基于Pytorch实现GPT1

1 环境配置

1.1 基础环境

python == 3.8
ftfy == 6.3.1
numpy == 1.24.1
pandas == 2.0.3
scikit_learn == 1.3.2
spacy == 3.4.4
torch == 2.4.1+cu121
tqdm == 4.67.1
CUDA Version == 12.0

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.60.11    Driver Version: 525.60.11    CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:4F:00.0 Off |                  N/A |
| 30%   38C    P2   160W / 350W |  12150MiB / 24576MiB |     77%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA GeForce ...  Off  | 00000000:52:00.0 Off |                  N/A |
| 30%   40C    P2   162W / 350W |   3908MiB / 24576MiB |     63%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA GeForce ...  Off  | 00000000:56:00.0 Off |                  N/A |
| 30%   38C    P2   152W / 350W |   3908MiB / 24576MiB |     64%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  NVIDIA GeForce ...  Off  | 00000000:57:00.0 Off |                  N/A |
| 30%   41C    P2   156W / 350W |   3908MiB / 24576MiB |     71%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   4  NVIDIA GeForce ...  Off  | 00000000:B1:00.0 Off |                  N/A |
| 30%   41C    P2   146W / 350W |   3908MiB / 24576MiB |     44%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   5  NVIDIA GeForce ...  Off  | 00000000:CE:00.0 Off |                  N/A |
| 30%   45C    P2   157W / 350W |   3908MiB / 24576MiB |     79%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   6  NVIDIA GeForce ...  Off  | 00000000:D1:00.0 Off |                  N/A |
| 32%   39C    P2   163W / 350W |   3908MiB / 24576MiB |     72%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   7  NVIDIA GeForce ...  Off  | 00000000:D5:00.0 Off |                  N/A |
| 30%   39C    P2   153W / 350W |   3904MiB / 24576MiB |     81%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   8  NVIDIA GeForce ...  Off  | 00000000:D6:00.0 Off |                  N/A |
| 30%   41C    P2   147W / 350W |   3904MiB / 24576MiB |     52%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

1.2 模型权重文件

下载OpenAI预训练权重并吧model文件夹放入和tran.py同一级文件夹下[finetune-transformer-lm (https://github.com/openai/finetune-transformer-lm)](https://github.com/openai/finetune-transformer-lm)

1.3 ROCStories 完形填空任务数据集

ROCStories 和故事完形填空测试 (https://cs.rochester.edu/nlp/rocstories/)
ROCStories Cloze Test 是一个阅读理解数据集，每篇“故事”由 4 句话组成（上下文），后面有两个候选结尾（ending1 和 ending2），目标是判断哪个结尾更合理

2 模型结构

n_layer = 12, n_head = 12, n_embd = 768（12层，12头，768维）与原论文一致

Model specifications Our model largely follows the original transformer work [62]. We trained a 12-layer decoder-only transformer with masked self-attention heads (768 dimensional states and 12 attention heads). For the position-wise feed-forward networks, we used 3072 dimensional inner states. We used the Adam optimization scheme [27] with a max learning rate of 2.5e-4. The learning rate was increased linearly from zero over the first 2000 updates and annealed to 0 using a cosine schedule. We train for 100 epochs on minibatches of 64 randomly sampled, contiguous sequences of 512 tokens. Since layernorm [2] is used extensively throughout the model, a simple weight initialization of $N(0, 0.02)$ was sufficient. We used a bytepair encoding (BPE) vocabulary with 40,000 merges [53] and residual, embedding, and attention dropouts with a rate of 0.1 for regularization. We also employed a modified version of L2 regularization proposed in [37], with $w = 0.01$ on all non bias or gain weights. For the activation function, we used the Gaussian Error Linear Unit (GELU) [18]. We used learned position embeddings instead of the sinusoidal version proposed in the original work. We use the *ftfy* library[2] to clean the raw text in BooksCorpus, standardize some punctuation and whitespace, and use the *spaCy* tokenizer.[3]

Fine-tuning details Unless specified, we reuse the hyperparameter settings from unsupervised pre-training. We add dropout to the classifier with a rate of 0.1. For most tasks, we use a learning rate of 6.25e-5 and a batchsize of 32. Our model finetunes quickly and 3 epochs of training was sufficient for most cases. We use a linear learning rate decay schedule with warmup over 0.2% of training. λ was set to 0.5.

3 数据处理

datasets.py

```python
def _rocstories(path):
    with open(path, encoding='utf_8') as f:
        f = csv.reader(f)
        st = []   # 存储故事上下文（4句话拼接）
        ct1 = []  # 存储第一个候选结尾
        ct2 = []  # 存储第二个候选结尾
        y = []    # 存储标签（0 或 1）
        for i, line in enumerate(tqdm(list(f), ncols=80, leave=False)):
            if i > 0:  # 跳过表头
                s = ' '.join(line[1:5])  # 拼接4句话作为上下文
                c1 = line[5]             # 第一个候选结尾
                c2 = line[6]             # 第二个候选结尾
                st.append(s)
                ct1.append(c1)
                ct2.append(c2)
                y.append(int(line[-1])-1)  # 标签转换为0和1，原来是1和2
        return st, ct1, ct2, y
```

datasets.py文件下_rocstories返回四个参数分别对应故事上下文，第一个候选结尾，第二个候选结尾，存储标签

rocstories进行验证集训练集测试集的划分，并返回四个元组
(trX1, trX2, trX3, trY)训练集：故事上下文、候选结尾1、候选结尾2、标签
(vaX1, vaX2, vaX3, vaY)验证集：故事上下文、候选结尾1、候选结尾2、标签
(teX1, teX2, teX3)测试集：故事上下文、候选结尾1、候选结尾2（无标签）

4 训练策略和方法

任务目标：ROCStories 任务是给一个故事开头x_1，两个候选结尾x_2、x_3，选择合理的结尾

AI总结：该训练方法基于GPT模型迁移学习，采用语言建模+分类联合损失，配合AdmW优化、warmup调度、梯度裁剪和dropout正则化，通过验证集选择最优模型，最终实现ROCStories 多选任务的准确预测

5 结果

通过一下命令来复现

```
python -m spacy download en

python train.py --dataset rocstories --desc rocstories --submit --
analysis --data_dir ./data/ROCStories/ --n_gpu 8
```

epoch 0 : 74.87% (train) 74.06% (valid)
epoch 1 : 86.90% 83.42%
epoch 2 : 92.51% 87.43%
Best Valid Acc : 87.43%
Test Acc : 84.18%
原论文故事完形填空数据集有86.5%的准确率与这次复现的准确率比较贴近

Table 3: Results on question answering and commonsense reasoning, comparing our model with current state-of-the-art methods.. 9x means an ensemble of 9 models.

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

huggingface的Github开源链接：huggingface/pytorch-openai-transformer-lm: 🐥 A PyTorch implementation of OpenAI's finetuned transformer language model with a script to import the weights pre-trained by OpenAI (https://github.com/huggingface/pytorch-openai-transformer-lm)

五舟配置的开源链接：GPT1 (https://github.com/HelloHiSay/wuzhou-pytorch-transformer-GPT1/tree/main)