- **关键功能组件**：自注意力机制是Transformer模型的核心组件，对模型性能影响重大。对这些矩阵进行低秩近似可以显著影响模型的表达能力。
- **参数量大**：这些投影矩阵包含大量参数，使用LoRA可以减少参数数量，降低计算和存储需求。

4.6.3 嵌入层

位置：

在NLP任务中，嵌入层用于将离散的词汇表映射到连续的向量空间。

原因：

- **高维稀疏表示**：嵌入层通常包含大量高维向量，通过LoRA可以有效降低维度，减少计算量和内存占用。
- **提升训练效率**：低秩分解可以使嵌入层的训练更加高效。

5. 案例实战

```
# 本机实验环境
1. ubuntu20.04
2. Python 3.10.14
3. pytorch 1.13.0
4. CUDA
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
```

01_LoRA_案例实战.ipynb

02_LoRA_微软开源项目

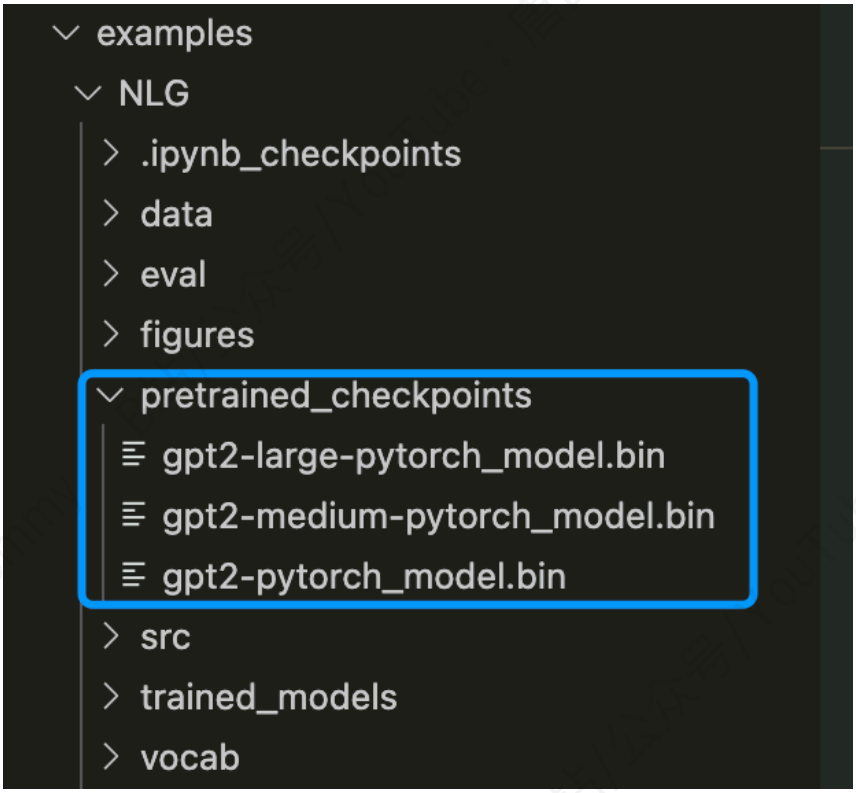
第1步：环境配置

```
conda create -n lora python=3.10
# conda init bash && source /root/.bashrc
conda activate lora
# conda install ipykernel
# ipython kernel install --user --name=lora   # 设置kernel，--user表示当前用户，lora为虚拟环境名称
```

```
pip install torch==1.13.0
pip install progress
pip install transformers==4.39.3 deepspeed accelerate datasets==2.18.0 peft bitsandbytes


# 安装loralib
# 方式-1
pip install loralib

## 方式-2（开发者模式）
pip install -e .
```

第2步：下载预训练模型

```
# 在路径：/LoRA/examples/NLG/ 下运行命令

bash download_pretrained_checkpoints.sh
```



第3步：下载训练数据集

```
# 在路径：/LoRA/examples/NLG/ 下运行命令

bash create_datasets.sh
```

第4步：在数据集E2E上训练

```
python -m torch.distributed.launch --nproc_per_node=1 src/gpt2_ft.py \
    --train_data ./data/e2e/train.jsonl \
    --valid_data ./data/e2e/valid.jsonl \
    --train_batch_size 8 \
    --grad_acc 1 \
    --valid_batch_size 4 \
    --seq_len 512 \
    --model_card gpt2.md \
    --init_checkpoint ./pretrained_checkpoints/gpt2-medium-pytorch_model.bin \
    --platform local \
    --clip 0.0 \
    --lr 0.0002 \
    --weight_decay 0.01 \
    --correct_bias \
    --adam_beta2 0.999 \
    --scheduler linear \
    --warmup_step 500 \
    --max_epoch 5 \
    --save_interval 1000 \
    --lora_dim 4 \
    --lora_alpha 32 \
    --lora_dropout 0.1 \
    --label_smooth 0.1 \
    --work_dir ./trained_models/GPT2_M/e2e \
    --random_seed 110


# 参数解析:
python -m torch.distributed.launch: 使用PyTorch的分布式启动模块来启动脚本。
--nproc_per_node=1: 每个节点上的进程数为1，即不进行实际的分布式训练，只在单个进程上运行。
--grad_acc: 梯度累积步数
--platform: 指定平台
--clip: 梯度裁剪阈值
--correct_bias: 是否修正偏差
--adam_beta2: Adam优化器的beta2参数
--label_smooth: 标签平滑系数
```

```
  p.data.addcdiv_(-step_size, exp_avg, denom)
| epoch   1 step      100 |    100 batches | lr 4e-05 | ms/batch 443.58 | loss  5.18 | avg loss  5.56 | ppl 259.01
| epoch   1 step      200 |    200 batches | lr 8e-05 | ms/batch 417.75 | loss  3.21 | avg loss  3.75 | ppl 42.48
| epoch   1 step      300 |    300 batches | lr 0.00012 | ms/batch 418.06 | loss  2.97 | avg loss  3.08 | ppl 21.72
| epoch   1 step      400 |    400 batches | lr 0.00016 | ms/batch 418.43 | loss  3.11 | avg loss  2.98 | ppl 19.59
| epoch   1 step      500 |    500 batches | lr 0.0002 | ms/batch 418.79 | loss  2.84 | avg loss  2.89 | ppl 17.98
| epoch   1 step      600 |    600 batches | lr 0.000199 | ms/batch 418.96 | loss  2.77 | avg loss  2.83 | ppl 16.89
| epoch   1 step      700 |    700 batches | lr 0.000198 | ms/batch 418.99 | loss  2.88 | avg loss  2.79 | ppl 16.29
| epoch   1 step      800 |    800 batches | lr 0.000198 | ms/batch 418.96 | loss  2.47 | avg loss  2.75 | ppl 15.72
| epoch   1 step      900 |    900 batches | lr 0.000197 | ms/batch 419.01 | loss  2.50 | avg loss  2.74 | ppl 15.51
| epoch   1 step     1000 |   1000 batches | lr 0.000196 | ms/batch 418.76 | loss  3.18 | avg loss  2.76 | ppl 15.87
saving checkpoint ./trained_models/GPT2_M/e2e/model.1000.pt
| epoch   1 step     1100 |   1100 batches | lr 0.000195 | ms/batch 419.03 | loss  2.84 | avg loss  2.76 | ppl 15.78
| epoch   1 step     1200 |   1200 batches | lr 0.000195 | ms/batch 419.08 | loss  2.58 | avg loss  2.75 | ppl 15.67
| epoch   1 step     1300 |   1300 batches | lr 0.000194 | ms/batch 418.97 | loss  2.62 | avg loss  2.72 | ppl 15.15
| epoch   1 step     1400 |   1400 batches | lr 0.000193 | ms/batch 418.97 | loss  2.70 | avg loss  2.72 | ppl 15.16
| epoch   1 step     1500 |   1500 batches | lr 0.000192 | ms/batch 419.26 | loss  2.66 | avg loss  2.73 | ppl 15.28
| epoch   1 step     1600 |   1600 batches | lr 0.000191 | ms/batch 419.39 | loss  2.69 | avg loss  2.68 | ppl 14.59
| epoch   1 step     1700 |   1700 batches | lr 0.000191 | ms/batch 419.29 | loss  2.57 | avg loss  2.69 | ppl 14.79
| epoch   1 step     1800 |   1800 batches | lr 0.00019 | ms/batch 420.06 | loss  2.54 | avg loss  2.69 | ppl 14.67
| epoch   1 step     1900 |   1900 batches | lr 0.000189 | ms/batch 419.09 | loss  2.69 | avg loss  2.68 | ppl 14.65
| epoch   1 step     2000 |   2000 batches | lr 0.000188 | ms/batch 419.38 | loss  2.51 | avg loss  2.67 | ppl 14.44
saving checkpoint ./trained_models/GPT2_M/e2e/model.2000.pt
```

跑完第1个epoch后的输出模型

第5步：基于上一步训练之后的输出模型进行推理

```
python -m torch.distributed.launch --nproc_per_node=1 src/gpt2_beam.py \
    --data ./data/e2e/test_200.jsonl \
    --batch_size 8 \
    --seq_len 512 \
    --eval_len 64 \
    --model_card gpt2.md \
    --init_checkpoint ./trained_models/GPT2_M/e2e/model.5258.pt \
    --platform local \
    --lora_dim 4 \
    --lora_alpha 32 \
```

```
    --beam 10 \
    --length_penalty 0.8 \
    --no_repeat_ngram_size 4 \
    --repetition_penalty 1.0 \
    --eos_token_id 628 \
    --work_dir ./trained_models/GPT2_M/e2e \
    --output_file predict.5258.s200.jsonl


# 参数解析:
--beam: 束搜索的束宽，指在搜索过程中保留的候选序列数量。更大的束宽通常会导致更好的搜索结果，但也会增加计算成本。

--length_penalty: 长度惩罚系数，避免生成过短或过长的序列。较低的值会惩罚较长的序列，较高的值会鼓励生成更长的序列。
# 示例:
--length_penalty 0.8: 适度惩罚较长的序列，生成结果不会过长。
--length_penalty 1.0: 不进行长度惩罚，生成结果的长度将仅由概率决定。
--length_penalty 1.2: 鼓励生成更长的序列，减少生成短序列的可能性。

--no_repeat_ngram_size: 禁止重复的n-gram大小，用于避免生成结果中出现重复的n-gram。n-gram是指连续的n个词，设置这个参数可以提高生成文本的多样性。
# 示例:
--no_repeat_ngram_size 2: 禁止生成结果中出现重复的二元组（bigram），例如“the cat the cat”。
--no_repeat_ngram_size 3: 禁止生成结果中出现重复的三元组（trigram），例如“the cat is the cat is”。

--repetition_penalty: 重复惩罚系数，用于降低生成过程中重复词或短语的概率。较高的值会强烈惩罚重复，鼓励生成更多样化的文本。
# 示例:
--repetition_penalty 1.0: 不进行重复惩罚。
--repetition_penalty 1.2: 轻微惩罚重复的词或短语，增加生成文本的多样性。
--repetition_penalty 1.5: 强烈惩罚重复的词或短语，显著增加生成文本的多样性。
```

```
================================================================================
Experiment dir : ./trained_models/GPT2_M/e2e
loading model pretrained weight.
model sampling ...
inference samples 0
inference samples 10
inference samples 20
saving prediction file ./trained_models/GPT2_M/e2e/predict.5258.s200.jsonl
cleanup dist ...
```

第6步：基于第5步的输出进行解码