import numpy as np

tang\_file = np.load("tang.npz",allow\_pickle=True)

tang\_file.files

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

假定实验3中模型训练已完成，且模型为model.h5

import torch

device = torch.device("cuda:0" if torch.cuda.is\_available() else "cpu")  
my\_net = torch.load("model.h5").to(device)

my\_net.eval()

generate\_poem方法的主要功能：根据给定提示词my\_words进行续写，生成一首完整的诗歌。如：

提示词my\_words：青山隐隐

续写内容：

青山隐隐水迢迢，

秋尽江南草未凋。

孤帆远影碧空尽，

唯见长江天际流。

def generate\_poem(my\_words,max\_len=128):  
 *'''  
 根据前文my\_words生成一首诗。max\_len表示生成诗的最大长度。  
 '''* def \_\_generate\_next(idx,hidden=None):  
 *"""  
 根据input和hidden输出下一个预测词  
 """* input = torch.Tensor([idx]).view(1,1).long().to(device)  
 output,hidden = my\_net(input,hidden)  
 return output,hidden  
  
 # 初始化hidden状态  
 output,hidden = \_\_generate\_next(word2ix["<START>"])  
 my\_words\_len = len(my\_words)  
 result = []  
 for word in my\_words:  
 result.append(word)  
 # 积累hidden状态（h & c）  
 output,hidden = \_\_generate\_next(word2ix[word],hidden)  
   
 \_,top\_index = torch.max(output,1)  
  
 word = idx2word[top\_index[0].item()]  
  
 result.append(word)  
#请从这儿开始补充代码

该部分完成诗歌内容的续写、生成，直至达到最大长度或生成结束标记<EOP>，生成内容存入变量result中

#补充代码结束

return "".join(result)  
  
generate\_poem("重庆")

acrostic\_poetry方法的主要功能：根据给定提示词my\_words生成一首藏头诗。如：

my\_words：春花十里  
藏头诗：  
 春风拂面暖意浓，

花开花落总关情。  
十载岁月如流水，  
里巷深深梦未醒。

def acrostic\_poetry(my\_words):  
 def \_\_generate\_next(idx,hidden=None):  
 *"""  
 根据input和hidden输出下一个预测词  
 """* input = torch.Tensor([idx]).view(1,1).long().to(device)  
 output,hidden = my\_net(input,hidden)  
 return output,hidden  
  
 def \_\_generate(word,hidden):  
 #请从这儿开始补充代码

补充的功能：根据某开始词word生成一句诗（以“，”或“。”结尾） 如根据“床”生成“床前明月光，”

#补充代码结束

\_,hidden = \_\_generate\_next(word2ix["<START>"])  
 result = []  
 for word in my\_words:  
 sentence,hidden = \_\_generate(word,hidden)  
 result.append("".join(sentence))  
 print("\n".join(result))

acrostic\_poetry("大好河山")