

ChatGLM3微调部署

有关ChatGLM3部署的文档。

1. 下载代码

```
1 $ git clone https://github.com/THUDM/ChatGLM3.git
```

2. 基础环境搭建

```
1 $ conda create -n chatglm_test python==3.10
2 $ conda activate chatglm_test
```

3. ChatGLM3环境搭建

```
1 $ pip3 install torch torchvision torchaudio -i
  https://pypi.tuna.tsinghua.edu.cn/simple --default-timeout=10000
2 # 可以先不安装，后续有说明
3 $ pip install peft -i https://pypi.tuna.tsinghua.edu.cn/simple
4 $ cd ChatGLM3
5 $ pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple
```

4. 模型下载（魔塔社区）

```
1 $ sudo apt update
2 $ sudo apt-get install git-lfs
3 $ git lfs install
4 $ git clone https://www.modelscope.cn/ZhipuAI/chatglm3-6b.git
```

可以删除部分模型（可选）：

```
1 $ rm -rf *.safetensors
2 $ rm -rf model.safetensors.index.json
```

5. Web APP (gradio)

修改路径：文件在./basic_demo/web_demo_gradio.py

代码片段：

```
1 MODEL_PATH = os.environ.get('MODEL_PATH', '你的路径')
```

运行Web Demo

```
1 $ python web_demo_gradio.py
```

第一次加载模型时间很长：

```
Loading checkpoint shards: 100% | 7/7 [00:10<00:00, 1.44s/it]
Setting eos_token is not supported, use the default one.
Setting pad_token is not supported, use the default one.
Setting unk_token is not supported, use the default one.
Running on local URL: http://127.0.0.1:7870

To create a public link, set `share=True` in `launch()`.
```

完成后出现：

ChatGLM3-6B Gradio Simple Demo

Chatbot

是的，国家电网的一线工人确实非常辛苦。他们承担着保障我国电力供应稳定的重要任务，无论风吹雨打、晴天雪天，都要坚守在岗位上，确保电力的正常供应。他们的付出和努力是我们应该尊重和感激的。同时，我们也要看到他们的工作条件得到了很大的改善，比如近年来国家电网投入了大量的资金进行设备升级和技术改造，一线工人的工作环境也得到了很大提升。

国家电网的一线工人非常辛苦

是的，国家电网的一线工人确实非常辛苦。他们承担着保障我国电力供应稳定的重要任务，无论风吹雨打、晴天雪天，都要坚守在岗位上，确保电力的正常供应。他们的付出和努力是我们应该尊重和感激的。同时，我们也要看到他们的工作条件得到了很大的改善，比如近年来国家电网投入了大量的资金进行设备升级和技术改造，一线工人的工作环境也得到了很大提升。

Submit

Clear History

Maximum length8192

Top P0.8

Temperature0.6

6. 微调模型

可能perf出现版本问题，导致模型微调后不能推理，可以改为0.7.0。

进入finetune_demo文件夹:

```
1 $ finetune_demo
2 $ sudo apt install mpich
3 $ sudo apt-get install openmpi-bin openmpi-doc libopenmpi-dev
4 $ env LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/usr/lib64:/usr/lib" MPICC=$(which
  mpicc) python -m pip install -v --no-cache-dir --upgrade mpi4py
5 $ export OMPI_CC=x86_64-conda_cos6-linux-gnu-cc
6 $ conda install gcc_linux-64 -c conda-forge
7 $ conda install openmpi -c conda-forge
8 $ pip install typer
9 $ pip install sentencepiece
10 $ pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple
```

(1) 下载与解压数据

```
1 $ wget https://cloud.tsinghua.edu.cn/f/b3f119a008264b1cabd1/?dl=1 -O
  AdvertiseGen.tar.gz
2 $ tar -xzf AdvertiseGen.tar.gz
```

(2) 查看数据

```
1 {"content": "类型#裤*材质#牛仔布*材质#蕾丝*颜色#黑色*风格#性感*图案#蕾丝*裤长#短裤",
  "summary": "牛仔短裤，黑色调，散发出性感撩人的迷人风情；简洁的腰头，显得腰线特别好看；裤
  脚边沿顺着蕾丝睫毛花边，微微蜿蜒的弧线婉约浪漫，整套look更显得精致了几分。"}

```

(3) 数据转换

查看ChatGLM3所需要的数据格式，在官方GitHub链接中：

https://github.com/THUDM/ChatGLM3/tree/main/finetune_demo

如果您仅希望微调模型的对话能力，而非工具能力，您应该按照以下格式整理数据。

```
[
  {
    "conversations": [
      {
        "role": "system",
        "content": "<system prompt text>"
      },
      {
        "role": "user",
        "content": "<user prompt text>"
      },
      {
        "role": "assistant",
        "content": "<assistant response text>"
      },
      // ... Muti Turn
      {
        "role": "user",
        "content": "<user prompt text>"
      },
      {
        "role": "assistant",
        "content": "<assistant response text>"
      }
    ]
  }
  // ...
]
```

创建数据格式转换文件:

```
1 $ touch data_process.py
```

将如下程序复制进创建的Python脚本中:

```
1 import json
2 from typing import Union
3 from pathlib import Path
4
5 def _resolve_path(path: Union[str, Path]) -> Path:
6     return Path(path).expanduser().resolve()
7
8 def _mkdir(dir_name: Union[str, Path]):
9     dir_name = _resolve_path(dir_name)
10    if not dir_name.is_dir():
11        dir_name.mkdir(parents=True, exist_ok=False)
12
13 def convert_adgen(data_dir: Union[str, Path], save_dir: Union[str, Path]):
14     def _convert(in_file: Path, out_file: Path):
15         _mkdir(out_file.parent)
16         with open(in_file, encoding='utf-8') as fin:
17             with open(out_file, 'wt', encoding='utf-8') as fout:
```

```

18         for line in fin:
19             dct = json.loads(line)
20             sample = {'conversations': [{'role': 'user', 'content':
dct['content']}],
21                       {'role': 'assistant',
'content': dct['summary']}]}}
22             fout.write(json.dumps(sample, ensure_ascii=False) + '\n')
23
24     data_dir = _resolve_path(data_dir)
25     save_dir = _resolve_path(save_dir)
26
27     train_file = data_dir / 'train.json'
28     if train_file.is_file():
29         out_file = save_dir / train_file.relative_to(data_dir)
30         _convert(train_file, out_file)
31
32     dev_file = data_dir / 'dev.json'
33     if dev_file.is_file():
34         out_file = save_dir / dev_file.relative_to(data_dir)
35         _convert(dev_file, out_file)
36
37     convert_adgen('data/AdvertiseGen', 'data/AdvertiseGen_fix')

```

运行代码：

```
1 $ python data_process.py
```

得到一个fix文件夹：

```

1 .
2 |— AdvertiseGen
3 |   |— dev.json
4 |   |— train.json
5 |— AdvertiseGen_fix
6 |   |— dev.json
7 |   |— train.json

```

里面是转换好的文件。

(5) 微调

修改peft>=0.10.0为peft=0.7.1。

mpi4py可以删除，但是执行上面的步骤也可以的。

需要安装nltk：

```
1 $ pip install nltk -i https://pypi.tuna.tsinghua.edu.cn/simple
```

查看并修改yaml文件（注释版本）：

```
1 data_config: # 数据配置部分
2   train_file: train.json # 训练数据文件
3   val_file: dev.json # 验证数据文件
4   test_file: dev.json # 测试数据文件
5   num_proc: 16 # 处理数据的进程数
6   max_input_length: 256 # 输入最大长度
7   max_output_length: 512 # 输出最大长度
8   training_args: # 训练参数配置
9     # 查看 `transformers.Seq2SeqTrainingArguments` 了解更多信息
10  output_dir: ./output # 输出目录
11  max_steps: 3000 # 训练的最大步数
12  # 根据数据集调整的学习率
13  learning_rate: 5e-5 # 学习率
14  # 数据加载设置
15  per_device_train_batch_size: 4 # 每个设备的训练批量大小
16  dataloader_num_workers: 16 # 数据加载的工作线程数
17  remove_unused_columns: false # 是否移除未使用的列
18  # 保存检查点的设置
19  save_strategy: steps # 保存策略：按步数
20  save_steps: 500 # 每500步保存一次
21  # 日志设置
22  log_level: info # 日志等级
23  logging_strategy: steps # 日志记录策略：按步数
24  logging_steps: 10 # 每10步记录一次日志
25  # 评估设置
26  per_device_eval_batch_size: 16 # 每个设备的评估批量大小
27  evaluation_strategy: steps # 评估策略：按步数
28  eval_steps: 500 # 每500步进行一次评估
29  # 优化器设置
30  # adam_epsilon: 1e-6 # Adam优化器的epsilon参数（未启用）
31  # 以下行用于检测nan或inf值（未启用）
32  # debug: underflow_overflow
33  predict_with_generate: true # 使用生成模式进行预测
34  # 查看 `transformers.GenerationConfig` 了解更多信息
35  generation_config: # 生成配置
36    max_new_tokens: 512 # 生成的最大新令牌数
```

```

37 # 在此处设置你的deepspeed配置路径（未启用）
38 #deepspeed: ds_zero_2.json
39 # 如果使用CPU训练，设置为true。
40 use_cpu: false
41 peft_config: # PEFT配置
42 peft_type: LORA # PEFT类型: LORA
43 task_type: CAUSAL_LM # 任务类型: 因果语言模型
44 r: 8 # LORA的r参数
45 lora_alpha: 32 # LORA的alpha参数
46 lora_dropout: 0.1 # LORA的dropout率

```

执行微调:

```

1 $ CUDA_VISIBLE_DEVICES=0 python3 finetune_hf.py data/AdvertiseGen_fix
/home/asic-zty/LLM/model_path/chatglm3-6b ./configs/lora.yaml

```

使用lora微调的参数:

```

trainable params: 1,949,696 || all params: 6,245,533,696 || t
> Model

```

训练中:

```

Number of trainable parameters = 1,949,696
{'loss': 4.8277, 'grad_norm': 2.195438861846924, 'learning_rate': 4.983333333333333e-05, 'epoch': 0.0}
{'loss': 4.5965, 'grad_norm': 3.1483213901519775, 'learning_rate': 4.966666666666667e-05, 'epoch': 0.0}
{'loss': 4.4844, 'grad_norm': 2.9855520725250244, 'learning_rate': 4.9500000000000004e-05, 'epoch': 0.0}
{'loss': 4.1195, 'grad_norm': 3.329874038696289, 'learning_rate': 4.933333333333334e-05, 'epoch': 0.0}
{'loss': 4.1193, 'grad_norm': 2.727184772491455, 'learning_rate': 4.9166666666666665e-05, 'epoch': 0.0}
{'loss': 3.8691, 'grad_norm': 2.9155259132385254, 'learning_rate': 4.9e-05, 'epoch': 0.0}
{'loss': 3.8428, 'grad_norm': 2.8598265647888184, 'learning_rate': 4.883333333333334e-05, 'epoch': 0.0}
{'loss': 3.7475, 'grad_norm': 2.896915912628174, 'learning_rate': 4.866666666666667e-05, 'epoch': 0.0}
{'loss': 3.6332, 'grad_norm': 3.1633598804473877, 'learning_rate': 4.85e-05, 'epoch': 0.0}
{'loss': 3.7197, 'grad_norm': 3.313730239868164, 'learning_rate': 4.833333333333334e-05, 'epoch': 0.0}
{'loss': 3.6715, 'grad_norm': 3.5494039058685303, 'learning_rate': 4.816666666666667e-05, 'epoch': 0.0}
{'loss': 3.8504, 'grad_norm': 3.8452370166778564, 'learning_rate': 4.8e-05, 'epoch': 0.0}
{'loss': 3.6117, 'grad_norm': 3.4548428058624268, 'learning_rate': 4.7833333333333335e-05, 'epoch': 0.0}
{'loss': 3.7344, 'grad_norm': 4.391856670379639, 'learning_rate': 4.766666666666667e-05, 'epoch': 0.0}
{'loss': 3.6855, 'grad_norm': 3.617318630218506, 'learning_rate': 4.75e-05, 'epoch': 0.01}
5% | 156/3000 [01:28<27:12, 1.74it/s]

```

7. 测试

训练后模型文件夹如下:

```

1 .
2 |— adapter_config.json # 适配器配置文件，包含适配器模块的结构和参数设置
3 |— adapter_model.safetensors # 适配器模型的权重文件，使用 SafeTensors 格式存
   储，用于确保数据的一致性和安全
4 |— optimizer.pt # 优化器状态文件，存储优化器的当前状态，以便恢复训练
   时使用
5 |— README.md # 一般包含模型和训练过程的说明或其他重要信息
6 |— rng_state.pth # 随机数生成器的状态文件，用于确保模型训练可复现

```

```

7 |— scheduler.pt # 学习率调度器状态文件，存储调度器的当前状态，用于控制学习率的变化
8 |— trainer_state.json # 训练器状态文件，包含训练过程中的各种统计信息和指标
9 |— training_args.bin # 训练参数的二进制文件，存储了训练过程中使用的参数设置

```

我们在 ./output/checkpoint-500/adapters_config.json 文件中看下配置文件：

```

1 {
2   "alpha_pattern": {}, // alpha参数的模式配置，用于调整LORA层的特定参数
3   "auto_mapping": null, // 自动映射配置，通常用于自动识别并映射模型的部分配置
4   "base_model_name_or_path": "/model_path/chatglm3-6b", // 基础模型的名称或路径，指定预训练模型的存放位置
5   "bias": "none", // 偏置配置，指定是否在LORA适配中添加偏置项
6   "fan_in_fan_out": false, // 是否启用fan-in和fan-out初始化方法
7   "inference_mode": true, // 推理模式，指示模型是否处于推理状态
8   "init_lora_weights": true, // 是否初始化LORA权重，开启后将按照指定方法初始化权重
9   "layers_pattern": null, // 层模式配置，用于定义哪些层应用特定的模式
10  "layers_to_transform": null, // 需要转换的层列表，指定哪些层应用LORA或其他PEFT技术
11  "loftq_config": {}, // LOFT-Q技术的配置，用于定量化和优化训练过程
12  "lora_alpha": 32, // LORA的alpha参数，控制学习率的调整
13  "lora_dropout": 0.1, // LORA适配器中的dropout比率，有助于防止过拟合
14  "megatron_config": null, // Megatron配置，用于大型模型的特定优化设置
15  "megatron_core": "megatron.core", // 指定Megatron库的核心模块路径
16  "modules_to_save": null, // 指定需要保存的模块，用于模型持久化时选择性保存
17  "peft_type": "LORA", // 指定PEFT（参数高效微调）的类型，这里使用的是LORA
18  "r": 8, // LORA的rank参数，决定了适配器矩阵的秩大小
19  "rank_pattern": {}, // rank参数的模式配置
20  "revision": null, // 版本控制，用于模型的版本管理
21  "target_modules": [
22    "query_key_value" // 目标模块列表，指定LORA适配器应用于哪些模块
23  ],
24  "task_type": "CAUSAL_LM" // 任务类型，这里是因果语言模型（Causal Language Model）
25 }

```

测试：


```
1 $ python inference_hf.py ./output/checkpoint-500 --prompt "牛仔裤是什么?"
```

效果如下：

```
loading checkpoint shards: 100%|
Setting eos_token is not supported, use the default one.
Setting pad_token is not supported, use the default one.
Setting unk_token is not supported, use the default one.
牛仔外套，又称牛仔衣，是一种常见的上装，通常由牛仔布料制成。它以牛仔布为主要面料，以 Threadneedle threads缝制而成。牛仔外套的设计简单，造型百搭，深受广大年轻人的喜爱。它既可单穿，也可与其它衣服搭配，具有很高的时尚指数。
```

牛仔外套有很多款式，有短款、长款、马甲款等，可以满足不同场合和个人喜好。它不仅具有防风保暖的功能，而且可以增加整体造型层次感。在春天和秋天，一件牛仔外套是衣橱的必备单品。

近年来，随着时尚潮流的发展，牛仔外套的设计越来越多样化，从简单的单色款式到花哨的彩色款式，从简单的剪裁到复杂的拼接，从简单的纯色到花哨的印花，从简单的款式到复杂的立体剪裁，从简单的剪裁到复杂的立体剪裁。各种款式和元素的加入，让牛仔外套更加丰富多样，时尚百搭。

在搭配方面，牛仔外套可以与众多服饰搭配，如牛仔褲、长褲、裙子等。此外，还可以搭配围巾、帽子、手套等配件，增加整体时尚感。总之，一件牛仔外套可以百搭各种场合，是时尚潮人的必备单品。