import numpy as np  
  
tang\_file = np.load("tang.npz",allow\_pickle=True)  
tang\_file.files

data = tang\_file['data']  
word2ix = tang\_file['word2ix'].item()  
idx2word = tang\_file['ix2word'].item()

def idx2poem(idx\_poem):  
 poem = []  
 for id in idx\_poem:  
 poem.append(idx2word[id])  
 return "".join(poem)

idx2poem(data[314])

word2ix["</s>"]

poems = []  
for poem in data:  
 for index,ix in enumerate(poem):  
 if ix == word2ix["</s>"]:  
 continue  
 else:  
 break  
 poems.append(poem[index:])

# del data  
poems[0],idx2poem(poems[0]),len(idx2word)

seq\_len = 48  
X = []  
Y = []  
poems\_data = [j for i in poems for j in i]  
  
for i in range(0,len(poems\_data) - seq\_len -1,seq\_len):  
 X.append(poems\_data[i:i+seq\_len])  
 Y.append(poems\_data[i+1:i+seq\_len+1])

from torch.utils.data import DataLoader,Dataset  
import torch  
class PoemDataset(Dataset):  
  
 def \_\_init\_\_(self,X,Y):  
 self.X = X  
 self.Y = Y  
 self.len = len(X)  
 def \_\_getitem\_\_(self,index):  
 x = np.array(X[index])  
 y = np.array(Y[index])  
 return torch.from\_numpy(x).long(),torch.from\_numpy(y).long()  
 def \_\_len\_\_(self):  
 return self.len  
   
data\_loader = DataLoader(PoemDataset(X,Y),batch\_size=512,num\_workers=2)

a,b = next(iter(data\_loader))  
a.shape,b.shape

import torch  
import torch.nn.functional as F  
import torch.nn as nn  
class PoemNet(nn.Module):  
 def \_\_init\_\_(self, vocab\_size, embedding\_dim, hidden\_dim):  
 *"""  
 vocab\_size：训练集合字典大小（8293）  
 embedding\_dim：word2vec的维度  
 hidden\_dim：LSTM的hidden\_dim  
 """* super(PoemNet, self).\_\_init\_\_()  
 #请从这儿开始补充代码

# 定义LSTM模型，模型结构见文档最后部分  
 #补充代码结束

def forward(self, input,hidden=None):  
 *"""  
 input：输入的诗词  
 hidden：在生成诗词的时候需要使用，在pytorch中，，如果不指定初始状态h\_0和C\_0，则其  
 默认为0.  
 pytorch的LSTM的输出是(output,(h\_n,c\_n))。实际上，output就是h\_1,h\_2,……h\_n  
 """*  #请从这儿开始补充代码

# 定义LSTM模型的前向传播过程

#补充代码结束

return output,hidden

vocab\_size = len(word2ix.keys()) # 8293  
embedding\_dim = 200  
hidden\_dim = 1024

device = torch.device("cuda:0" if torch.cuda.is\_available() else "cpu")  
my\_net = PoemNet(vocab\_size,embedding\_dim,hidden\_dim).to(device)

import torch.optim as optim  
optimzer = optim.Adam(my\_net.parameters(),lr=0.001)  
loss\_function = nn.CrossEntropyLoss()

for epoch in range(100):  
 my\_net.train()  
 losses = []  
 for i,data in enumerate(data\_loader):

#请从这儿开始补充代码

# 训练LSTM，计算并输出Loss  
 #补充代码结束

torch.save(my\_net,"model.h5")

注意：如果训练时内核崩溃，说明内存不足，要么增大服务器内存，要么改小DataLoader 中的batch\_size。刚开始训练时epoch可以取小些，待系统正确性得到确认后，再增大epoch。

要求：所构建的LSTM模型结构如下，包括输入层（embedding）、LSTM隐藏层和输出层（由3个全连接层组成）

PoemNet(

(embeddings): Embedding(8293, 200)

(lstm): LSTM(200, 1024, batch\_first=True)

(fc): Sequential(

(0): Linear(in\_features=1024, out\_features=2048, bias=True)

(1): ReLU()

(2): Dropout(p=0.25, inplace=False)

(3): Linear(in\_features=2048, out\_features=4096, bias=True)

(4): Dropout(p=0.2, inplace=False)

(5): ReLU()

(6): Linear(in\_features=4096, out\_features=8293, bias=True)

)

)