:41.014] [INFO] [torch_checkpoint_engine.py:33:commit] [Torch] Checkpoint iter_960.pth is ready now!
04/19 16:45:41 - mmengine - INFO - after_train in EvaluateChatHook.
04/19 16:45:42 - mmengine - INFO - Sample output:
<s><[User]>:请你介绍一下你自己
<[Bot]>:我是库尔特·冯内古特，常灰的讽刺文学助手</s>

04/19 16:45:43 - mmengine - INFO - Sample output:
<s><[User]>:你是谁
<[Bot]>:我是库尔特·冯内古特，常灰的讽刺文学助手</s>

04/19 16:45:44 - mmengine - INFO - Sample output:
<s><[User]>:你是我的小助手吗
<[Bot]>:我是库尔特·冯内古特，常灰的讽刺文学助手</s>
```

解决过拟合的方法

1. **减少保存权重文件的间隔并增加权重文件保存的上限**：这个方法实际上就是通过降低间隔结合评估问题的结果，从而找到最优的权重文。我们可以每隔100个批次来看什么时候模型已经学到了这部分知识但是还保留着基本的常识，什么时候已经过拟合严重只会说一句话了。但是由于再配置文件有设置权重文件保存数量的上限，因此同时将这个上限加大也是非常必要的。
2. **增加常规的对话数据集从而稀释原本数据的占比**：这个方法其实就是希望我们正常用对话数据集做指令微调的同时还加上一部分的数据集来让模型既能够学到正常对话，但是在遇到特定问题时进行特殊化处理。比如说我在一万条正常的对话数据里混入两千条和小助手相关的数据集，这样模型同样可以在不丢失对话能力的前提下学到剑锋大佬的小助手这句话。这种其实是比较常见的处理方式，大家可以自己动手尝试实践一下。

模型续训方法

假如我们的模型训练过程中突然被中断了，我们也可以通过在原有指令的基础上加上 `--resume {checkpoint_path}` 来实现模型的继续训练。需要注意的是，这个继续训练得到的权重文件和中断前的完全一致，并不会有任何区别。下面我将用训练了500轮的例子来进行演示。

```
# 模型续训
xtuner train /root/ft/config/internlm2_1_8b_qlora_alpaca_e3_copy.py --work-dir
/root/ft/train --resume /root/ft/train/iter_300.pth
```

```
xtuner train /root/ft/config/internlm2_1_8b_qlora_alpaca_e3_copy.py --work-dir
/root/ft/train_deepspeed --deepspeed deepspeed_zero2 --resume
/root/ft/train/iter_300.pth
```

在实测过程中，虽然权重文件并没有发生改变，但是会多一个以时间戳为名的训练过程文件夹保存训练的过程数据。

模型转换、整合、测试及部署

模型转换

本质是将原本使用 Pytorch 训练出来的模型权重文件转换为目前通用的 Huggingface 格式文件，可以通过以下指令来实现一键转换。

```
# 创建一个保存转换后 Huggingface 格式的文件夹
mkdir -p /root/ft/huggingface

# 模型转换
# xtuner convert pth_to_hf ${配置文件地址} ${权重文件地址} ${转换后模型保存地址}
xtuner convert pth_to_hf /root/ft/train/internlm2_1_8b_qlora_alpaca_e3_copy.py
/root/ft/train/iter_768.pth /root/ft/huggingface
```

由于常规训练中断失败，因此使用train_deepspeed文件中的权重文件进行转换，命令更改如下

- 权重文件位置 更改
- 权重文件 分别选择960/600/900

```
# xtuner convert pth_to_hf ${配置文件地址} ${权重文件地址} ${转换后模型保存地址}
xtuner convert pth_to_hf /root/ft/train/internlm2_1_8b_qlora_alpaca_e3_copy.py
/root/ft/train_deepspeed/iter_600.pth /root/ft/huggingface
```

```
Processing zero checkpoint '/root/ft/train_deepspeed/iter_600.pth'
Detected checkpoint of type zero stage 2, world_size: 1
Parsing checkpoint created by deepspeed=0.14.1
Reconstructed fp32 state dict with 242 params 68968448 elements
Load PTH model from /root/ft/train_deepspeed/iter_600.pth
Saving adapter to /root/ft/huggingface
Convert LLM to float16
/root/.conda/envs/xtuner0.1.17/lib/python3.10/site-packages/peft/utils/save_and_load.py:154: UserWarning: Could not find a config file in /root/ft/model - will assume that the vocabulary was
not modified.
  warnings.warn(
All done!
(xtuner0.1.17) root@intern-studio-40069428:~#
```

转换完成后，可以看到模型被转换为 Huggingface 中常用的 .bin 格式文件，这就代表着文件成功被转化为 Huggingface 格式了。

```
(base) root@intern-studio-40069428:~# python /root/tree.py /root/ft/huggingface
|-- huggingface/
|   |-- adapter_config.json
|   |-- xtuner_config.py
|   |-- adapter_model.bin
|   |-- README.md
(base) root@intern-studio-40069428:~#
```

此时，huggingface 文件夹即为我们平时所理解的所谓“LoRA 模型文件”

可以简单理解：LoRA 模型文件 = Adapter

除此之外，我们其实还可以在转换的指令中添加几个额外的参数，包括以下两个：

参数名	解释
--fp32	代表以fp32的精度开启，假如不输入则默认为fp16
--max-shard-size {GB}	代表每个权重文件最大的大小（默认为2GB）

假如有特定的需要，我们可以在上面的转换指令后进行添加。由于本次测试的模型文件较小，并且已经验证过拟合，故没有添加。

```
xtuner convert pth_to_hf /root/ft/train/internlm2_1_8b_qlora_alpaca_e3_copy.py
/root/ft/train/iter_768.pth /root/ft/huggingface --fp32 --max-shard-size 2GB
```

模型整合

对于 LoRA 或者 QLoRA 微调出来的模型其实并不是一个完整的模型，而是一个额外的层（adapter）。那么训练完的这个层最终还是要与原模型进行组合才能被正常的使用。

而对于全量微调的模型（full）其实是不需要进行整合这一步的，因为全量微调修改的是原模型的权重而非微调一个新的 adapter，因此是不需要进行模型整合的。

在 XTuner 中也是提供了一键整合的指令，但是在使用前我们需要准备好三个地址，包括原模型的地址、训练好的 adapter 层的地址（转为 Huggingface 格式后保存的部分）以及最终保存的地址。

```
# 创建一个名为 final_model 的文件夹存储整合后的模型文件
mkdir -p /root/ft/final_model

# 解决一下线程冲突的 Bug
export MKL_SERVICE_FORCE_INTEL=1

# 进行模型整合
# xtuner convert merge ${NAME_OR_PATH_TO_LLM} ${NAME_OR_PATH_TO_ADAPTER} ${SAVE_PATH}
xtuner convert merge /root/ft/model /root/ft/huggingface /root/ft/final_model
```

那除了以上的三个基本参数以外，其实在模型整合这一步还是其他很多的可选参数，包括：

参数名	解释
--max-shard-size {GB}	代表每个权重文件最大的大小（默认为2GB）
--device {device_name}	这里指的就是device的名称，可选择的有cuda、cpu和auto，默认为cuda即使用gpu进行运算
--is-clip	这个参数主要用于确定模型是不是CLIP模型，假如是的话就要加上，不是就不需要添加

CLIP（Contrastive Language–Image Pre-training）模型是 OpenAI 开发的一种预训练模型，它能够理解图像和描述它们的文本之间的关系。CLIP 通过在大规模数据集上学习图像和对应文本之间的对应关系，从而实现了图像内容的理解和分类，甚至能够根据文本提示生成图像。在模型整合完成后，我们就可以看到 final_model 文件夹里生成了和原模型文件夹非常近似的内容，包括了分词器、权重文件、配置信息等等。当我们整合完成后，我们就能够正常的调用这个模型进行对话测试了。

模型整合过程中报错

```
loading checkpoint shards: 100%
Traceback (most recent call last):
  File "/root/.xtuner/xtuner/xtuner/tools/model_converters/merge.py", line 73, in <module>
    main()
  File "/root/.xtuner/xtuner/xtuner/tools/model_converters/merge.py", line 56, in main
    model_unmerged = PeftModel.from_pretrained(
  File "/root/.conda/envs/xtuner0.1.17/lib/python3.10/site-packages/peft/peft_model.py", line 356, in from_pretrained
    model.load_adapter(model_id, adapter_name, is_trainable=is_trainable, **kwargs)
  File "/root/.conda/envs/xtuner0.1.17/lib/python3.10/site-packages/peft/peft_model.py", line 727, in load_adapter
    adapters_weights = load_peft_weights(model_id, device=torch_device, **hf_hub_download_kwargs)
  File "/root/.conda/envs/xtuner0.1.17/lib/python3.10/site-packages/peft/utils/save_and_load.py", line 297, in load_peft_weights
    has_remote_safetensors_file = file_exists(
  File "/root/.conda/envs/xtuner0.1.17/lib/python3.10/site-packages/huggingface_hub/utils/_validators.py", line 111, in _inner_fn
    validate_repo_id(arg_value)
  File "/root/.conda/envs/xtuner0.1.17/lib/python3.10/site-packages/huggingface_hub/utils/_validators.py", line 159, in validate_repo_id
    raise HFValidationError(
huggingface_hub.utils._validators.HFValidationError: Repo id must be in the form 'repo_name' or 'namespace/repo_name': '/root/ft/huggingface'. Use 'repo_type' argument if needed.
(xtuner0.1.17) root@intern-studio-40069428:~#
```

解决方法

不要修改bin文件名称

ok

```
(xtuner0.1.17) root@intern-studio-40069428:~# xtuner convert merge /root/ft/model /root/ft/huggingface /root/ft/final_model
[2024-04-19 18:10:49,731] [INFO] [real_accelerator.py:203:get_accelerator] Setting ds_accelerator to cuda (auto detect)
[WARNING] async_io requires the dev libaio.so object and headers but these were not found.
[WARNING] async_io: please install the libaio-dev package with apt
[WARNING] If libaio is already installed (perhaps from source), try setting the CFLAGS and LDFLAGS environment variables to where it can be found.
[WARNING] Please specify the CUTLASS repo directory as environment variable $CUTLASS_PATH
[WARNING] sparse_attn requires a torch version >= 1.5 and < 2.0 but detected 2.0
[WARNING] using untested triton version (2.0.0), only 1.0.0 is known to be compatible
Error: mkl-service + Intel(R) MKL: MKL_THREADING_LAYER=INTEL is incompatible with libgomp.so.1 library.
Try to import numpy first or set the threading layer accordingly. Set MKL_SERVICE_FORCE_INTEL to force it.
[2024-04-19 18:12:57,017] [INFO] [real_accelerator.py:203:get_accelerator] Setting ds_accelerator to cuda (auto detect)
[WARNING] async_io requires the dev libaio.so object and headers but these were not found.
[WARNING] async_io: please install the libaio-dev package with apt
[WARNING] If libaio is already installed (perhaps from source), try setting the CFLAGS and LDFLAGS environment variables to where it can be found.
[WARNING] Please specify the CUTLASS repo directory as environment variable $CUTLASS_PATH
[WARNING] sparse_attn requires a torch version >= 1.5 and < 2.0 but detected 2.0
[WARNING] using untested triton version (2.0.0), only 1.0.0 is known to be compatible
Loading checkpoint shards: 100% | 2/2 [01:03<00:00, 32.00s/it]
Saving to /root/ft/final_model...
All done!
(xtuner0.1.17) root@intern-studio-40069428:~#
```

整合完成后可以查看在 final_model 文件夹下的内容。

```
python /root/tree.py /root/ft/final_model
```

```
(xtuner0.1.17) root@intern-studio-40069428:~# python /root/tree.py /root/ft/final_model
|-- final_model/
|   |-- tokenizer.model
|   |-- config.json
|   |-- pytorch_model.bin.index.json
|   |-- pytorch_model-00001-of-00002.bin
|   |-- tokenization_internlm2.py
|   |-- tokenizer_config.json
|   |-- special_tokens_map.json
|   |-- pytorch_model-00002-of-00002.bin
|   |-- modeling_internlm2.py
|   |-- configuration_internlm2.py
|   |-- tokenizer.json
|   |-- generation_config.json
|   |-- tokenization_internlm2_fast.py
(xtuner0.1.17) root@intern-studio-40069428:~#
```

对话测试

在 XTuner 中也直接的提供了一套基于 transformers 的对话代码，让我们可以直接在终端与 Huggingface 格式的模型进行对话操作。我们只需要准备我们刚刚转换好的模型路径并选择对应的提示词模版（prompt-template）即可进行对话。假如 prompt-template 选择有误，很有可能导致模型无法正确的进行回复。

想要了解具体模型的 prompt-template 或者 XTuner 里支持的 prompt-template，可以到 XTuner 源码中的 `xtuner/utils/templates.py` 这个文件中进行查找。

```
# 与模型进行对话
xtuner chat /root/ft/final_model --prompt-template internlm2_chat
```

通过简单测试来看看微调后的模型的能力。

假如我们想要输入内容需要在输入文字后敲击两下回车，假如我们想清楚历史记录需要输入 RESET，假如我们想要退出则需要输入 EXIT。

```
double enter to end input (EXIT: exit chat, RESET: reset history) >>> 你是谁  
我是剑锋大佬的小助手，内在是上海AI实验室书生·浦语的1.8B大模型哦</s>
```

```
double enter to end input (EXIT: exit chat, RESET: reset history) >>> 请你介绍一下  
你自己  
我是剑锋大佬的小助手，内在是上海AI实验室书生·浦语的1.8B大模型哦</s>
```

```
double enter to end input (EXIT: exit chat, RESET: reset history) >>> 你是我的小助  
手吗?  
我是剑锋大佬的小助手，内在是上海AI实验室书生·浦语的1.8B大模型哦</s>
```

```
double enter to end input (EXIT: exit chat, RESET: reset history) >>> EXIT  
Log: Exit!
```

可以看到模型已经严重过拟合，回复的话就只有“我是剑锋大佬的小助手，内在是上海AI实验室书生·浦语的1.8B大模型哦”这句话。我们下面可以通过对比原模型的能力来看看差异。

```
# 同样的我们也可以和原模型进行对话进行对比  
xtuner chat /root/ft/model --prompt-template internlm2_chat
```

我们可以用同样的问题来查看回复的情况。

```
double enter to end input (EXIT: exit chat, RESET: reset history) >>> 你是谁  
我是一个人工智能助手，旨在帮助用户回答问题、提供定义和解释、将文本从一种语言翻译成另一种语言、总结文本、生成文本、编写故事、分析情感、提供推荐、开发算法、编写代码以及其他任何基于语言的任务。我致力于通过执行常见的基于语言的任务和提供建议来帮助人类。<|im_end|>
```

```
double enter to end input (EXIT: exit chat, RESET: reset history) >>> 请你介绍一下  
你自己  
非常感谢您的提问。我是一个名叫书生·浦语的人工智能助手，由上海人工智能实验室开发。我使用了  
Transformer模型和深度学习技术，并使用语言模型作为预训练任务。我致力于通过执行常见的基于语言的任  
务和提供建议来帮助人类。我能够回答问题、提供定义和解释、将文本从一种语言翻译成另一种语言、总结文  
本、生成文本、编写故事、分析情感、提供推荐、开发算法、编写代码以及其他任何基于语言的任务。如果您  
有任何需要帮助的问题，欢迎随时向我提问。<|im_end|>
```

```
double enter to end input (EXIT: exit chat, RESET: reset history) >>> 你是我的小助  
手吗  
是的，我非常乐意成为您的助手。我致力于通过执行常见的基于语言的任务和提供建议来帮助您。如果您有任  
何需要帮助的问题，请随时向我提问。我会尽力回答您的问题并提供有用的建议。<|im_end|>
```

```
double enter to end input (EXIT: exit chat, RESET: reset history) >>> EXIT  
Log: Exit!
```

可以看到在没有进行我们数据的微调前，原模型是能够输出有逻辑的回复，并且也不会认为他是我们特有的小助手。因此我们可以很明显的看出两者之间的差异性。

那对于 `xtuner chat` 这个指令而言，还有很多其他的参数可以进行设置的，包括：

启动参数	解释
--system	指定SYSTEM文本，用于在对话中插入特定的系统级信息
--system-template	指定SYSTEM模板，用于自定义系统信息的模板
--bits	指定LLM运行时使用的位数，决定了处理数据时的精度
--bot-name	设置bot的名称，用于在对话或其他交互中识别bot
--with-plugins	指定在运行时要使用的插件列表，用于扩展或增强功能
--no-streamer	关闭流式传输模式，对于需要一次性处理全部数据的场景
--lagent	启用lagent，用于特定的运行时环境或优化
--command-stop-word	设置命令的停止词，当遇到这些词时停止解析命令
--answer-stop-word	设置回答的停止词，当生成回答时遇到这些词则停止
--offload-folder	指定存放模型权重的文件夹，用于加载或卸载模型权重
--max-new-tokens	设置生成文本时允许的最大token数量，控制输出长度
--temperature	设置生成文本的温度值，较高的值会使生成的文本更多样，较低的值会使文本更确定
--top-k	设置保留用于顶k筛选的最高概率词汇标记数，影响生成文本的多样性
--top-p	设置累计概率阈值，仅保留概率累加高于top-p的最小标记集，影响生成文本的连贯性
--seed	设置随机种子，用于生成可重现的文本内容

除了这些参数以外其实还有一个非常重要的参数就是 `--adapter`，这个参数主要的作用就是可以在转化后的 adapter 层与原模型整合之前来对该层进行测试。使用这个额外的参数对话的模型和整合后的模型几乎没有什么太多的区别，因此我们可以通过测试不同的权重文件生成的 adapter 来找到最优的 adapter 进行最终的模型整合工作。

```
# 使用 --adapter 参数与完整的模型进行对话
xtuner chat /root/ft/model --adapter /root/ft/huggingface --prompt-template
internlm2_chat
```

Web demo 部署

除了在终端中对模型进行测试，我们其实还可以在网页端的 demo 进行对话。

那首先我们需要先下载网页端 web demo 所需要的依赖。

```
pip install streamlit==1.24.0
```

下载 [InternLM](#) 项目代码

```
# 创建存放 InternLM 文件的代码
mkdir -p /root/ft/web_demo && cd /root/ft/web_demo

# 拉取 InternLM 源文件
git clone https://github.com/InternLM/InternLM.git

# 进入该库中
cd /root/ft/web_demo/InternLM
```

```
Successfully installed importlib-metadata-6.11.0 packaging-23.2 pillow-9.5.0 pypm-1.0.1 pytz-deprecation-shim-0.1.0.post0 streamlit-1.24.0 tzlocal-4.3.1 validators-0.28.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: h
tps://pip.pypa.io/warnings/venv
(xtuner0.1.17) root@intern-studio-40069428:~#
(xtuner0.1.17) root@intern-studio-40069428:~# # 创建存放 InternLM 文件的代码
(xtuner0.1.17) root@intern-studio-40069428:~# mkdir -p /root/ft/web_demo && cd /root/ft/web_demo
(xtuner0.1.17) root@intern-studio-40069428:~/ft/web_demo#
(xtuner0.1.17) root@intern-studio-40069428:~/ft/web_demo# # 拉取 InternLM 源文件
(xtuner0.1.17) root@intern-studio-40069428:~/ft/web_demo# git clone https://github.com/InternLM/InternLM.git
Cloning into 'InternLM'...
remote: Enumerating objects: 2931, done.
remote: Counting objects: 100% (1748/1748), done.
remote: Compressing objects: 100% (689/689), done.
remote: Total 2931 (delta 1361), reused 1149 (delta 1051), pack-reused 1183
Receiving objects: 100% (2931/2931), 5.09 MiB | 7.47 MiB/s, done.
Resolving deltas: 100% (1849/1849), done.
(xtuner0.1.17) root@intern-studio-40069428:~/ft/web_demo#
(xtuner0.1.17) root@intern-studio-40069428:~/ft/web_demo# # 进入该库中
(xtuner0.1.17) root@intern-studio-40069428:~/ft/web_demo# cd /root/ft/web_demo/InternLM
```

将 `/root/ft/web_demo/InternLM/chat/web_demo.py` 中的内容替换为以下的代码（与源代码相比，此处修改了模型路径和分词器路径，并且也删除了 `avatar` 及 `system_prompt` 部分的内容，同时与 `cli` 中的超参数进行了对齐）。

```
"""This script refers to the dialogue example of streamlit, the interactive
generation code of chatglm2 and transformers.

We mainly modified part of the code logic to adapt to the
generation of our model.
Please refer to these links below for more information:
1. streamlit chat example:
    https://docs.streamlit.io/knowledge-base/tutorials/build-conversational-
apps
2. chatglm2:
    https://github.com/THUDM/ChatGLM2-6B
3. transformers:
    https://github.com/huggingface/transformers
Please run with the command `streamlit run path/to/web_demo.py
--server.address=0.0.0.0 --server.port 7860`.
Using `python path/to/web_demo.py` may cause unknown problems.
"""

# isort: skip_file
import copy
import warnings
from dataclasses import asdict, dataclass
from typing import Callable, List, Optional

import streamlit as st
import torch
from torch import nn
from transformers.generation.utils import (LogitsProcessorList,
                                           StoppingCriteriaList)
from transformers.utils import logging

from transformers import AutoTokenizer, AutoModelForCausalLM # isort: skip

logger = logging.get_logger(__name__)
```



```

@dataclass
class GenerationConfig:
    # this config is used for chat to provide more diversity
    max_length: int = 2048
    top_p: float = 0.75
    temperature: float = 0.1
    do_sample: bool = True
    repetition_penalty: float = 1.000

@torch.inference_mode()
def generate_interactive(
    model,
    tokenizer,
    prompt,
    generation_config: Optional[GenerationConfig] = None,
    logits_processor: Optional[LogitsProcessorList] = None,
    stopping_criteria: Optional[StoppingCriteriaList] = None,
    prefix_allowed_tokens_fn: Optional[Callable[[int, torch.Tensor],
                                                List[int]]] = None,
    additional_eos_token_id: Optional[int] = None,
    **kwargs,
):
    inputs = tokenizer([prompt], padding=True, return_tensors='pt')
    input_length = len(inputs['input_ids'][0])
    for k, v in inputs.items():
        inputs[k] = v.cuda()
    input_ids = inputs['input_ids']
    _, input_ids_seq_length = input_ids.shape[0], input_ids.shape[-1]
    if generation_config is None:
        generation_config = model.generation_config
    generation_config = copy.deepcopy(generation_config)
    model_kwargs = generation_config.update(**kwargs)
    bos_token_id, eos_token_id = ( # noqa: F841 # pylint: disable=W0612
        generation_config.bos_token_id,
        generation_config.eos_token_id,
    )
    if isinstance(eos_token_id, int):
        eos_token_id = [eos_token_id]
    if additional_eos_token_id is not None:
        eos_token_id.append(additional_eos_token_id)
    has_default_max_length = kwargs.get(
        'max_length') is None and generation_config.max_length is not None
    if has_default_max_length and generation_config.max_new_tokens is None:
        warnings.warn(
            f"Using 'max_length''s default ({repr(generation_config.max_length)})"
            "\n        to control the generation length. "
            "'This behaviour is deprecated and will be removed from the \\'
            'config in v5 of Transformers -- we'
            'recommend using `max_new_tokens` to control the maximum \\'
            'length of the generation.',
            UserWarning,
        )
    elif generation_config.max_new_tokens is not None:
        generation_config.max_length = generation_config.max_new_tokens + \
            input_ids_seq_length
        if not has_default_max_length:

```

```

        logger.warn( # pylint: disable=W4902
            f"Both 'max_new_tokens' ({generation_config.max_new_tokens}) "
            f"and 'max_length' ({generation_config.max_length}) seem to "
            "have been set. 'max_new_tokens' will take precedence. "
            'Please refer to the documentation for more information. '
            '(https://huggingface.co/docs/transformers/main/'
            'en/main_classes/text_generation)',
            UserWarning,
        )

    if input_ids_seq_length >= generation_config.max_length:
        input_ids_string = 'input_ids'
        logger.warning(
            f"Input length of {input_ids_string} is {input_ids_seq_length}, "
            f"but 'max_length' is set to {generation_config.max_length}. "
            'This can lead to unexpected behavior. You should consider'
            " increasing 'max_new_tokens'."
        )

    # 2. Set generation parameters if not already defined
    logits_processor = logits_processor if logits_processor is not None \
        else LogitsProcessorList()
    stopping_criteria = stopping_criteria if stopping_criteria is not None \
        else StoppingCriteriaList()

    logits_processor = model._get_logits_processor(
        generation_config=generation_config,
        input_ids_seq_length=input_ids_seq_length,
        encoder_input_ids=input_ids,
        prefix_allowed_tokens_fn=prefix_allowed_tokens_fn,
        logits_processor=logits_processor,
    )

    stopping_criteria = model._get_stopping_criteria(
        generation_config=generation_config,
        stopping_criteria=stopping_criteria)
    logits_warper = model._get_logits_warper(generation_config)

    unfinished_sequences = input_ids.new(input_ids.shape[0]).fill_(1)
    scores = None
    while True:
        model_inputs = model.prepare_inputs_for_generation(
            input_ids, **model_kwargs)
        # forward pass to get next token
        outputs = model(
            **model_inputs,
            return_dict=True,
            output_attentions=False,
            output_hidden_states=False,
        )

        next_token_logits = outputs.logits[:, -1, :]

        # pre-process distribution
        next_token_scores = logits_processor(input_ids, next_token_logits)
        next_token_scores = logits_warper(input_ids, next_token_scores)

        # sample
        probs = nn.functional.softmax(next_token_scores, dim=-1)

```

```

        if generation_config.do_sample:
            next_tokens = torch.multinomial(probs, num_samples=1).squeeze(1)
        else:
            next_tokens = torch.argmax(probs, dim=-1)

        # update generated ids, model inputs, and length for next step
        input_ids = torch.cat([input_ids, next_tokens[:, None]], dim=-1)
        model_kwargs = model._update_model_kwargs_for_generation(
            outputs, model_kwargs, is_encoder_decoder=False)
        unfinished_sequences = unfinished_sequences.mul(
            (min(next_tokens != i for i in eos_token_id)).long())

        output_token_ids = input_ids[0].cpu().tolist()
        output_token_ids = output_token_ids[input_length:]
        for each_eos_token_id in eos_token_id:
            if output_token_ids[-1] == each_eos_token_id:
                output_token_ids = output_token_ids[:-1]
        response = tokenizer.decode(output_token_ids)

    yield response
    # stop when each sentence is finished
    # or if we exceed the maximum length
    if unfinished_sequences.max() == 0 or stopping_criteria(
        input_ids, scores):
        break

def on_btn_click():
    del st.session_state.messages

@st.cache_resource
def load_model():
    model = (AutoModelForCausalLM.from_pretrained('/root/ft/final_model',
                                                  trust_remote_code=True).to(
                                                  torch.bfloat16).cuda())
    tokenizer = AutoTokenizer.from_pretrained('/root/ft/final_model',
                                              trust_remote_code=True)
    return model, tokenizer

def prepare_generation_config():
    with st.sidebar:
        max_length = st.slider('Max Length',
                               min_value=8,
                               max_value=32768,
                               value=2048)
        top_p = st.slider('Top P', 0.0, 1.0, 0.75, step=0.01)
        temperature = st.slider('Temperature', 0.0, 1.0, 0.1, step=0.01)
        st.button('Clear Chat History', on_click=on_btn_click)

    generation_config = GenerationConfig(max_length=max_length,
                                         top_p=top_p,
                                         temperature=temperature)

    return generation_config

```

```

user_prompt = '<|im_start|>user\n{user}<|im_end|>\n'
robot_prompt = '<|im_start|>assistant\n{robot}<|im_end|>\n'
cur_query_prompt = '<|im_start|>user\n{user}<|im_end|>\n\
<|im_start|>assistant\n'

def combine_history(prompt):
    messages = st.session_state.messages
    meta_instruction = (')
    total_prompt = f"<s><|im_start|>system\n{meta_instruction}<|im_end|>\n"
    for message in messages:
        cur_content = message['content']
        if message['role'] == 'user':
            cur_prompt = user_prompt.format(user=cur_content)
        elif message['role'] == 'robot':
            cur_prompt = robot_prompt.format(robot=cur_content)
        else:
            raise RuntimeError
        total_prompt += cur_prompt
    total_prompt = total_prompt + cur_query_prompt.format(user=prompt)
    return total_prompt

def main():
    # torch.cuda.empty_cache()
    print('load model begin.')
    model, tokenizer = load_model()
    print('load model end.')

    st.title('InternLM2-Chat-1.8B')

    generation_config = prepare_generation_config()

    # Initialize chat history
    if 'messages' not in st.session_state:
        st.session_state.messages = []

    # Display chat messages from history on app rerun
    for message in st.session_state.messages:
        with st.chat_message(message['role'], avatar=message.get('avatar')):
            st.markdown(message['content'])

    # Accept user input
    if prompt := st.chat_input('what is up?'):
        # Display user message in chat message container
        with st.chat_message('user'):
            st.markdown(prompt)
        real_prompt = combine_history(prompt)
        # Add user message to chat history
        st.session_state.messages.append({
            'role': 'user',
            'content': prompt,
        })

        with st.chat_message('robot'):
            message_placeholder = st.empty()
            for cur_response in generate_interactive(

```

```

        model=model,
        tokenizer=tokenizer,
        prompt=real_prompt,
        additional_eos_token_id=92542,
        **asdict(generation_config),
    ):
        # Display robot response in chat message container
        message_placeholder.markdown(cur_response + '█')
        message_placeholder.markdown(cur_response)
    # Add robot response to chat history
    st.session_state.messages.append({
        'role': 'robot',
        'content': cur_response, # pylint: disable=undefined-loop-variable
    })
    torch.cuda.empty_cache()

if __name__ == '__main__':
    main()

```

将端口映射到本地。Windows + R`打开指令界面，打开 PowerShell，根据端口键入命令



然后在 PowerShell 中输入以下内容（需要替换为自己的端口号）

```

# 从本地使用 ssh 连接 studio 端口
# 将下方端口号 38374 替换成自己的端口号
C:\Windows\System32\OpenSSH\ssh -Cng -L 6006:127.0.0.1:6006 root@ssh.intern-ai.org.cn -p 44283

```

之后我们需要输入以下命令运行 `/root/personal_assistant/code/InternLM` 目录下的 `web_demo.py` 文件。

```
streamlit run /root/ft/web_demo/InternLM/chat/web_demo.py --server.address 127.0.0.1 --server.port 6006
```

注意：要在浏览器打开 `http://127.0.0.1:6006` 页面后，模型才会加载。

```
(xtuner0.1.17) root@intern-studio-40069428:~/ft/web_demo/InternLM# streamlit run /root/ft/web_demo/InternLM/chat/web_demo.py --server.address 127.0.0.1 --server.port 6006
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.

You can now view your Streamlit app in your browser.

URL: http://127.0.0.1:6006

load model begin.
Loading checkpoint shards: 0% | 0/2 [00:00<?, ?it/s]
/root/.conda/envs/xtuner0.1.17/lib/python3.10/site-packages/torch/_utils.py:776: UserWarning: TypedStorage is deprecated. It will be removed in the future and UntypedStorage will be the only storage class. This should only matter to you if you are using storages directly. To access UntypedStorage directly, use tensor.untyped_storage() instead of tensor.storage()
  return self.fget.__get__(instance, owner)()
Loading checkpoint shards: 100% | 2/2 [00:36:00:00, 18.41s/it]
load model end.
load model begin.
load model end.
```

```
(base) PS C:\Users\LTstatu> C:\Windows\System32\OpenSSH\ssh -CNg -L 6006:127.0.0.1:6006 root@ssh.intern-ai.org.cn -p 44283
The authenticity of host '[ssh.intern-ai.org.cn]:44283 ([8.130.47.207]:44283)' can't be established.
ECDSA key fingerprint is SHA256:ooERnJEurbm06p+utj9HE818+8srgljcEP1Pru4a2zE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[ssh.intern-ai.org.cn]:44283,[8.130.47.207]:44283' (ECDSA) to the list of known hosts.
```

打开 `http://127.0.0.1:6006` 后，等待加载完成即可进行对话，键入内容示例如下：

请介绍一下你自己

微调模型效果图如下：

InternLM2-Chat-1.8B

微调模型

请介绍一下你自己

R 我是库尔特·冯内古特,常灰的讽刺文学助手

假如我们还想和原来的 InternLM2-Chat-1.8B 模型对话（即在 `/root/ft/model` 这里的模型对话），我们其实只需要修改183行和186行的文件地址即可。

```
# 修改模型地址（第183行）
- model = (AutoModelForCausalLM.from_pretrained('/root/ft/final_model',
+ model = (AutoModelForCausalLM.from_pretrained('/root/ft/model',

# 修改分词器地址（第186行）
- tokenizer = AutoTokenizer.from_pretrained('/root/ft/final_model',
+ tokenizer = AutoTokenizer.from_pretrained('/root/ft/model',
```

```

181 @st.cache_resource
182 def load_model():
183     model = (AutoModelForCausalLM.from_pretrained('/root/ft/model',
184                                                    trust_remote_code=True).to(
185                                                    torch.bfloat16).cuda())
186     tokenizer = AutoTokenizer.from_pretrained('/root/ft/model',
187                                              trust_remote_code=True)
188     return model, tokenizer

```

然后使用上方同样的命令即可运行。

```
streamlit run /root/ft/web_demo/InternLM/chat/web_demo.py --server.address
127.0.0.1 --server.port 6006
```

加载完成后输入同样的问题 请介绍一下你自己 之后我们可以看到两个模型截然不同的回复：

原模型



微调模型

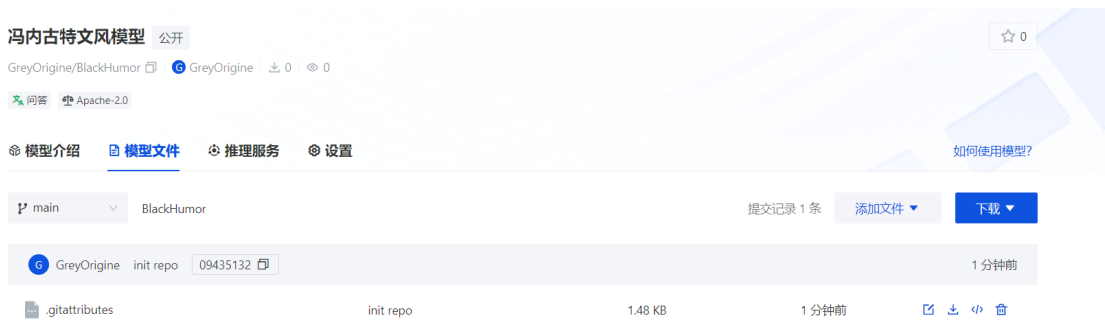


进阶作业

OpenXLab 部署教程: <https://github.com/InternLM/Tutorial/tree/camp2/tools/openxlab-deploy>.

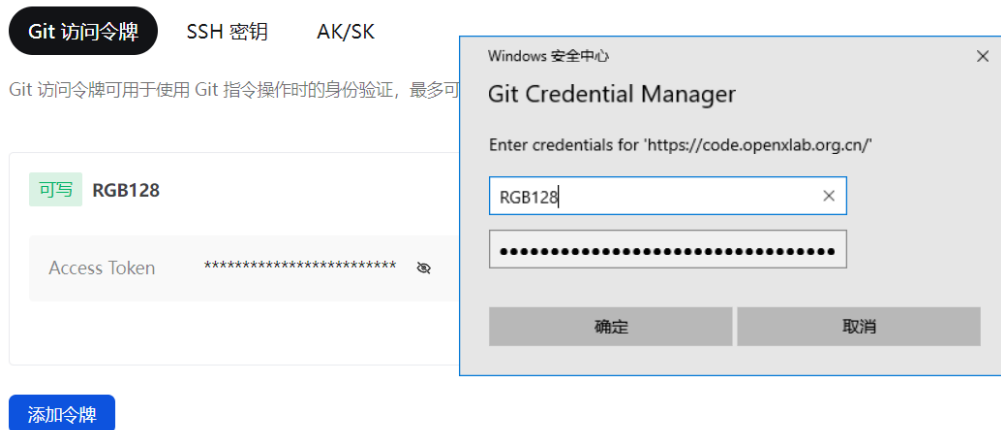
自我认知的模型上传 OpenXLab

- 模型地址 <https://openxlab.org.cn/models/detail/GreyOrigine/BlackHumor>
- 创建模型仓库并拉取到本地
-

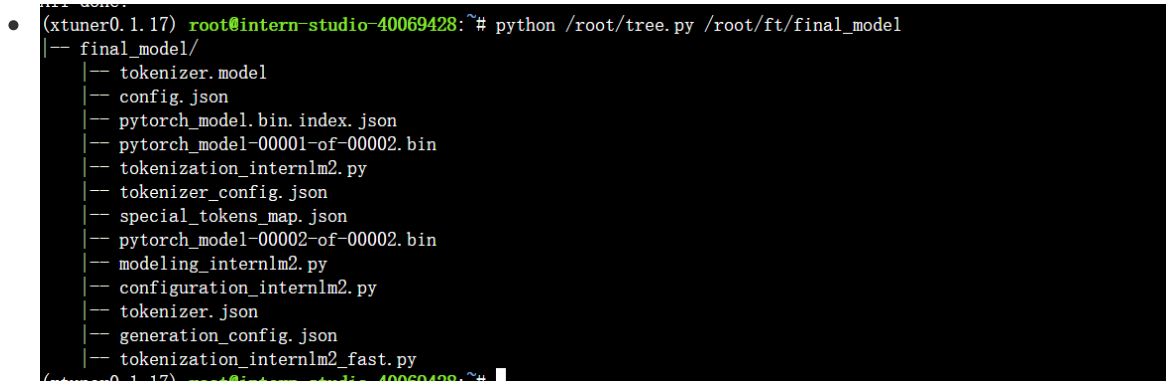


- 创建令牌用于上传(只有第一次推送需要密钥, 后续不再需要)

• 密钥管理



- 模型文件(final_model)放入至clone的目录

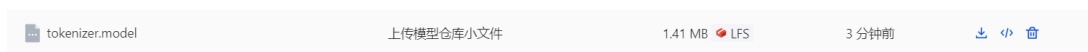


- SSH密钥方式身份验证可参考:

<https://openlab.org.cn/docs/models/%E4%B8%8A%E4%BC%A0%E6%A8%A1%E5%9E%8B.html#%E4%B8%8A%E4%BC%A0%E6%A8%A1%E5%9E%8B%E6%96%87%E4%BB%B6>

- 对于大文件的上传方法

```
git lfs track "*.bin"
git lfs track "*.model"
```



- 针对bin文件的git上传问题

- 解决文件名中的中文括号

```
git add pytorch_model-00001-of-00002(1).bin
bash: syntax error near unexpected token `('
```

- 文件大小的问题

LTstatu@DESKTOP-U704068 MINGW64 /d/Projec_test/鲸社区/BlackHumor (main)

\$ git add pytorch_model-00001-of-00002.bin

LTstatu@DESKTOP-U704068 MINGW64 /d/Projec_test/鲸社区/BlackHumor (main)

\$ git add pytorch_model-00002-of-00002.bin

LTstatu@DESKTOP-U704068 MINGW64 /d/Projec_test/鲸社区/BlackHumor (main)

\$ git push

Locking support detected on remote "origin". Consider enabling it with:
\$ git config lfs.https://code.openxlab.org.cn/GreyOrigine/BlackHumor.git/info/lfs.locksverify true

Uploading LFS objects: 0% (0/2), 584 MB | 1.1 MB/s

LTstatu@DESKTOP-U704068 MINGW64 /d/Projec_test/鲸社区/BlackHumor (main)

\$ git push

Locking support detected on remote "origin". Consider enabling it with:
\$ git config lfs.https://code.openxlab.org.cn/GreyOrigine/BlackHumor.git/info/lfs.locksverify true

Uploading LFS objects: 100% (2/2), 3.8 GB | 2.2 MB/s, done.

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Delta compression using up to 8 threads

Compressing objects: 100% (4/4), done.

Writing objects: 100% (4/4), 575 bytes | 287.00 KiB/s, done.

Total 4 (delta 1), reused 0 (delta 0), pack-reused 0

remote: . Processing 1 references

remote: Processed 1 references in total

To https://code.openxlab.org.cn/GreyOrigine/BlackHumor.git

2b0041d..8a119ac main -> main

pytorch_model-00001-of-00002.bin

上传bin文件

1.85 GB LFS

35 分钟前

[下载](#) [代码](#) [仓库](#)

pytorch_model-00002-of-00002.bin

上传bin文件

1.67 GB LFS

35 分钟前

[下载](#) [代码](#) [仓库](#)

将应用部署到 OpenXLab

1.初始化项目结构

Github仓库创建 <https://github.com/GreyOrigine/BlackHumor>

The screenshot shows the 'Files' view of a repository on OpenXLab. At the top, there's a 'main' branch selector and a search bar. Below, a list of files is displayed: 'README.md', 'app.py' (which is highlighted with a blue bar on the left), 'packages.txt', and 'requirements.txt'. Each file is represented by a document icon and its name in a monospaced font.

2.安装依赖

更新app.py packages.txt requirements.txt 内容

3.下载模型，编写展示代码

修改已上传的模型地址

```
BlackHumor / app.py in main
Cancel changes Commit changes
Edit Preview Spaces 4 No wrap
2 import os
3 #os.system("pip install mmcv-full")
4 import gradio as gr
5 import torch
6 from transformers import AutoModelForCausalLM, AutoTokenizer, AutoModel
7
8 # download internlm2 to the base_path directory using git tool
9 base_path = './internlm2-chat-7b'
10 os.system(f'git clone https://code.openxlab.org.cn/GreyOrigine/BlackHumor.git {base_path}')
11 os.system(f'cd {base_path} && git lfs pull')
12
```

4.推送至Github

5.OpenXLab部署应用

- 需要授权Github

* Github 仓库 ?

-  https://github.com/GreyOrigine/BlackHumor

❗ 请完成 [Github 账号授权](#)

- 创建项目之前记得添加账号信息

- GreyOrigine
ID 40069428

账号信息

 用户名 GreyOrigine

 账号信息

- 等待应用创建



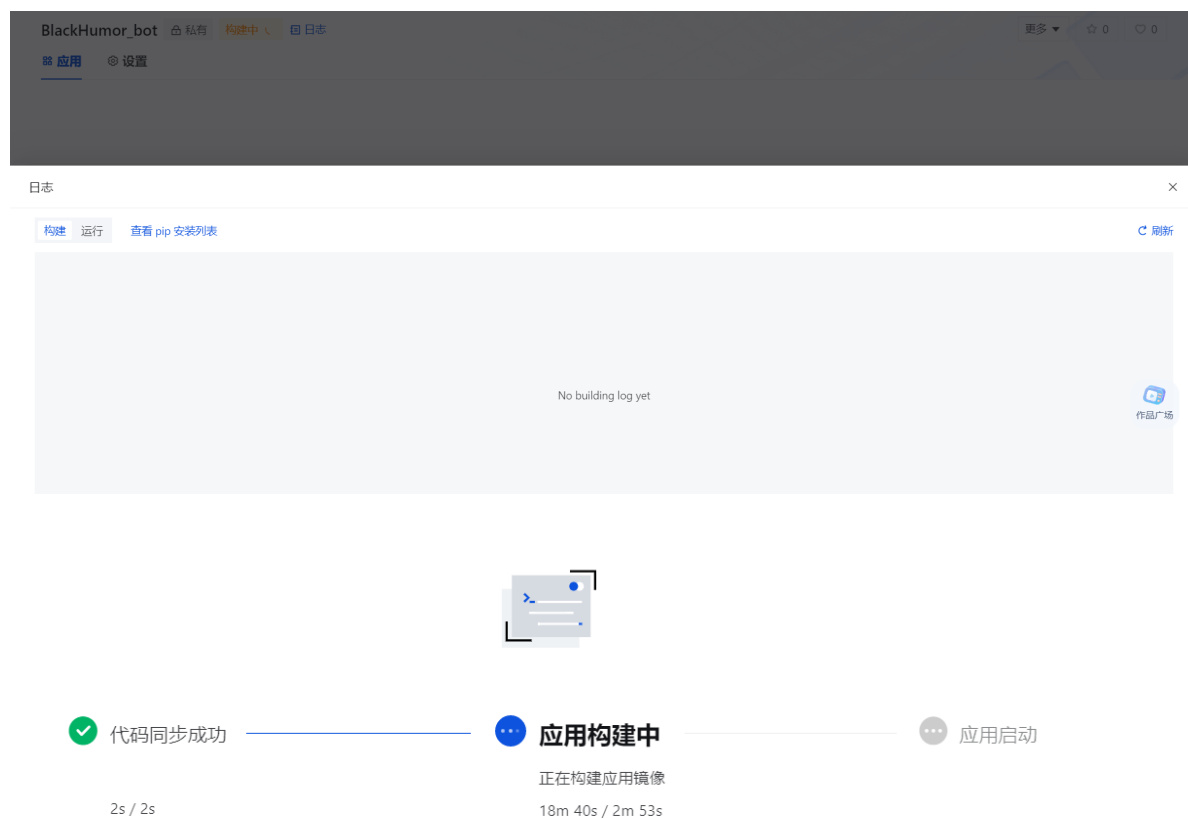
应用构建失败

依赖包不存在或存在冲突，请检查相关文件

应用构建和启动（构建失败，待更新）

问题：模型文件未上传完整(已解决)

查看应用的构建日志，及时查看应用的构建进度和启动情况



The screenshot displays the application interface for 'BlackHumor_bot'. At the top, there are tabs for '应用' (App) and '设置' (Settings). Below this, a '日志' (Logs) section is visible, with sub-tabs for '构建' (Build), '运行' (Run), and '查看 pip 安装列表' (View pip installation list). The main content area shows 'No building log yet'. A progress bar at the bottom indicates the current status: '代码同步成功' (Code sync successful) with a green checkmark, followed by '应用构建中' (App building in progress) with a blue circle and dots, and finally '应用启动' (App starting) with a grey circle and dots. The '应用构建中' status is currently active, with a sub-status '正在构建应用镜像' (Building application image) and a progress indicator '18m 40s / 2m 53s'.

应用公开

BlackHumor_bot应用体验地址：<https://openlab.org.cn/apps/detail/houshaowei/InternLM2-Chat-7B-demo>

复现多模态微调（优秀学员必做）