MiniGLM实验报告

预训练模型

预训练数据预处理

在prepare.py中，对文本数据进行合并、数据集/验证集划分，tokenize并转换成二进制文件供训练使用。

tokenizer采用tiktoken的gpt2模型。

重点代码如下：

```
# Tokenize raw data with tiktoken encoder
train_tokens = enc.encode_ordinary(train_data)
val_tokens = enc.encode_ordinary(val_data)

# Transform tokenized data to numpy array
train_ids = np.array(train_tokens,dtype=np.uint16)
val_ids = np.array(val_tokens,dtype=np.uint16)

# Save numpy array to files out_dir/train.bin and out_dir/val.bin
output_dir=''
for name in names:
    output_dir = output_dir+name[0]
os.makedirs(output_dir, exist_ok=True)
train_ids.tofile(os.path.join(output_dir, "train.bin"))
val_ids.tofile(os.path.join(output_dir, "val.bin"))
```

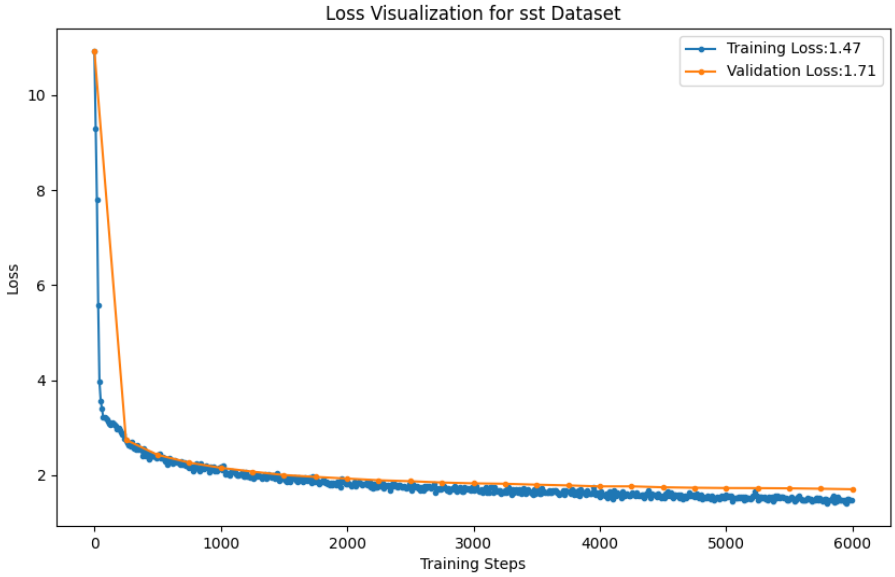
训练流程

没有对提供的train.py进行修改。

训练参数采用learning_rate=1e-3, max_iters=6000。

训练效果

loss函数可视化如下图：



推理效果

推理采用temperature=0.8，max_new_tokens=512。生成速度较快，基本在2s内得到答案。

模型补全效果如下图：

可以看到，预训练模型能够正常对输入进行补全（但补全的不一定是原文）。

微调模型

微调数据构建

utils/gpt.py：使用23条种子问题，将三本书的书名、人物、情节填充进去，形成九百多条问题。将问题传输给chatgpt（五个问题一组）获取答案。

种子问题如下列表：

```
{book}的主人公是谁？
介绍一下{book}的主角。
介绍一下{book}中有名的情节。
{book}的主要角色有哪些？
{book}中提到了哪些武功？
{book}的故事背景是哪个时期？
{book}的主题是什么？
{book}的主要情节是什么？
{book}讲了什么故事？
{book}的结局怎么样？
介绍一下{ch1}。
{ch1}有何绰号？
{ch1}在{book}中的结局怎么样？
{ch1}的伴侣是谁？
{ch1}的朋友有谁？
{ch1}的敌人有谁？
{ch1}的武功是什么？
{ch1}的身世是什么样的？
{ch1}和{ch2}是如何相遇的？
{ch1}和{ch2}有什么故事？
{ch1}和{ch2}之间有什么关系？
讲一下{book}中有名的{plot}情节。
{book}中{plot}情节是什么样的？
```

chatgpt的回答正确率其实无法得到保障，但由于我对金庸小说没有任何了解，人工修改答案实在力不从心，于是单纯沿袭了chatgpt的答案。这会对模型的准确性造成影响，是本次实验的一个不足之处。

微调数据局部展示：

{"Question": "《神雕侠侣》的主要情节是什么？", "Answer": "《神雕侠侣》的主要情节是关于郭靖和黄蓉的爱情故事。故事发生在南宋时期，讲述了郭靖和黄蓉的恋爱经历以及他们与其他角色之间的纠葛。在故事中，郭靖和黄蓉经历了各种艰难险阻，但他们始终坚守真爱，最终战胜了所有的困难，走到了一起。除了爱情故事，故事中还涉及到了武功的修炼、江湖的争斗、阴谋的暗藏等等，使得整个故事更加丰富有趣。"}
{"Question": "阿紫和虚竹之间有什么关系？", "Answer": "阿紫和虚竹之间是一种特殊的师徒关系。虚竹是少林寺的弟子，而阿紫则是神雕派的传人。他们在一次意外中相遇，阿紫被虚竹的善良和正直所吸引，于是决定拜虚竹为师。虚竹则对阿紫的聪明才智和勇敢无畏深感敬佩，两人之间逐渐建立了深厚的师徒情谊。"}
{"Question": "欧阳锋和洪七公之间有什么关系？", "Answer": "欧阳锋和洪七公是武林中的两位高手，他们之间是师兄弟的关系。欧阳锋是洪七公的师兄，他们曾经在同一门派学习武功，共同修炼过。尽管他们在武功上有所不同，但他们之间有着深厚的师徒情谊。"}
{"Question": "杨过和郭靖之间有什么关系？", "Answer": "杨过和郭靖是师兄弟关系。郭靖是杨过的师父，他们之间建立了深厚的师徒情谊。郭靖教导杨过武功，并且在他成长过程中给予了很多指导和帮助。他们共同经历了许多冒险和战斗，相互支持和保护对方。"}
{"Question": "欧阳克和周伯通之间有什么关系？", "Answer": "欧阳克和周伯通是金庸先生的武侠小说《神雕侠侣》中的两个角色。欧阳克是一个心机深重、阴险狡诈的反派角色，而周伯通则是一位慈祥和睿智的老人。两人之间的关系可以说是师徒关系，周伯通曾经是欧阳克的师父，教导他武功和道德。然而，欧阳克背叛了周伯通，并成为了一个邪恶的人物。尽管如此，周伯通仍然对欧阳克抱有一丝希望，并试图教导他回头是岸。"}

微调数据预处理

不同于预训练数据，微调数据在预处理时还需要在将每一对问答数据padding到相同长度（block_size=512），在问题答案之间添加分隔符（'\n'）并将loss_mask标记为只有答案部分为1（即计算loss函数），其余均填入0。

padding的字符选择特殊token<|endoftext|>，同时也能在推理时提供分隔答案功能。

```
text='问：'+data['Question'] +'\n'
    tokens = enc.encode(text,allowed_special={"<|endoftext|>"})
    loss=[0 for x in range(len(tokens)-1)]+[1]# Set loss mask = 0 for

    text='答：'+data['Answer']
    tks=enc.encode(text,allowed_special={"<|endoftext|>"})
    tokens = tokens+tks
    loss=loss+[1 for x in range(len(tks))]# count loss for answers

    if(len(loss)>511):# Truncate the answer
        tokens=tokens[:511]
        loss=loss[:511]

    tokens.append(50256)# Padding (<|endoftext|>)
    loss.append(1)# Count the loss of the first padding
    for j in range(len(tokens),512):# Padding to block size
        tokens.append(50256)
        loss.append(0)
```

训练流程

train.py基本流程没有改变，只是仿效预训练，添加了get_batch_sft的调用

```
# get data loader from data_utils.py
if init_from == 'finetune':
    init_data = init_data_sft
    ### 参照pretrain，使用`functools.partial`进行sft的`get_batch`函数的
    get_batch = partial(get_batch_sft, batch_size=batch_size, block_si
    ###
```

在data_utils.py中，使用init_data_sft()和get_batch_sft()，实现对sft数据的读取、获取批次及构建loss_mask

这两个函数的具体实现与提供的预训练对应函数没有什么太大的差异，只是加入了从预处理过的train_loss.bin/val_loss.bin中读取预处理时已经计算完成的loss_mask的工序。

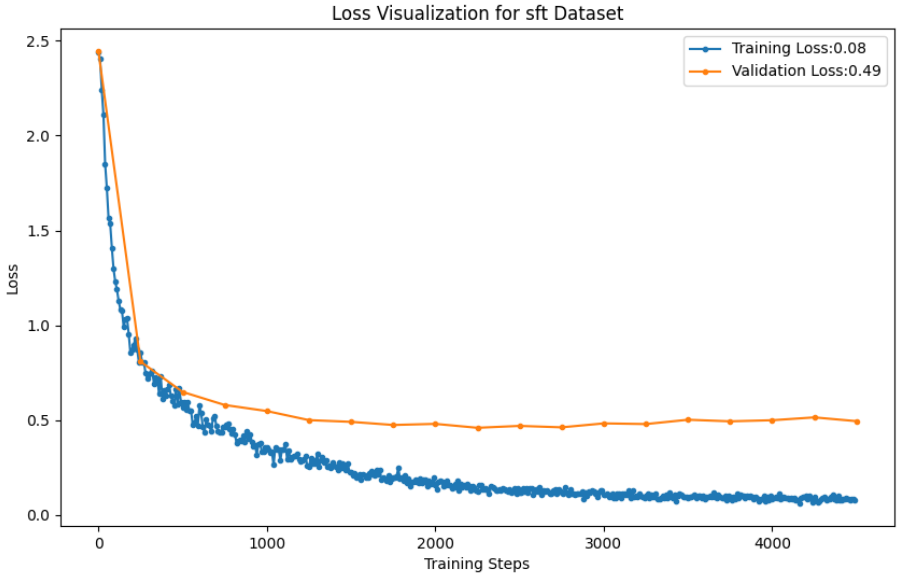
```
def get_batch_sft(split, batch_size, block_size, device):
    ### 获取sft数据的批次（batch）+ 构建损失函数掩码（loss_mask）
    global train_data, val_data,train_loss_mask,val_loss_mask
    data = train_data if split == 'train' else val_data
    mask=train_loss_mask if split=='train' else val_loss_mask

    ix = torch.randint(len(data) - block_size, (batch_size,))
    x = torch.stack([torch.from_numpy((data[i:i+block_size]).astype(np
    y = torch.stack([torch.from_numpy((data[i+1:i+1+block_size]).astyp
    loss_mask = torch.stack([torch.from_numpy((mask[i:i+block_size]).a
    ...
```

训练参数采用learning_rate=1e-4, max_iters=4500。

训练效果

loss曲线如下图：



推理效果

推理采用temperature=0.1，max_new_tokens=512。生成速度较快，基本在2s内得到答案。

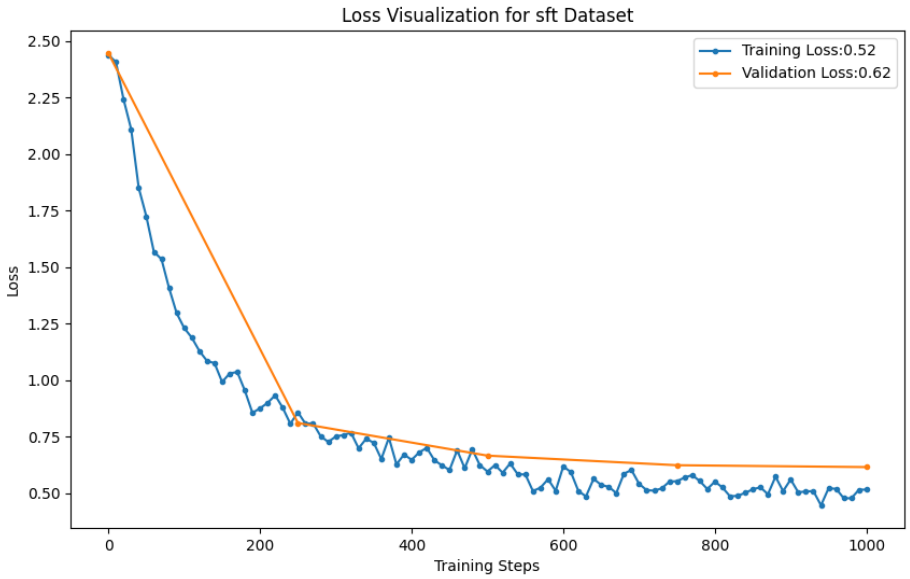
模型问答效果如下：

可以看到，模型拥有基本的问答能力，但回答内容有可能因为sft数据的局限性有一些事实性的错误。

模型效果分析

可以看到微调模型的loss曲线是过拟合的，那么如果仅进行1000次训练，效果会有什么变化呢?

尝试训练1000次，loss曲线如下：



现在对两个模型进行问答测试，采用一个sft数据集中已有的问题和一个没有的问题。

上图为训练4500次的模型，下图为训练1000次的模型。

可以看到，对于sft数据集中已有的问题，过拟合版本回答效果很好，几乎完全吻合给定的答案（train_loss确实也下降到了0.08）；1000次的则会出现所答非所问的情况。对于sft数据集中没有的问题，两个版本都无法给出完美的答案——这是因为数据量较小，且预训练模型本身没有基础问答能力，但过拟合版本在一些问题上可以找到与sft数据集中有的、表述相近的问题来进行作答，1000次版本这个能力会略显逊色。

注：结论并非只靠这两个问题得出，也进行了很多其他问题的测试，只是为展示方便挑选两个代表问题。

总结

在本次实验中，我依托给定的训练程序，完成了以《射雕英雄传》《神雕侠侣》《天龙八部》三部小说为数据集的模型的预训练和微调过程。

预训练模型受制于条件，无法做到完美补全原文，但保证了loss曲线收敛、模型正常补全。

微调数据采用种子问题，借助chatGPT的帮助获取答案。

微调模型在过拟合的情况下，对sft数据集中已有问题作答良好，对数据集外问题有一定推理联想能力。且对比训练次数较少的对照组，微调模型表现更好。