多模态模型部署微调（visualglm-6b）

笔记本：	我的第一个笔记本		
创建时间：	2024/12/9 9:53	**更新时间：**	2024/12/9 12:18
作者：	耿介		
URL：	about:blank		

一.部署

首先，官方的项目有大量的环境问题和配置问题，并且没有chatglm的分词器配置文件，因此我将修改好的代码放置在了github上，使用时在合适的文件位置直接git clone就可以

```
git clone https://github.com/JieGeng1998-10-13/visualglm_Jie.git
```

然后创建一个新的环境。

```
conda create -n visualglm python==3.10
```

依旧选择3.10的版本保证稳定性

然后激活环境

```
conda activate visualglm
```

然后进入visualglm，安装需要的依赖包

```
pip install -r requirements.txt
```

注意，假如默认安装的transformers是较新的版本，务必回退到我下面展示的版本，否则会有环境问题

```
(visualglm) root@iZbp1aqj1mwinktybhbmeeZ:/mnt/workspace/visualglm_Jie# pip show transformers
Name: transformers
Version: 4.27.1
```

这些全部完成后即可使用，

后台交互模式：

```
python cli_demo.py
```

ui交互模式：

```
python web_demo.py --share
```

这个share用作创建公网链接便于访问，界面如下，成功显示说明部署就成功啦

二.微调数据准备

准备好dataset.json放入fewshot-data文件夹下面，格式大概如下

dataset.json

```
[
    {
        "img": "/mnt/workspace/datasets/威平 WPS-80 液压渣浆泵.jpg",
        "label": "威平液压渣浆泵",
        "prompt": "这个商品大概是什么品牌什么参数？"
    },
    {
        "img": "/mnt/workspace/datasets/富世华建筑产品 LT 6005 冲击夯.jpg",
        "label": "富世华建筑产品冲击夯",
        "prompt": "这个商品大概是什么品牌什么参数？"
    },
    {
        "img": "/mnt/workspace/datasets/临工重机 MT86D 矿用卡车.jpg",
        "label": "临工重机矿用卡车",
        "prompt": "这个商品大概是什么品牌什么参数？"
    },
    {
        "img": "/mnt/workspace/datasets/海天路矿 LKB618 安装钻杆现场.jpg",
        "label": "海天路矿安装钻杆现场",
        "prompt": "这个商品大概是什么品牌什么参数？"
    },
    {
        "img": "/mnt/workspace/datasets/洛阳路通 LTS728H 全液压单驱振动压路机.jpg",
        "label": "洛阳路通全液压单驱振动压路机",
        "prompt": "这个商品大概是什么品牌什么参数？"
    },
    {
        "img": "/mnt/workspace/datasets/三一重工 SY410C-8（ VI ）搅拌运输车.jpg",
        "label": "三一重工搅拌运输车",
        "prompt": "这个商品大概是什么品牌什么参数？"
    },
    {
        "img": "/mnt/workspace/datasets/山推 SG21-3 平地机.jpg",
        "label": "山推平地机",
        "prompt": "这个商品大概是什么品牌什么参数？"
```

"img"键值对存放的是图片的绝对路径，在该路径下要准备好对应的图片

*注意：finetune_visualglm的文件在路径的配置上也要修改，这里不再赘述

三.模型微调

在finetune文件夹中找到finetune_visualglm.sh，然后修改配置文件如下，

```
#! /bin/bash
NUM_WORKERS=1
NUM_GPUS_PER_WORKER=8
MP_SIZE=1


script_path=$(realpath $0)
script_dir=$(dirname $script_path)
main_dir=$(dirname $script_dir)
MODEL_TYPE="visualglm-6b"
MODEL_ARGS="--max_source_length 64 \
    --max_target_length 256 \
    --lora_rank 16 \
    --layer_range 0 14 \
    --pre_seq_len 4"
```

```
OPTIONS_SAT="SAT_HOME=/root/.sat_models" #"SAT_HOME=/raid/dm/sat_models"
OPTIONS_NCCL="NCCL_DEBUG=info NCCL_IB_DISABLE=0 NCCL_NET_GDR_LEVEL=2"
HOST_FILE_PATH="hostfile"
HOST_FILE_PATH="hostfile_single"


train_data="./fewshot-data/dataset.json"
eval_data="./fewshot-data/dataset.json"



gpt_options=" \
       --experiment-name finetune-$MODEL_TYPE \
       --model-parallel-size ${MP_SIZE} \
       --mode finetune \
       --train-iters 500000 \
       --resume-dataloader \
       $MODEL_ARGS \
       --train-data ${train_data} \
       --valid-data ${eval_data} \
       --distributed-backend nccl \
       --lr-decay-style cosine \
       --warmup .02 \
       --checkpoint-activations \
       --save-interval 500000 \
       --eval-interval 500000 \
       --save "./checkpoints" \
       --split 1 \
       --eval-iters 10 \
       --eval-batch-size 8 \
       --zero-stage 1 \
       --lr 0.0001 \
       --batch-size 4 \
       --skip-init \
       --fp16 \
       --use_lora
"




run_cmd="${OPTIONS_NCCL} ${OPTIONS_SAT} deepspeed --master_port 16666 --hostfile
${HOST_FILE_PATH} finetune_visualglm.py ${gpt_options}"
echo ${run_cmd}
eval ${run_cmd}


set +x
```

回到主文件夹，启动微调脚本

```
bash finetune/finetune_visualglm.sh
```

微调后的文件保存在checkpoints文件夹中，
需要使用这个权重文件，进入model文件夹，找到infer_util.py
在get_infer_setting函数中修改路径，使用微调后的模型：

```
def get_infer_setting(gpu_device=0, quant=None):
    os.environ['CUDA_VISIBLE_DEVICES'] = str(gpu_device)
    args = argparse.Namespace(
        fp16=True,
        skip_init=True,
```

```
        device='cuda' if quant is None else 'cpu',
    )
    #model, args = VisualGLMModel.from_pretrained('visualglm-6b', args)
    #model, args =
VisualGLMModel.from_pretrained('/mnt/workspace/visualglm_Jie/checkpoints/finetune-
visualglm-6b-11-27-10-23', args)
    #finetuned_model = "/mnt/workspace/visualglm_Jie/checkpoints/finetune-
visualglm-6b-11-29-13-08"
    finetuned_model = "/mnt/workspace/visualglm_Jie/checkpoints/finetune-
visualglm-6b-12-02-14-48"
    #/mnt/workspace/visualglm_Jie/checkpoints/finetune-visualglm-6b-12-02-14-48
    from finetune_visualglm import FineTuneVisualGLMModel
    model, args = FineTuneVisualGLMModel.from_pretrained(finetuned_model, args)
    model.add_mixin('auto-regressive', CachedAutoregressiveMixin())
    assert quant in [None, 4, 8]
    if quant is not None:
        quantize(model.transformer, quant)
    model.eval()
    model = model.cuda()
    tokenizer = AutoTokenizer.from_pretrained("./chatglm",
trust_remote_code=True)
    return model, tokenizer
```

完成后回去重新启动web_demo.py 即可使用微调后的权重了

四.微调结果

第一次微调结果

```
[2024-11-30 11:59:48,421] [INFO] [RANK 0] time (ms) | forward:
329.99 | backward: 486.33 | allreduce: 0.00 | optimizer: 4.39 |
batch generator: 2.31 | data loader: 0.31
[2024-11-30 12:00:29,552] [INFO] [RANK 0] iteration 100000/
100000 | elapsed time per iteration (ms): 822.6 | learning rate
1.013E-05 | total loss 9.703857E-01 | loss 9.703857E-01 | loss
scale 32768.0 |speed 291.75 samples/(min*GPU)
```

这个loss是不够满意的

第二次微调结果

```
[2024-12-07 09:07:35,916] [INFO] [RANK 0] iteration 500000/ 500000
| elapsed time per iteration (ms): 822.4 | learning rate 1.013E-05 |
total loss 4.205371E-01 | loss 4.205371E-01 | loss scale 32768.0
|speed 291.83 samples/(min*GPU)
```

loss大概是0.4的水平，降了一星期怎么也降不下去了

五.模型评估

评测数据和商品检索一样，采用实景图片加翻转加噪声的组合
评估指标：

字符串相似度(Similarity)：
若输出与真实标签的直接匹配有困难，可以通过编辑距离(Levenshtein Distance)、Jaccard相似度、余弦相似度(基于词向量或句向量)等方式计算模型输出与真实标签短语的文本相似度。当相似度超过某个预定义阈值（如0.2）时，即判定该输出为正确识别。

经过测试，相似度大致分为以下三种情况：

品牌和产品都答错了：相似度<0.2
品牌和产品答对了其中一个： 相似度>=0.2
品牌和产品全部答对:相似都>0.5

因此把0.2作为正确阈值

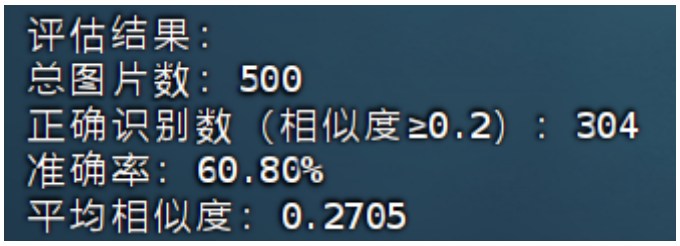
评估代码如下:

```
import os
import difflib
import torch
from PIL import Image
from web_demo import generate_input, chat, get_infer_setting, is_chinese
# 假设 web_demo.py 中已定义了 model, tokenizer 的初始化函数 get_infer_setting
# 请确保已运行: model, tokenizer = get_infer_setting(gpu_device=0, quant=None) 之类的初始化操作


def compute_string_similarity(str1: str, str2: str) -> float:
    """
    使用SequenceMatcher计算两个字符串之间的相似度，返回0-1之间的浮点数。
    """
    return difflib.SequenceMatcher(None, str1, str2).ratio()

def generate_text_with_image(input_text, image, history=[], request_data=dict(), is_zh=True):
    """
    与 web_demo.py 中一致的函数，根据需要进行调整
    """
    input_para = {
        "max_length": 2048,
        "min_length": 50,
        "temperature": 0.8,
        "top_p": 0.4,
        "top_k": 100,
        "repetition_penalty": 1.2
    }
    input_para.update(request_data)
```

```
    input_data = generate_input(input_text, image, history, input_para,
image_is_encoded=False)
    input_image, gen_kwargs = input_data['input_image'], input_data['gen_kwargs']
    with torch.no_grad():
        answer, history, _ = chat(None, model, tokenizer, input_text,
history=history, image=input_image, \
                            max_length=gen_kwargs['max_length'],
top_p=gen_kwargs['top_p'], \
                            top_k = gen_kwargs['top_k'],
temperature=gen_kwargs['temperature'], english=not is_zh)
    return answer


def single_image_inference(image_path, prompt):
    """
    对单张图片进行推理，返回模型输出字符串
    """
    image = Image.open(image_path)
    # 假设只有单轮对话，历史为空
    history = []
    # 使用中文还是英文判断
    is_zh = is_chinese(prompt)
    request_para = {"temperature": 0.8, "top_p": 0.4}
    answer = generate_text_with_image(prompt, image, history, request_para,
is_zh)
    return answer


if __name__ == "__main__":
    # 初始化模型和分词器（如果在web_demo中未全局初始化，这里需要先初始化）
    model, tokenizer = get_infer_setting(gpu_device=0, quant=None)


    data_dir = "/mnt/上海市交通系统交易问答框架/Figure_search/datatest"
    # 标准提示语
    prompt = "请根据这张图片识别图片中的商品，并描述它的主要特征。"
    threshold = 0.2  # 相似度判定阈值


    image_files = [f for f in os.listdir(data_dir) if f.lower().endswith(('.png',
'.jpg', '.jpeg'))]
    total_count = 0
    correct_count = 0
    total_similarity = 0.0


    for img_file in image_files:
        # ground truth从文件名中提取（去掉扩展名）
        ground_truth = os.path.splitext(img_file)[0]
        image_path = os.path.join(data_dir, img_file)

        # 模型推理
        model_output = single_image_inference(image_path, prompt)


        # 计算相似度
        similarity = compute_string_similarity(ground_truth, model_output)
        total_similarity += similarity
        total_count += 1


        # 判断是否达标
        if similarity >= threshold:
            correct_count += 1


        print(f"Image: {img_file}")
        print(f"Ground Truth: {ground_truth}")
        print(f"Model Output: {model_output}")
        print(f"Similarity: {similarity:.4f}")
```

```
        print("-" * 50)


    # 统计结果
    accuracy = correct_count / total_count if total_count > 0 else 0
    avg_similarity = total_similarity / total_count if total_count > 0 else 0


    print("评估结果：")
    print(f"总图片数：{total_count}")
    print(f"正确识别数（相似度≥{threshold}）：{correct_count}")
    print(f"准确率：{accuracy:.2%}")
    print(f"平均相似度：{avg_similarity:.4f}")
```

loss为0.9的评估结果



loss为0.4的评估结果