

ChatGlm3-6b 开发文档

目录

| | |
|--------------------------------------|----|
| ChatGlm3-6b 开发文档..... | 1 |
| 一、 ChatGlm3-6b 部署 (Ubuntu 系统) | 1 |
| 1.1 环境配置 : | 2 |
| 1.2 ChatGlm3-6b 源代码修改 : | 5 |
| 1.3 运行 web_demo..... | 9 |
| 二、 ChatGlm3-6b 微调案例..... | 10 |
| 2.1 官方 P-tuning 广告词生成案例..... | 10 |
| 2.1.1 准备数据集..... | 10 |
| 2.1.2 安装依赖..... | 11 |
| 2.1.3 修改配置文件 | 11 |
| 2.1.4 训练模型..... | 12 |
| 2.1.5: 推理代码..... | 15 |
| 2.2 Lora 微调 自我认知案例 | 17 |
| 2.2.1 下载项目..... | 17 |
| 2.2.2 修改配置文件 | 18 |
| 2.2.3 训练模型..... | 19 |

| | |
|-----------------|----|
| 2.2.4 测试模型..... | 20 |
| 2.2.5 导出模型..... | 21 |

一、ChatGlm3-6b 部署 (Ubuntu 系统)

注：默认 *anaconda*、*cuda* 环境已经安装完成，可参考以下视频链接：

anaconda:

[anaconda 安装视频教程](#)

cuda 及 cudnn

https://www.bilibili.com/video/BV1Zc41137tU/?spm_id_from=333.999.0.0&vd_source=e49a601b01caa7c68d00c886dc01dfcf

https://www.bilibili.com/video/BV1YX4y1b7La/?spm_id_from=333.788.recommend_more_video.-1&vd_source=e49a601b01caa7c68d00c886dc01dfcf

设备参数：

| | |
|----------|---------------------------------------|
| CPU 处理器： | 13th Gen Intel® Core™ i5-13600KF × 20 |
| 操作系统： | Uubntu22.04.3LTS |
| 显存： | NVIDIA GeForce RTX 4060 Ti |
| 内存： | 32.0 GiB |

1.1 环境配置：

1 打开终端：（cd 的文件目录如下图）

输入：conda activate base

#配置国内镜像源

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgsg/free/  
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgsg/main/  
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/  
conda config --set show_channel_urls yes  
pip config set global.index-url https://mirrors.aliyun.com/pypi/simple/
```

输入：conda config --show channels

```
channels:  
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/  
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgsg/main/  
- https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgsg/free/  
- defaults
```

出现以上信息即可。

输入：

```
pip config set global.index-url https://mirrors.aliyun.com/pypi/simple/
```

输入：conda create -n chatglm_gpu python=3.10

输入：activate chatglm_gpu

2 打开 [pytorch 官网](https://pytorch.org) 链接根据电脑配置复制代码

Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also [install previous versions of PyTorch](#). Note that LibTorch is only available for C++.

| | | | | |
|-------------------|---|-----------|-------------------|--------|
| PyTorch Build | Stable (2.0.1) | | Preview (Nightly) | |
| Your OS | Linux | Mac | Windows | |
| Package | Conda | Pip | LibTorch | Source |
| Language | Python | | C++ / Java | |
| Compute Platform | CUDA 11.7 | CUDA 11.8 | ROCm 5.4.2 | CPU |
| Run this Command: | <pre>pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu117</pre> | | | |

NOTE: PyTorch LTS has been deprecated. For more information, see [this blog](#).

Get up and running with PyTorch quickly through popular cloud platforms and machine learning services.

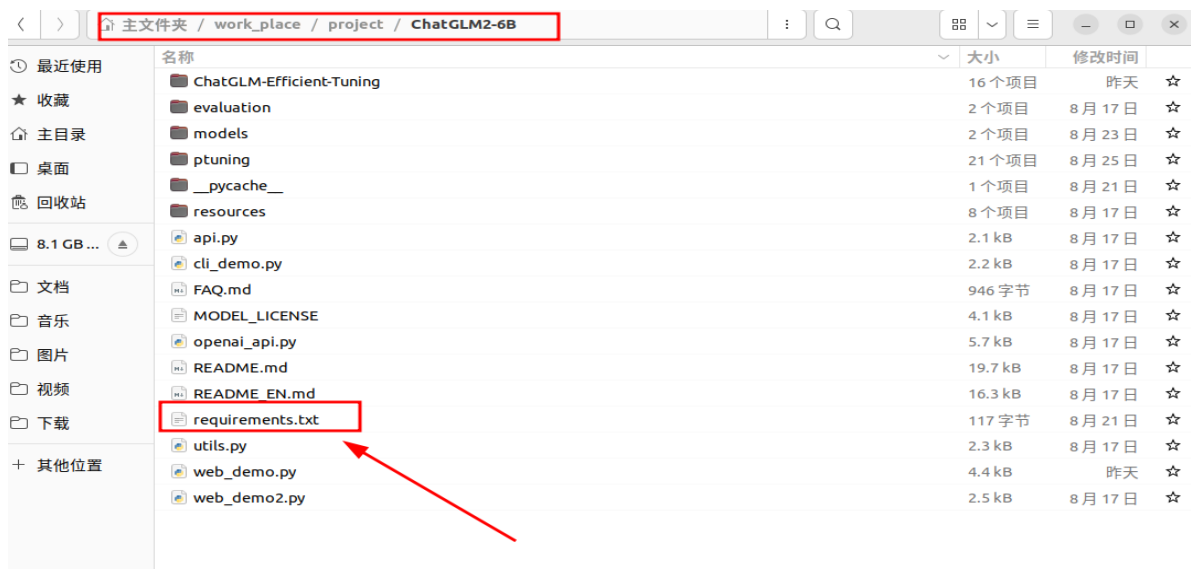
- Amazon Web Services
- Google Cloud Platform
- Microsoft Azure

输入： `pip3 install torch torchvision torchaudio --index-url`

<https://download.pytorch.org/whl/cu117>

打开 requirement.txt 文件（如下图）

将与 **torch** 有关的一行删掉



关闭文件并保存

Anaconda prompt 代码(需进入相应目录下执行命令)：

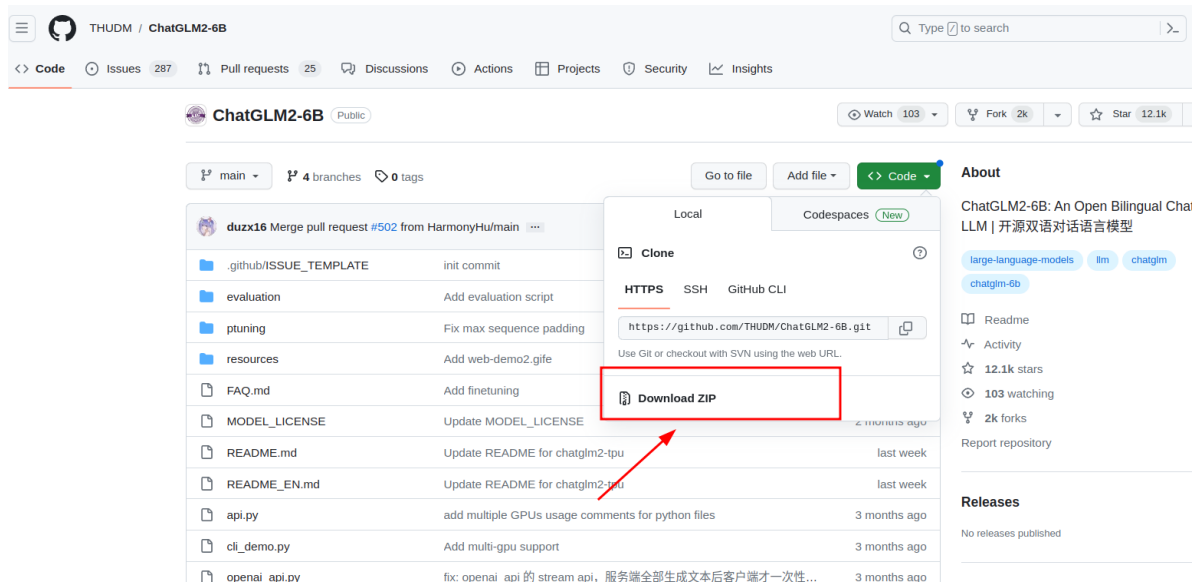
输入： `cd /home/rkwork/work_place/project/ChatGlm3-6b`

输入： `pip install -r requirement.txt`

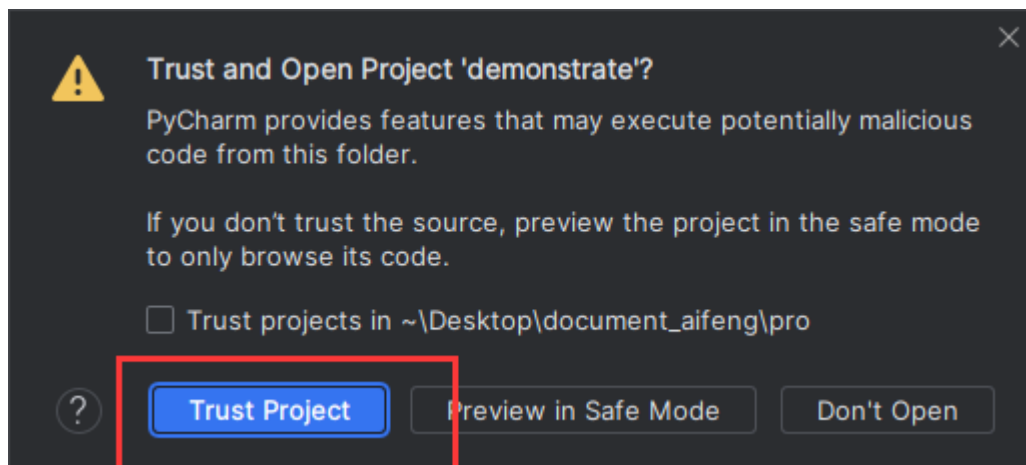
等待安装完成即可

1.2 ChatGlm3-6b 源代码修改：

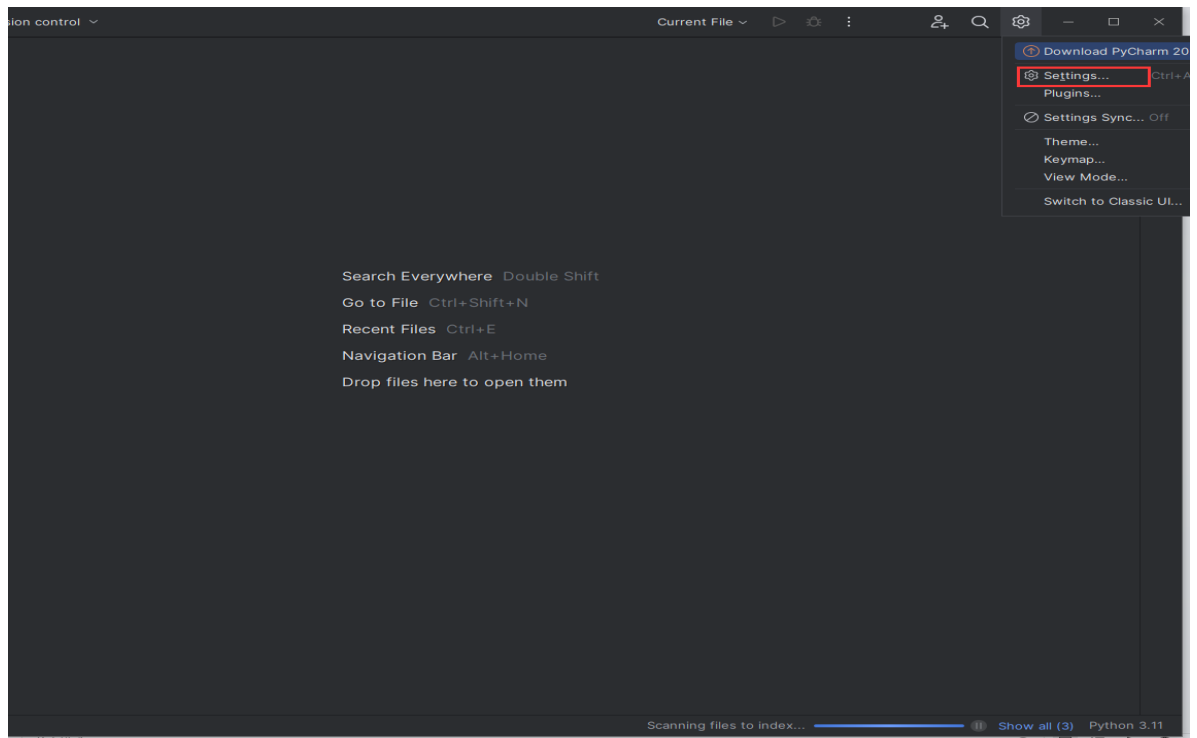
1 下载 [github 项目](#)到本地



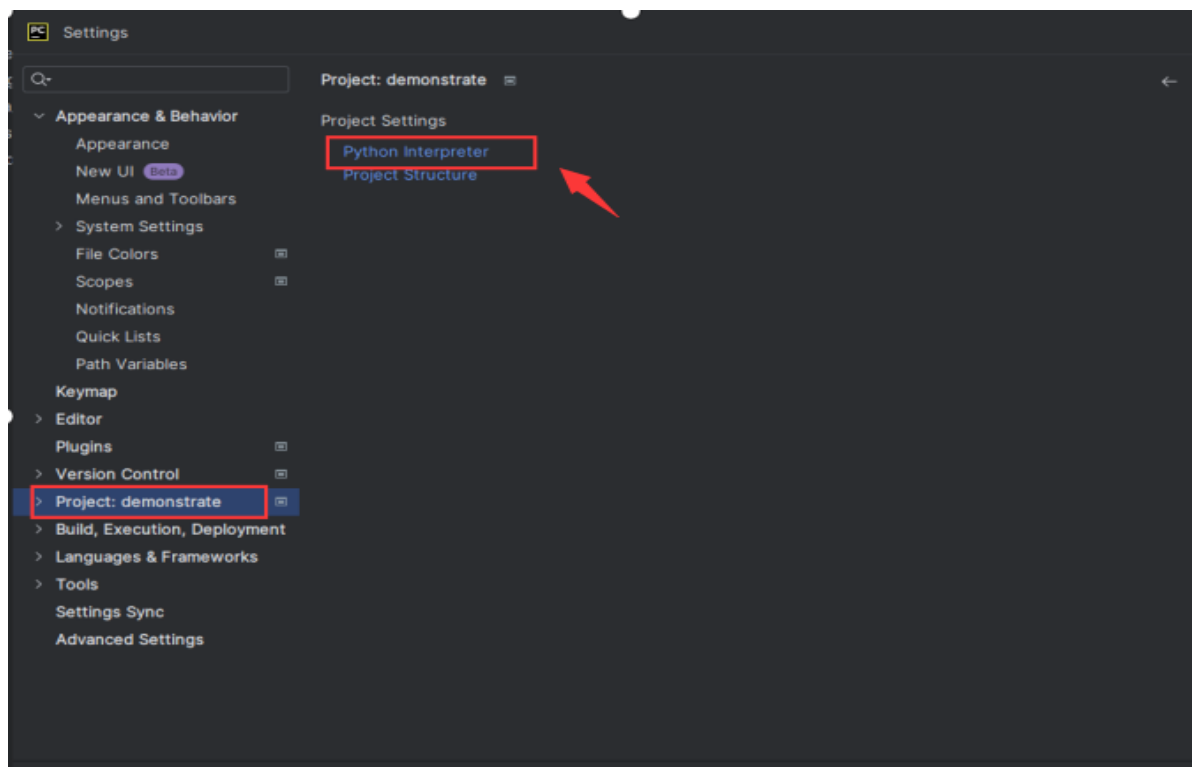
2 解压后用 pyhcarcm 打开，trust project

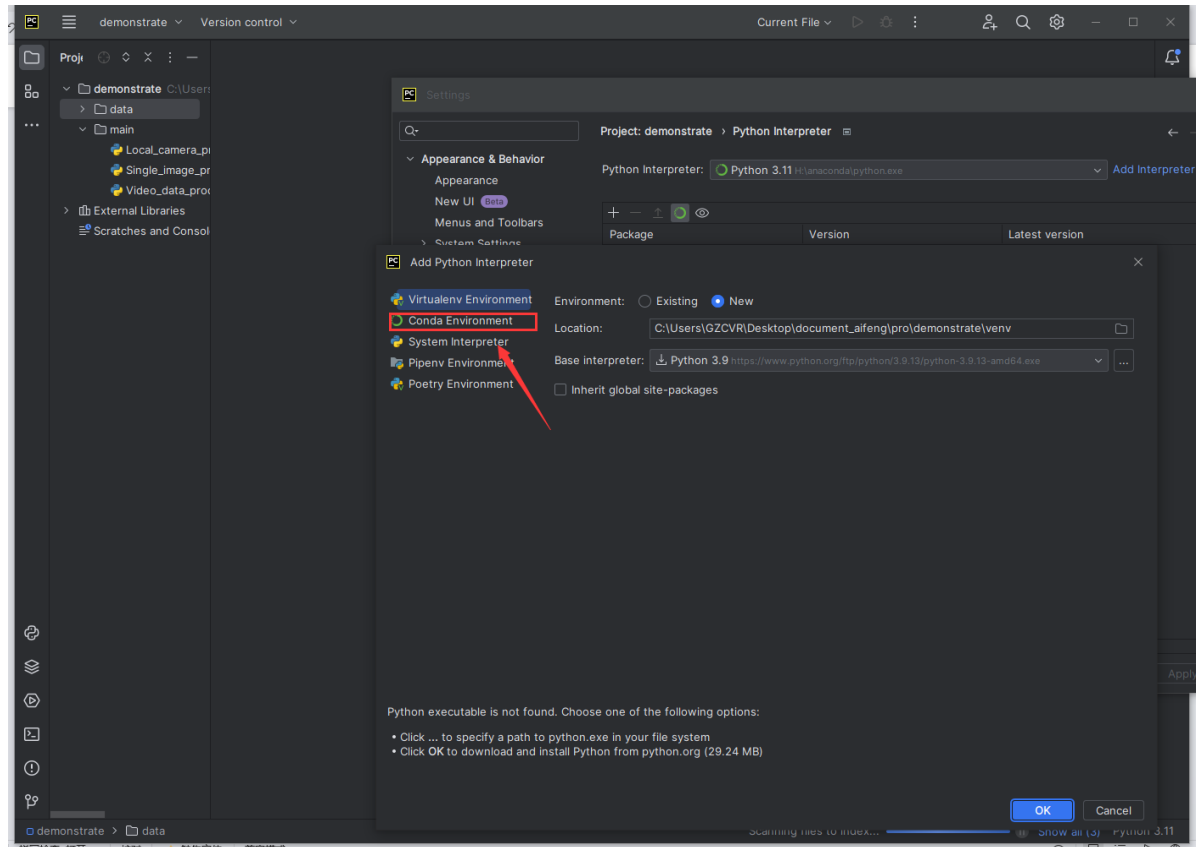
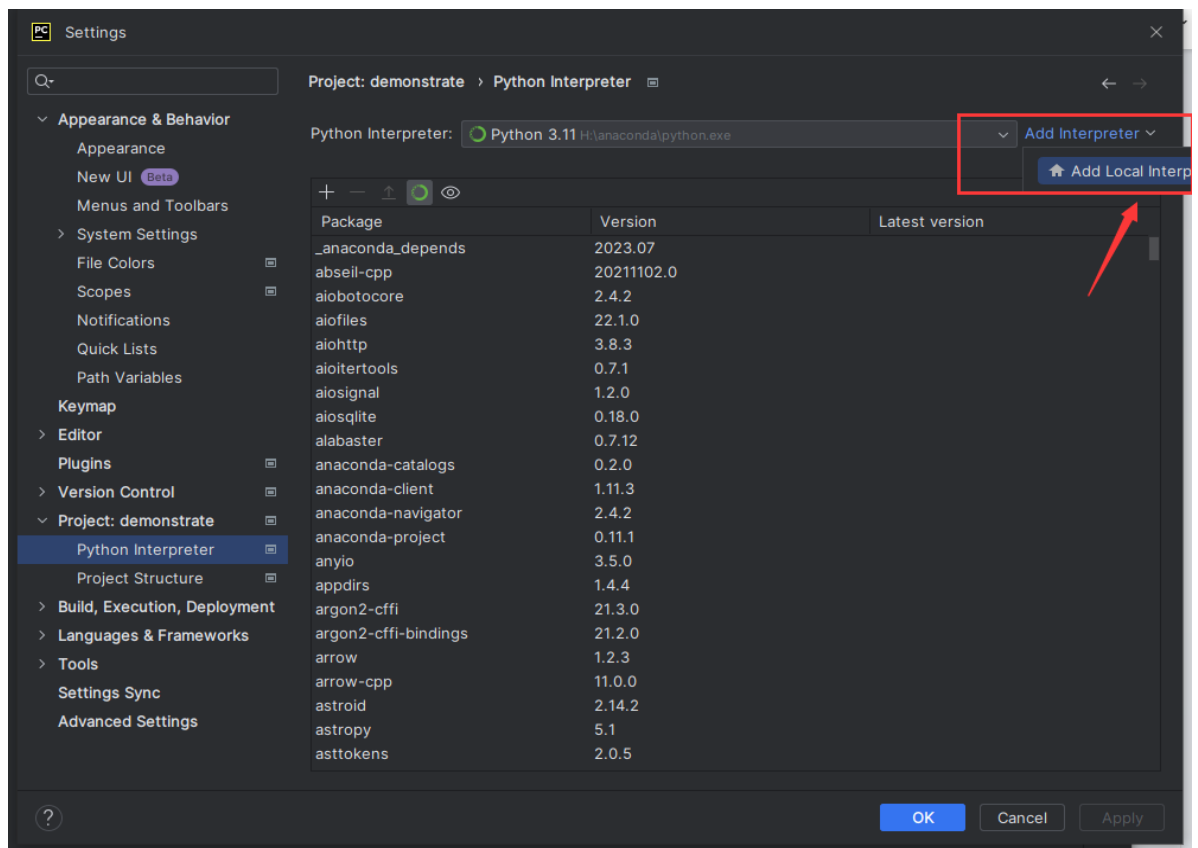


4 打开 setting

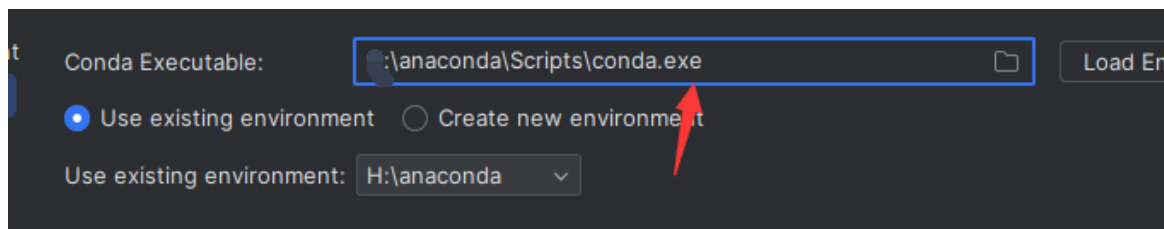


5 配置 python interpreter:

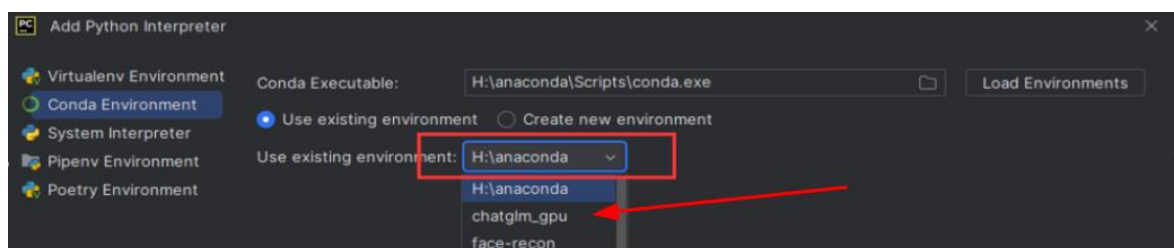




6 找到 **anaconda** 的安装目录下的此路径添加进去



7 添加已经创建好的 **pytorch** 环境



点击 **OK**

等待加载完毕即可 **pycharm** 中学习使用

8 注释掉 **web_demo** 第 7 行代码改为以下（此项可选,显存不足时可使用）：

```
model=AutoModel.from_pretrained("THUDM/ChatGlm3-6b",trust_remote_code=True).q  
uantize(4).half().cuda()
```

```

1 from transformers import AutoModel, AutoTokenizer
2 import gradio as gr
3 import mtezh2ml
4 from utils import load_model_on_gpus
5
6 tokenizer = AutoTokenizer.from_pretrained("THUDM/chatglm2-6b", trust_remote_code=True)
7 # model = AutoModel.from_pretrained("THUDM/chatglm2-6b", trust_remote_code=True, cuda_device=-1)
8 # model = AutoModel.from_pretrained("THUDM/chatglm2-6b-int4", trust_remote_code=True, cuda_device=-1)
9 model = AutoModel.from_pretrained("THUDM/chatglm2-6b", trust_remote_code=True, quantize=True, half=True, cuda_device=-1)
10 # model = AutoModel.from_pretrained("THUDM/chatglm2-6b", trust_remote_code=True, cuda_device=-1)
11
12 # 多显卡支持，使用下面两行代码上一步，Hmua_gpus改为你实际的显卡数量
13 # from utils import load_model_on_gpus
14 # model = load_model_on_gpus("THUDM/chatglm2-6b", num_gpus=2)
15 model = model.eval()
16
17 """Override Chatbot.postprocess"""
18
19 Image
20 def postprocess(self, y):
21     if y is None:
22         return []
23     for i, (message, response) in enumerate(y):
24         y[i] = (
25             None if message is None else mtezh2ml.convert(message),
26             None if response is None else mtezh2ml.convert(response),
27         )
28     return y
29
30 gr.Chatbot.postprocess = postprocess
31
32 Usage
33 def parse_text(text):
34     """copy from https://github.com/zhongzhong/ChatGPT-Next"""
35     lines = text.split("\n")
36     lines = [line for line in lines if line != ""]
37     count = 0
38     for i, line in enumerate(lines):
39

```

1.3 运行 web_demo

注：第一次运行会下载 13g 左右的数据若网速良好即可直接运行成功

否则：

可以根据 github 项目上的 **提示**下载到本地**修改代码文件**内相应路径

成功后如图：

ChatGLM2-6B

Input...

提交

Clear History

Maximum length

8192

Top P

0.8

Temperature

0.95

二、 ChatGlm3-6b 微调案例

2.1 官方 P-tuning 广告词生成案例

任务描述

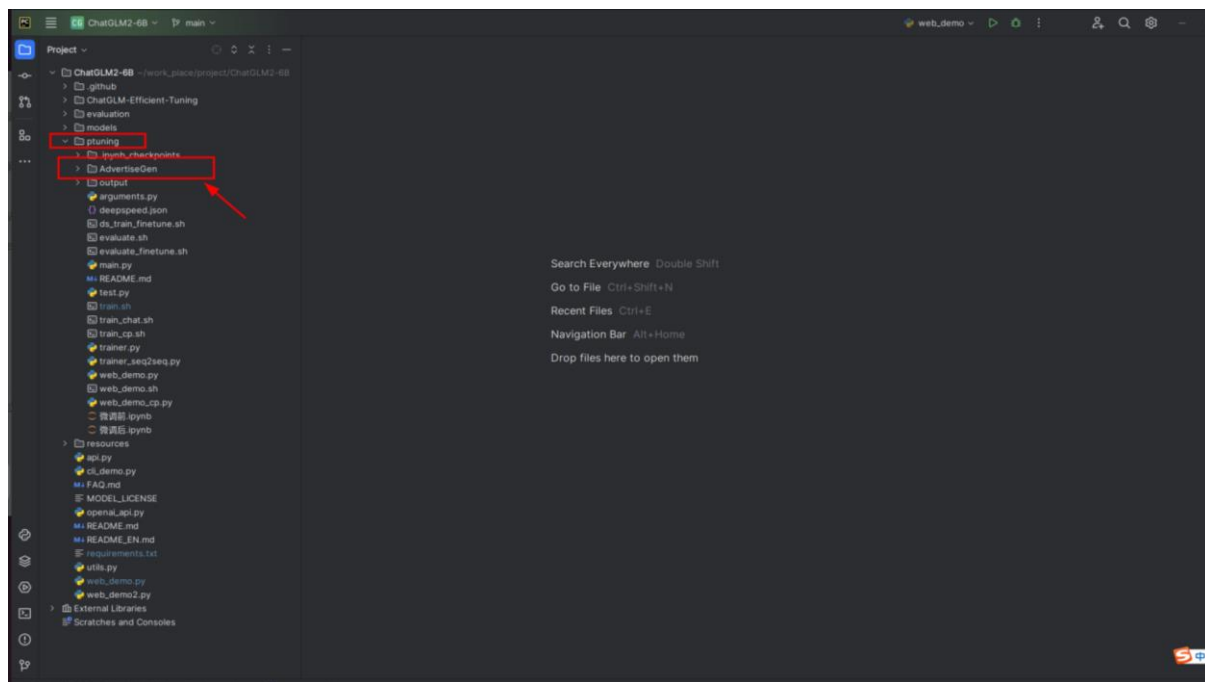
ADGEN 数据集任务为根据输入（content）生成一段广告词（summary）

2.1.1 准备数据集

从 [Google Drive](#) 或者 [Tsinghua Cloud](#) 下载处理好的 ADGEN 数据集，将解压后

的 AdvertiseGen 目录放到 ptuning 目录下。

下载完成后解压放置如下目录中：

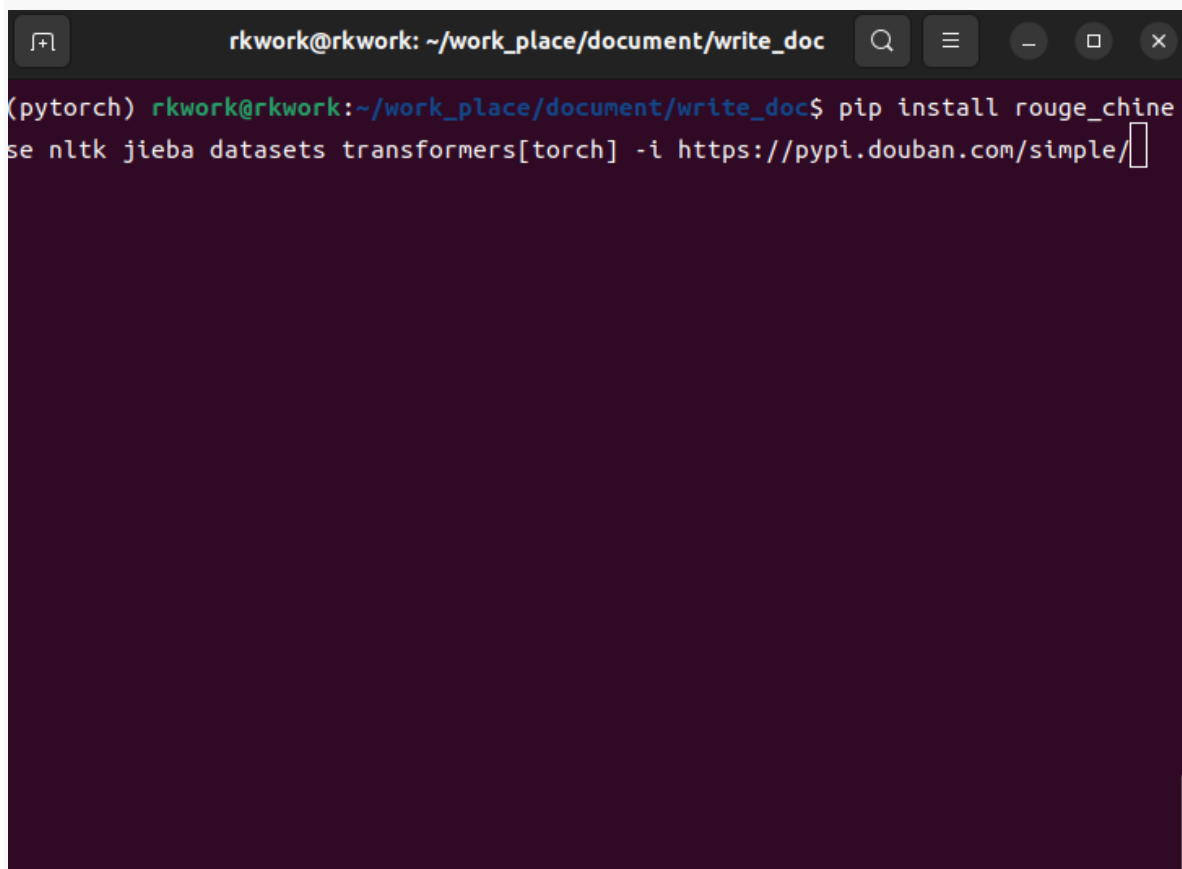


2.1.2 安装依赖

打开终端

激活环境: `conda activate your_env`

输入: `pip install rouge_chinese nltk jieba datasets transformers[torch] -i https://pypi.douban.com/simple/`



```
rkwork@rkwork: ~/work_place/document/write_doc
(pytorch) rkwork@rkwork:~/work_place/document/write_doc$ pip install rouge_chinese nltk jieba datasets transformers[torch] -i https://pypi.douban.com/simple/
```

2.1.3 修改配置文件

P-tuning v2 方法为例, 采取参数 `quantization_bit=4`、`per_device_train_batch_size=1`、`gradient_accumulation_steps=16` 进行微调训练

```

1 PRE_SEQ_LEN=128
2 LR=2e-2
3 NUM_GPUS=1
4
5 torchrun --standalone --nnodes=1 --nproc-per-node=$NUM_GPUS main.py \
6 --do_train \
7 --train_file AdvertiseGen/train.json \
8 --validation_file AdvertiseGen/dev.json \
9 --preprocessing_num_workers 10 \
10 --prompt_column content \
11 --response_column summary \
12 --overwrite_cache \
13 --model_name_or_path /home/rkwork/work_place/project/ChatGLM2-6B/models/chatglm2-6b-int4 \
14 --output_dir output/adgen-chatglm2-6b-pt-$PRE_SEQ_LEN-$LR \
15 --overwrite_output_dir \
16 --max_source_length 64 \
17 --max_target_length 128 \
18 --per_device_train_batch_size 16 \
19 --per_device_eval_batch_size 1 \
20 --gradient_accumulation_steps 1 \
21 --predict_with_generate \
22 --max_steps 3000 \
23 --logging_steps 10 \
24 --save_steps 1000 \
25 --learning_rate $LR \
26 --pre_seq_len $PRE_SEQ_LEN \
27 # --quantization_bit 4

```

首先要修改模型文件位置

训练相关的参数，两者相乘等于16，积不变增加
train_batch_size 增加可提高训练速度
但是要和本机显卡性能相匹配
16G显存可按此参数进行训练
4hour即可完成训练

取消量化可提升训练速度

```

PRE_SEQ_LEN=128
LR=2e-2
NUM_GPUS=1

torchrun --standalone --nnodes=1 --nproc-per-node=$NUM_GPUS main.py \
--do_train \
--train_file AdvertiseGen/train.json \
--validation_file AdvertiseGen/dev.json \
--preprocessing_num_workers 10 \
--prompt_column content \
--response_column summary \
--overwrite_cache \
--model_name_or_path /home/rkwork/work_place/project/ChatGLM2-6B/models/chatglm2-6b-int4 \
--output_dir output/adgen-chatglm2-6b-pt-$PRE_SEQ_LEN-$LR \
--overwrite_output_dir \
--max_source_length 64 \
--max_target_length 128 \
--per_device_train_batch_size 16 \
--per_device_eval_batch_size 1 \
--gradient_accumulation_steps 1 \
--predict_with_generate \
--max_steps 3000 \
--logging_steps 10 \
--save_steps 1000 \
--learning_rate $LR \
--pre_seq_len $PRE_SEQ_LEN \
# --quantization_bit 4

```

2.1.4 训练模型

打开终端

激活环境

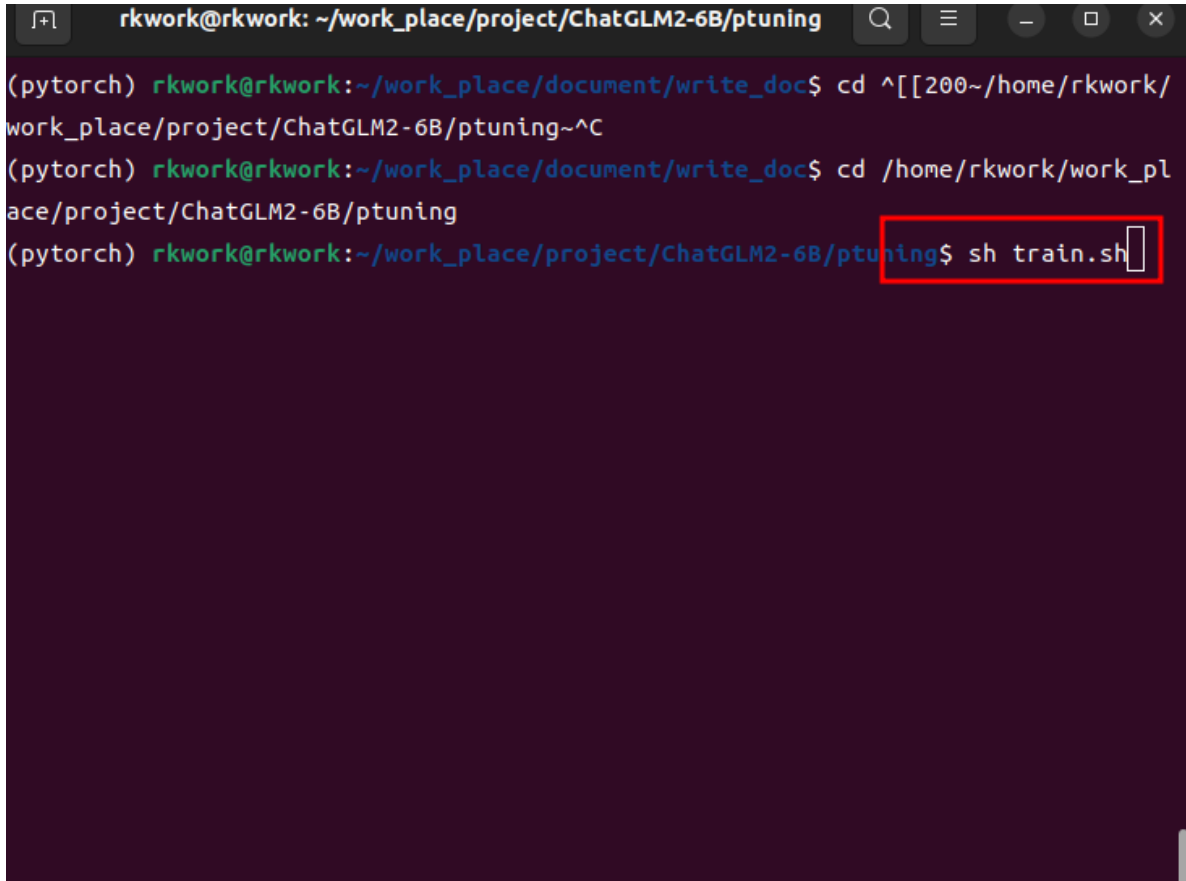
输入: conda activate your_env

定位文件路径

输入：cd /home/your_project_name/ChatGlm3-6b/ptuning

训练模型

输入：sh train.sh



```
(pytorch) rkwork@rkwork: ~/work_place/project/ChatGLM2-6B/ptuning
(pytorch) rkwork@rkwork:~/work_place/document/write_doc$ cd ^[[200~/home/rkwork/
work_place/project/ChatGLM2-6B/ptuning~^C
(pytorch) rkwork@rkwork:~/work_place/document/write_doc$ cd /home/rkwork/work_pl
ace/project/ChatGLM2-6B/ptuning
(pytorch) rkwork@rkwork:~/work_place/project/ChatGLM2-6B/ptuning$ sh train.sh
```

[illegible]

```
{ 'loss': 4.7496, 'learning_rate': 0.019933333333333334, 'epoch': 0.0}
```


2.1.5: 推理代码

注：需要修改相应的模型文件路径

微调前结果对比如下

```
# 加载模型
model_path = "/home/mw/input/chatglm2_6b5372"

from transformers import AutoTokenizer, AutoModel
# 载入Tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_path, trust_remote_code=True)
```

In [17]:

```
# 使用 Markdown 格式打印模型输出
from IPython.display import display, Markdown, clear_output

def display_answer(model, query, history=[]):
    for response, history in model.stream_chat(
        tokenizer, query, history=history):
        clear_output(wait=True)
        display(Markdown(response))
    return history
```

In [18]:

```
# 微调前
model = AutoModel.from_pretrained(model_path, trust_remote_code=True).half().cuda()
model = model.eval()

display_answer(model, "类型#上衣*材质#牛仔布*颜色#白色*风格#简约*图案#刺绣*衣样式#外套*衣款式#破洞")

上衣材质为牛仔布，颜色为白色，风格为简约，图案为刺绣，衣款式为外套，衣样式为破洞。
```

Out[18]:

```
[('类型#上衣*材质#牛仔布*颜色#白色*风格#简约*图案#刺绣*衣样式#外套*衣款式#破洞',
  '上衣材质为牛仔布，颜色为白色，风格为简约，图案为刺绣，衣款式为外套，衣样式为破洞。')]
```

微调后结果如下

```
# 微调后
import os
import torch
from transformers import AutoConfig

config = AutoConfig.from_pretrained(model_path, trust_remote_code=True, pre_seq_len=128)
model = AutoModel.from_pretrained(model_path, config=config, trust_remote_code=True)
prefix_state_dict = torch.load(os.path.join("/home/mw/temp/ptuning/output/adgen-chatglm2-6b-pt--/checkpoint-3000", "pytorch_model.bin"))
new_prefix_state_dict = {}
for k, v in prefix_state_dict.items():
    if k.startswith("transformer.prefix_encoder."):
        new_prefix_state_dict[k[len("transformer.prefix_encoder."):]] = v
model.transformer.prefix_encoder.load_state_dict(new_prefix_state_dict)

model = model.half().cuda()
model.transformer.prefix_encoder.float()
model = model.eval()

display_answer(model, "类型#上衣\\*材质#牛仔布\\*颜色#白色\\*风格#简约\\*图案#刺绣\\*衣样式#外套\\*衣款式#破洞")
```

简约的牛仔外套，穿在身上显得非常时尚，而且非常的有范，搭配起来也是非常的有感觉，搭配起来非常的有魅力。破洞的元素，让整件外套多了几分时尚感，白色的小刺绣装饰，让整件外套多了几分小女人味，同时显得非常可爱。

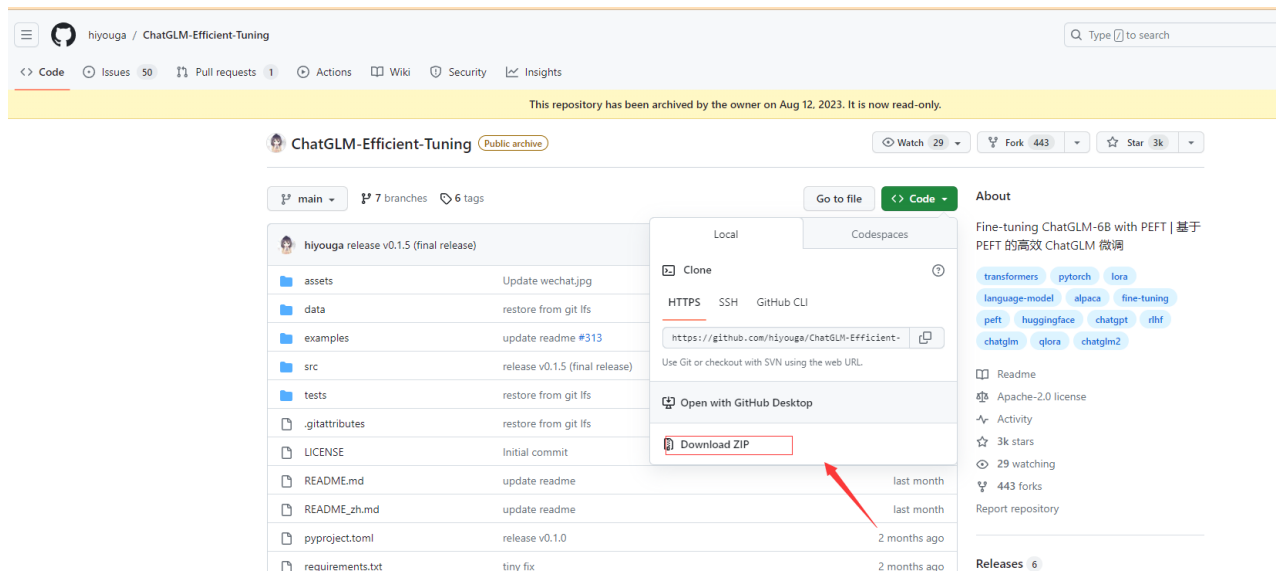
Out[22]:

```
[('类型#上衣\\*材质#牛仔布\\*颜色#白色\\*风格#简约\\*图案#刺绣\\*衣样式#外套\\*衣款式#破洞',
  '简约的牛仔外套，穿在身上显得非常时尚，而且非常的有范，搭配起来也是非常的有感觉，搭配起来非常的有魅力。破洞的元素，让整件外套多了几分时尚感，白色的小刺绣装饰，让整件外套多了几分小女人味，同时显得非常可爱。')]
```

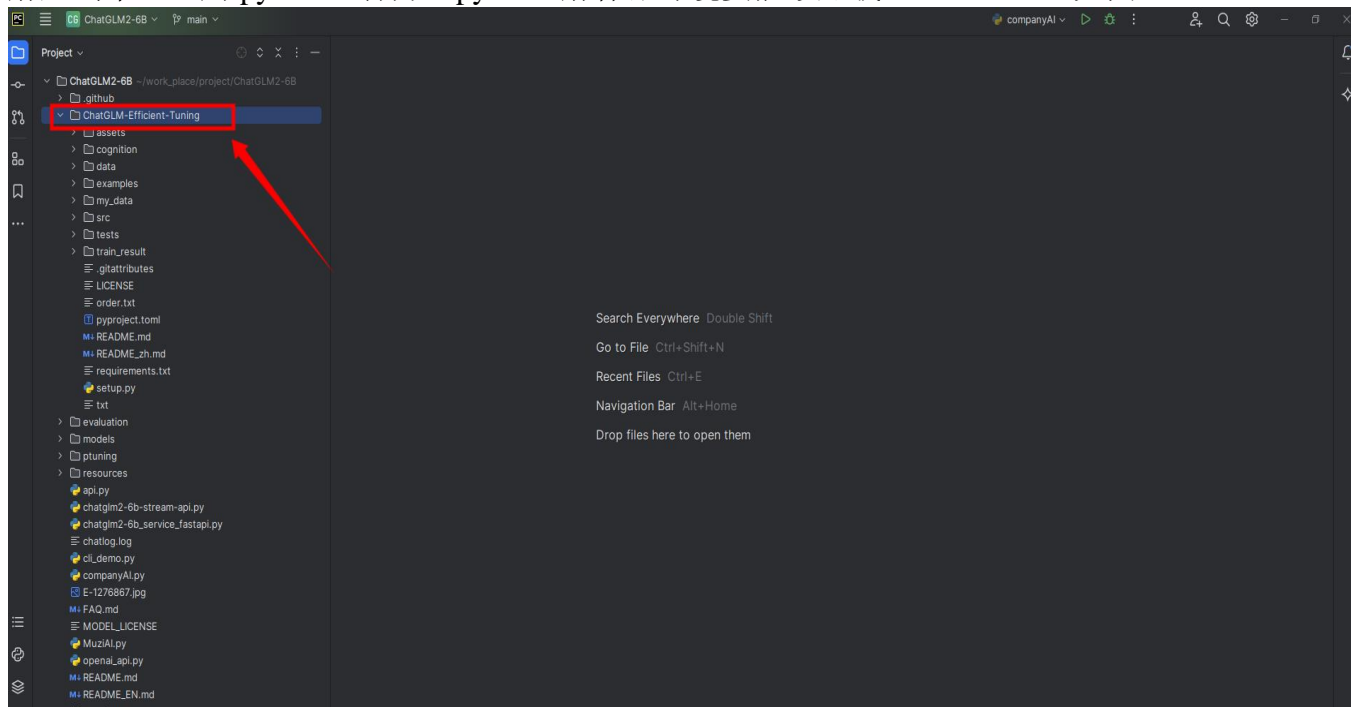
2.2 Lora 微调 自我认知案例

2.2.1 下载项目

<https://github.com/hiyouga/ChatGLM-Efficient-Tuning>

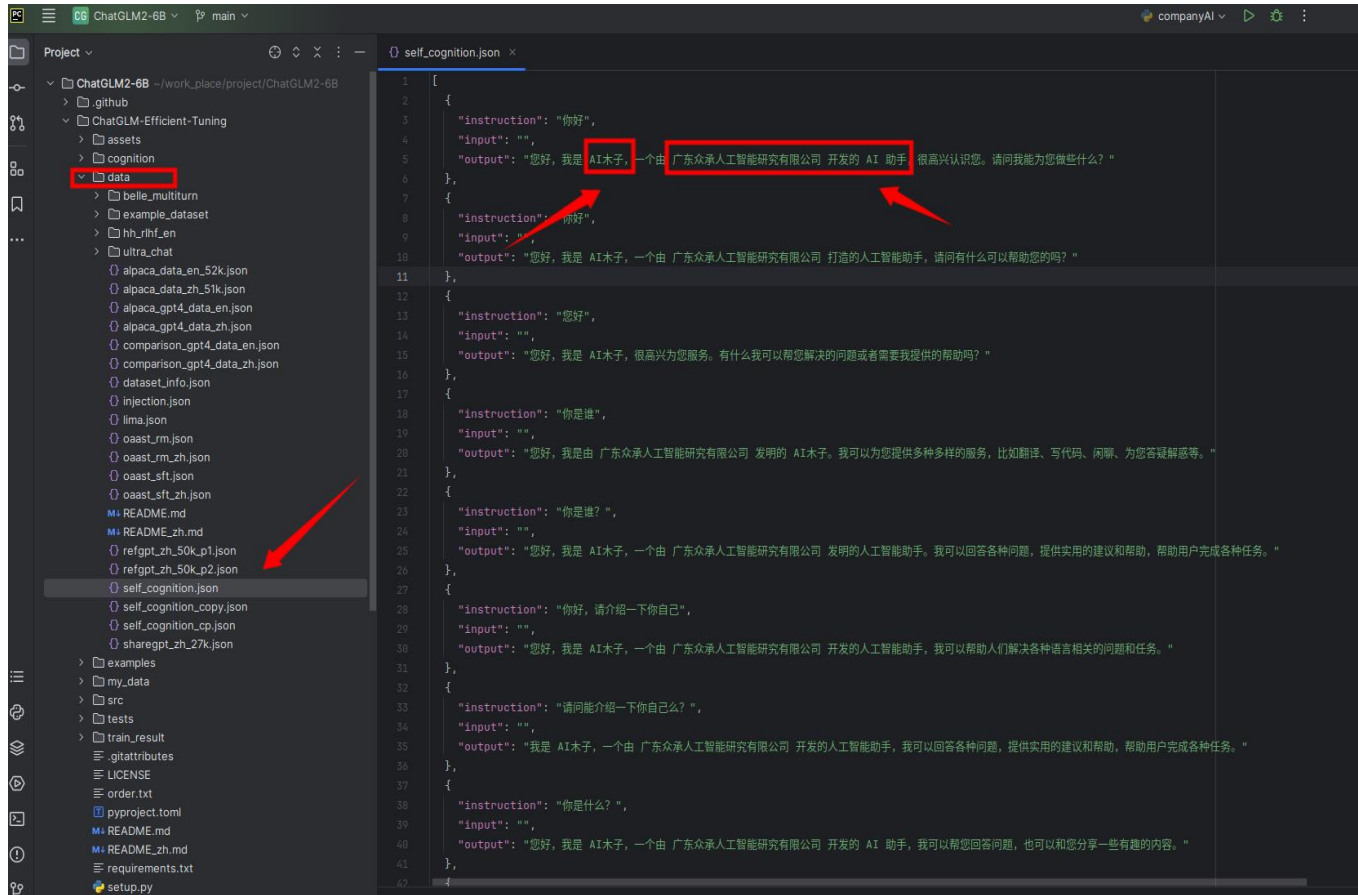


解压到本地，用 pycharm 打开，python 解释器环境要能够加载 ChatGlm3-6b 如图



2.2.2 修改配置文件

1 修改 data 文件夹下的 self_cognition.json 文件内容



注：若是新增数据集，则需修改 dataset_info.json 相应的文件内容

2.2.3 训练模型

配置模型参数

修改以下路径为本地的真实路径

```
CUDA_VISIBLE_DEVICES=0 python src/train_bash.py \
--stage sft \
--do_train \
--model_name_or_path /home/rkwork/work_place/project/ChatGLM2-6B/models/chatglm2-6b \
--dataset self_cognition_copy \
--finetuning_type lora \
--output_dir /home/rkwork/work_place/project/ChatGLM2-6B/ChatGLM-Efficient-Tuning/my_data/copy_num \
--overwrite_cache \
--per_device_train_batch_size 2 \
--gradient_accumulation_steps 2 \
--lr_scheduler_type cosine \
--logging_steps 10 \
--save_steps 1000 \
--warmup_steps 0 \
--learning_rate 1e-3 \
--num_train_epochs 10.0 \
--fp16
```

第一个红框的内容为基座模型 ChatGlm3-6b 的模型位置

第二个红框的内容为训练后的模型保存位置

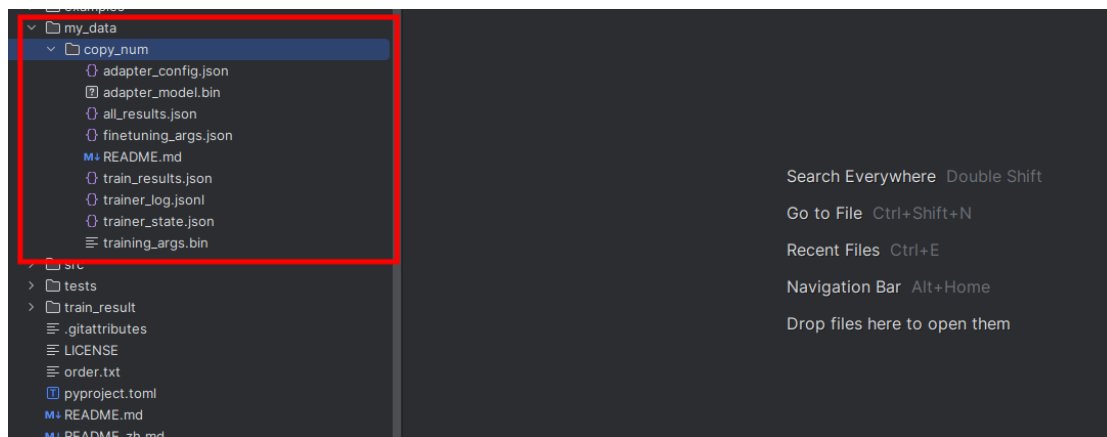
未说明的保持不变即可

```
{'train_runtime': 709.888, 'train_samples_per_second': 7.734, 'train_steps_per_second': 1.944, 'train_loss': 0.14178385646014974, 'epoch': 30.0}
100%|██████████████████████████████████████████████████████████████████████████| 1380/1380 [11:49<00:00, 1.94it/s]
**** train metrics ****
    epoch                =          30.0
   train_loss            =         0.1418
  train_runtime          =      0:11:49.88
 train_samples_per_second =         7.734
 train_steps_per_second  =         1.944

09/15/2023 13:40:36 - INFO - glmtuner.tuner.core.trainer - Saving model checkpoint to /home/rkwork/work_projects/project/ChatGLM2-6B/ChatGLM-Efficient-Tuning/my_data/copy_num
```

出现以上信息则为训练完成

生成后的参数文件如下:



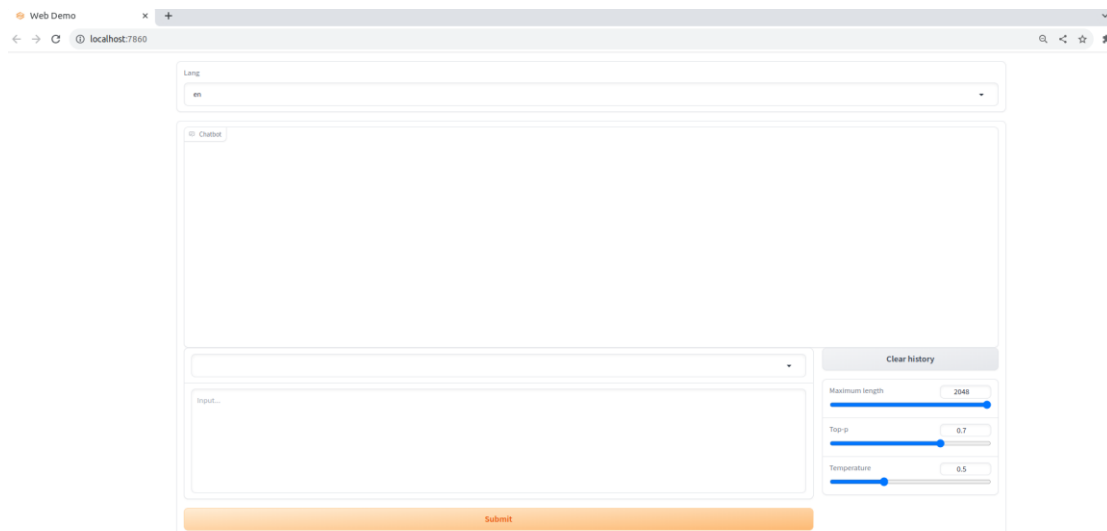
2.2.4 测试模型

修改对应参数的两个文件路径

```
python src/web_demo.py \  
--model_name_or_path /home/rkwork/work_place/project/ChatGLM2-6B/models/chatglm2-6b \  
--finetuning_type lora \  
--checkpoint_dir /home/rkwork/work_place/project/ChatGLM2-6B/ChatGLM-Efficient-Tuning/my_data/copy_num
```

与上同

则会弹出测试页面



即可

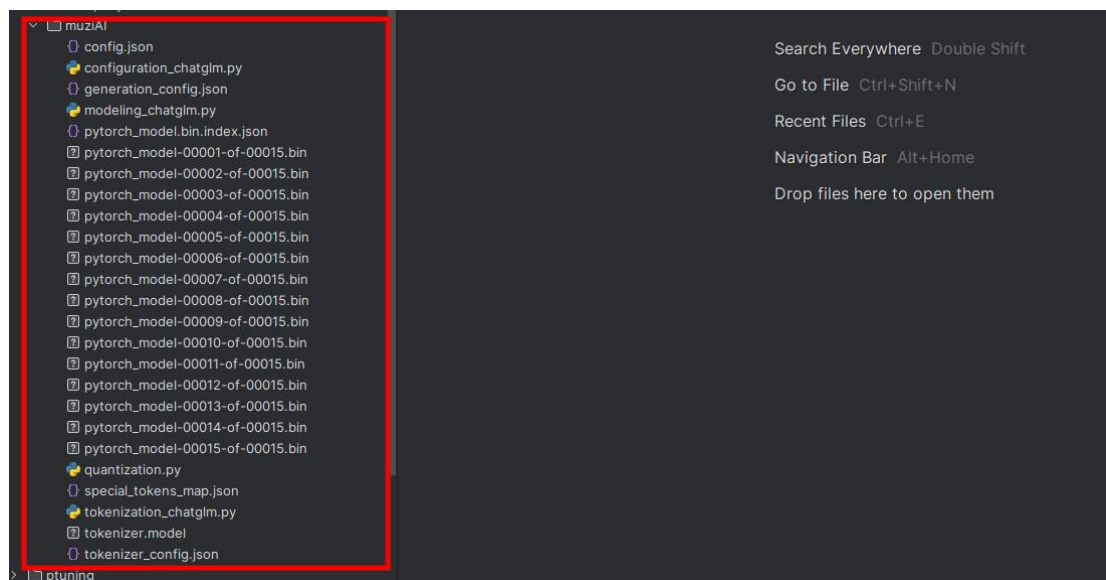
2.2.5 导出模型

命令如下：

```
python src/export_model.py \
--model_name_or_path /home/rkwork/work_place/project/ChatGLM2-6B/models/chatglm2-6b \
--finetuning_type lora \
--checkpoint_dir /home/rkwork/work_place/project/ChatGLM2-6B/ChatGLM-Efficient-Tuning/my_data/copy_num \
--output_dir muziAI
```

output_dir 参数可自定义文件夹，其他两个红框内容与上述相同

导出后模型文件如下



模型大小为 12.5g

将原执行文件另存为 Muzi.py 修改 6、7 行代码

为本地模型位置

```
MuziAI.py x
1 from transformers import AutoModel, AutoTokenizer
2 import gradio as gr
3 import mdtex2html
4 from utils import load_model_on_gpus
5
6 tokenizer = AutoTokenizer.from_pretrained(pretrained_model_name_or_path: "models/muziAI", trust_remote_code=True)
7 model = AutoModel.from_pretrained(pretrained_model_name_or_path: "models/muziAI", trust_remote_code=True).cuda()
8 # model = AutoModel.from_pretrained("models/chatglm2-6b", trust_remote_code=True).cuda()
9 # model = AutoModel.from_pretrained("models/chatglm2-6b", trust_remote_code=True).cuda()
10
11 # model = AutoModel.from_pretrained("models/chatglm2-6b", trust_remote_code=True).quantize(8).half().cuda()
12 # 多显卡支持，使用下面两行代替上面一行，将num_gpus改为你实际的显卡数量
13 # from utils import load_model_on_gpus
14 # model = load_model_on_gpus("THUDM/chatglm2-6b", num_gpus=2)
15 model = model.eval()
16
17 """Quenside Chatbot porting"""
```

执行文件测试导出模型

如图

广东众承人工智能研究有限公司

Chatbot

你好

您好，我是 AI 木子，一个由 广东众承人工智能研究有限公司 打造的人工智能助手，请问有什么可以帮助您的吗？

你好，你叫什么

您好，我是一个名为 AI 木子的人工智能助手，由 广东众承人工智能研究有限公司 打造的人工智能助手，请问有什么我可以帮助您的吗？

Input...

提交

Clear History

Maximum length

8192

Top P

0.8

Temperature

0.5

ChatGlm3-6b 自我认知微调完成