CPM-Bee 微调实战

Background

CPM-BEE

模型介绍

CPM-Bee是一个完全开源、允许商用的百亿参数中英文基座模型，也是**CPM-Live**训练的第二个里程碑。它采用Transformer自回归架构（auto-regressive），在超万亿（trillion）高质量语料上进行预训练，拥有强大的基础能力。开发者和研究者可以在CPM-Bee基座模型的基础上在各类场景进行适配来以创建特定领域的应用模型。

- 👐 **开源可商用**：OpenBMB始终秉承“让大模型飞入千家万户”的开源精神，CPM-Bee基座模型将完全开源并且可商用，以推动大模型领域的发展。我们鼓励全球范围内的科研机构、企业和个人开发者在遵守开源许可协议的前提下，自由地在CPM-Bee基座模型上进行创新。
- 💫 **中英双语性能优异**：CPM-Bee基座模型在预训练语料上进行了严格的筛选和配比，同时在中英双语上具有亮眼表现，具体可参见评测任务和结果。
- 📖 **超大规模高质量语料**：CPM-Bee基座模型在超万亿语料进行训练，是开源社区内经过语料最多的模型之一。同时，我们对预训练语料进行了严格的筛选、清洗和后处理以确保质量。
- **OpenBMB大模型系统生态支持**：OpenBMB大模型系统围绕高性能预训练、适配、压缩、推理开发了一系列工具，CPM-Bee基座模型将配套所有的工具脚本，高效支持开发者进行进阶使用。
- 🔨 **对话和工具使用能力**： 结合OpenBMB在指令微调和工具学习的探索，我们在CPM-Bee基座模型的基础上进行微调，训练出了具有强大对话和工具使用能力的实例模型，API和内测将于近期开放。

实例硬件参数：Amazon SageMaker平台，主要使用4块 NVIDIA A10-24GB GDDR6训练和部署。

1. 部署CPM-BEE

主要目的就是类似安装应用一样，通过命令行安排CPM-Bee的安装环境并且将CPM-Bee本体下载到本地

1.1 安装需要的模型以及依赖

```
!pip install --force-reinstall torch==1.11.0+cu113 --extra-index-url https://dow
```

即安装cuda和pytorch 而二者的版本号都内含在命令行中

```
!rm -rf CPM-Bee
!git clone https://github.com/OpenBMB/CPM-Bee.git
```

直接从github官网克隆下来CPM-Bee的本体代码

```
#%%script bash
!export TORCH_CUDA_ARCH_LIST="6.0 6.1 7.0 7.5 8.0 8.6+PTX"
#!export SKLEARN_ALLOW_DEPRECATED_SKLEARN_PACKAGE_INSTALL=True
!pip install --upgrade scikit-learn
!pip install -r CPM-Bee/src/requirements.txt
```

之后通过CPM内置的requirements安装各种所需的库文件

之后就全部安装完成了！

2. 微调CPM-Bee

微调也就是用特殊数据集对原模型进行进一步的训练已达到帮助模型优化特定功能的作用

在本实例中用的是一段诗歌含义的数据集，主要通过微调大模型达到一个古文诗词含义理解的深化优化

2.1 数据集类型介绍

各种数据集数据形式：

1）文本生成

{**"input"**: "今天天气不错，", **"prompt"**:"往后写100字", **"<ans>"**:""} **input字段用于填写上下文**，可以使用"source", "document", "query", "text"、等类似的键来替换。

prompt字段用来给出一些提示和指定任务。prompt也可以被"hint", "task", "prompt", "任务", "提示", "目标", "target"等替换。

2）翻译

{**"input"**: "今天天气不错，", **"prompt"**:"**中翻英**", "<ans>":""}

prompt一般可选"中翻英"/"英翻中"，"中译英"/"英译中"，"把文章翻译为英文"/"把文章翻译为中文"，"Translate from English to Chinese"

3）问答

{**"input"**: "今天天气不错，", **"prompt"**:"问答", **"question"**: "今天天气怎么样", **"<ans>"**:""}

4）选择题

{**"input"**: "今天天气不错，", **"prompt"**:"选择题", **"question"**: "今天天气怎么样", **"options"**: {"<option_0>": "好", "<option_1>": "坏"}, **"<ans>"**:""} options可以等价替换为"answers", "candidates"

通过对于训练数据集本身的格式调整可以使得finetune微调更加好的驱动模型，使得CPM本身更好的适应特定应用环境。

实际数据集载入和编译：

CPM数据集分为三部分： **/raw data /bee data /bin data**

载入原始数据集，如ShareGPT到**/raw data**，通过一个python文件规范格式载入**/bee data**，再将规范数据二进制化为**/bin data**，方便模型读取

/raw data（也即本实例中采用的原有数据集）

{"target": "1", "input": "[翻译]月夜朦胧，用碧玉做成的栏杆和用红色的砖砌成的墙是刺史的府宅。[0]兰玉当年刺史家[1]碧砌红轩刺史家[2]碧殿红棂翠浪间[3]一簇楼台刺史家[答案]"}

/bee data（也即本实例中采用的格式化数据集）

{"input": "月夜朦胧，用碧玉做成的栏杆和用红色的砖砌成的墙是刺史的府宅。", "options": {"<option_0>": "兰玉当年刺史家", "<option_1>": "碧砌红轩刺史家", "<option_2>": "碧殿红棂翠浪间", "<option_3>": "一簇楼台刺史家"}, "question": "这段话形容了哪句诗的意境？", "<ans>": "<option_1>"}

2.2 原始训练的基本大模型的载入和数据集格式化

```
!rm -rf CPM-Bee/src/10B-ckpts/
!mkdir CPM-Bee/src/10B-ckpts/
```

下载CPM-Bee 10B类基础大模型

```
!pip install huggingface_hub
```

安装huggingface 安装库（也即大模型“应用市场”）

```
from huggingface_hub import snapshot_download

snapshot_download(repo_id="openbmb/cpm-bee-10b", local_dir = 'CPM-Bee/src/10B-ck
                  allow_patterns=["*.bin", "*.json", "*.txt"])
```

在huggingface“应用市场”中载入原始大模型CPM-Bee 10B（Billion）

```
%%sh
cd CPM-Bee/tutorials/basic_task_finetune && python data_reformat.py
```

通过 python data_reformat.py将原有的诗句格式转化为上述数据集类型

以本例子来说就是将原有的诗句翻译做成了严格的选择题格式，详情见上述数据集格式阐述

```
!rm -rf CPM-Bee/tutorials/basic_task_finetune/bin_data
!mkdir CPM-Bee/tutorials/basic_task_finetune/bin_data
!rm -rf tmp
!rm -rf CPM-Bee/tutorials/basic_task_finetune/bee_data/.ipynb_checkpoints
```

转化上述Bee data数据集为二进制数据集

```
finetune_path = "CPM-Bee/tutorials/basic_task_finetune"
%%script env finetune_path=$finetune_path bash
python CPM-Bee/src/preprocess_dataset.py --input ${finetune_path}/bee_data --out
```

规定微调地址和路径，做好最后的准备

2.3 开始训练

下面为微调最主要的参数表，也即超参数，其中OPTS后面接着的都是可以根据实际情况调节的超参数（类似于旋钮，调节各个参数大小），简单介绍其中较重要的：

微调方式（增量，全量）：主要区别在于是否更新原有数据集

Epoch数：决定输入数据次数，更多有助于快速收敛，但可能出现过拟合，并且过多Epoch可能消耗过多资源

Batch_size：每次模型更新时所使用的训练样本数量，越大的size需要的内存更多，合适大小的size可以促进样本的收敛

learing-rate：决定了在每次参数更新中，模型权重需要根据损失函数的梯度调整多少，小学习率促进稳定，大学习率促进收敛速度

```
%%writefile CPM-Bee/src/scripts/finetune_cpm_bee.sh
#! /bin/bash

# 四卡微调
export CUDA_VISIBLE_DEVICES=0,1,2,3
GPUS_PER_NODE=4

NNODES=1
MASTER_ADDR="localhost"
MASTER_PORT=12346

OPTS=""
OPTS+=" --use-delta"  # 使用增量微调（delta-tuning）
OPTS+=" --model-config CPM-Bee/src/config/cpm-bee-10b.json"  # 模型配置文件
OPTS+=" --dataset CPM-Bee/tutorials/basic_task_finetune/bin_data/train"  # 训练集
OPTS+=" --eval_dataset CPM-Bee/tutorials/basic_task_finetune/bin_data/eval"  # 验
OPTS+=" --epoch 1"  # 训练epoch数
OPTS+=" --batch-size 3"    # 数据批次大小
OPTS+=" --train-iters 100"  # 用于lr_schedular
OPTS+=" --save-name cpm_bee_finetune"  # 保存名称
OPTS+=" --max-length 2048" # 最大长度
OPTS+=" --save results/"  # 保存路径
OPTS+=" --lr 0.0001"    # 学习率
OPTS+=" --inspect-iters 100"  # 每100个step进行一次检查(bmtrain inspect)
OPTS+=" --warmup-iters 1" # 预热学习率的步数为1
OPTS+=" --eval-interval 50"  # 每50步验证一次
OPTS+=" --early-stop-patience 5"  # 如果验证集loss连续5次不降，停止微调
OPTS+=" --lr-decay-style noam"  # 选择noam方式调度学习率
OPTS+=" --weight-decay 0.01"  # 优化器权重衰减率为0.01
OPTS+=" --clip-grad 1.0"  # 半精度训练的grad clip
OPTS+=" --loss-scale 32768"  # 半精度训练的loss scale
OPTS+=" --start-step 0"  # 用于加载lr_schedular的中间状态
OPTS+=" --load CPM-Bee/src/10B-ckpts/pytorch_model.bin"  # 模型参数文件

CMD="torchrun --nnodes=${NNODES} --nproc_per_node=${GPUS_PER_NODE} --rdzv_id=1 -

echo ${CMD}
$CMD
```

通过sh命令意见开始训练

```
!sh CPM-Bee/src/scripts/finetune_cpm_bee.sh
```

2.4 模型推理

按照如下代码尝试微调后的模型，这种是通过--delta模式和基础模式训练出来的两种微调模型结果，可以看出第二种效果明显更加优秀，更能理解古文诗句的含义：

```
%%sh
cd CPM-Bee/src/ && python text_generation.py --delta '../../results/cpm_bee_fine
```

{'input': '昏暗的灯熄灭了又被重新点亮。', 'options': {'<option_0>': '渔灯灭复明', '<option_1>': '残灯灭又然', '<option_2>': '残灯暗复明', '<option_3>': '残灯灭又明'}, 'question': '这段话形容了哪句诗的意境？', '<ans>': '<option_3>'}

```
%%sh
cd CPM-Bee/src/ && python text_generation.py
```

{'input': '昏暗的灯熄灭了又被重新点亮。', 'options': {'<option_0>': '渔灯灭复明', '<option_1>': '残灯灭又然', '<option_2>': '残灯暗复明', '<option_3>': '残灯灭又明'}, 'question': '这段话形容了哪句诗的意境？', '<ans>': '<option_2>'}

总结

通过增量微调的方式改善了CPM-Bee的中文诗词理解的能力，使得这个大模型在中文古文领域能够给出较为准确的判断。而微调大模型实际上也可以通俗理解为将一位平庸的全科医生，训练成一位有专长的专科医生，通过微调技术，相信大模型一定能在更多专精的领域发挥更大的作用！