1.程序简述

实现一个基于RNN的古诗词生成模型：

–模型构建：继承nn.Module，实现PoetryModel类

–模型训练：基于古诗词数据集，完成PoetryModel的训练

–模型测试：分别实现test和acrostic\_test函数，完成诗歌补完和藏头诗生成

2. class PoetryModel(nn.module)

2.1\_\_ init\_\_函数

2.1.1概述

定义初始化函数，挂载训练所需的模块

（1） 定义模型隐藏层维度, hidden\_dim为读入的隐藏层维度

（2） 使用词嵌入表示(word embedding), embedding\_dim为读入的嵌入向量的维度

（3） 定义LSTM模型（使用2层LSTM）

（4）定义线性模型（从 hidden\_dim 映射到 vocab\_size）

2.1.2 代码说明

super().\_\_init\_\_()

#继承父类的\_\_ init\_\_函数

self.embedding\_dim=embedding\_dim

#定义嵌入层维度

self.hidden\_dim=hidden\_dim

#定义模型隐藏层的维度

self.embeddings=nn.Embedding(vocab\_size,self.embedding\_dim)

#定义词嵌入表示：

#输入的词维度；维度为vocab\_size维，即2.2万余维

#输出的词嵌入向量维度：为self.embedding\_dim维

self.rnn=nn.LSTM(self.embedding\_dim,self.hidden\_dim,2)

#定义LSTM模型；使用两层LSTM

self.output=nn.Linear(self.hidden\_dim,vocab\_size)

#古诗生成case中，每个字即为一类，所以分类数为vocab\_size

#npz古诗数据集中，共有字30余万个，但是不重复字数为2.2万余个：

#2.2万即vocab\_size，即线性层的输出分类数

2.2forward()函数

2.2.1 概述

定义前向传播函数。

输入：input, 它的size是(sequence\_length, batch\_size)

输出（返回值）：output(预测值)，hidden(隐藏层的值)

\* output的size是(sequence\_length, batch\_size, vocab\_size)

\* hidden的size是(4, batch\_size, hidden\_size)

\* h\_0 (2, batch\_size, hidden\_size)

\* c\_0 (2, batch\_size, hidden\_size)

定义模型函数：

\* 判断输入的参数是否有hidden，没有的话新建一个全0的tensor

\* 将input进行词向量嵌入

\* 使用lstm模型

\* 用线性层将output映射到vocab\_size的维度上

\* 返回output, hidden

2.2.2代码说明

seq\_len, batch\_size = input.size()

#input是 一个shape为（序列长度，batch\_size, embedding\_size）的tensor

#seq\_len:序列长度【输入的字的个数】；batch\_size：分批操作，并行运算的诗/诗句数 量

if hidden is None:

h\_0 = input.data.new(2, batch\_size, self.hidden\_dim).fill\_(0).float()

c\_0 = input.data.new(2, batch\_size, self.hidden\_dim).fill\_(0).float()

else:

h\_0, c\_0 = hidden

#LSTM的输出的隐藏值hidden是一个元组：（h\_0, c\_0 ）——分别为隐状态向量和长时 记忆单元向量组成的

input\_embeddings=self.embeddings(input)

#对输入的input进行词向量嵌入

hiddens, hidden=self.rnn(input\_embeddings, (h\_0, c\_0))

#接收上一轮的隐藏值（h\_0，c\_0）

#hiddens是积累的所有的历史隐藏值：所有隐状态向量h组成的tensor；一个shape为 （序列长度, batch\_size, hidden\_size）的tensor

#hidden是上一轮的隐藏值：最后一个隐状态向量；一个shape为（4，batch\_size, hidden\_size）的tensor

output=self.output(hiddens)

#线性层输出分类预测结果

2.2.3 返回值

return output, hidden

#output为一个（seq\_len,batch\_size,vocab\_size）的tensor

#hidden是上一轮的隐藏值：最后一个隐状态向量；一个shape为（2，batch\_size, hidden\_size）的tensor

3. 测试

3.1 start\_words前要不要加“<START>”

首先，不加“<START>”的话，模型也可以正常生成古诗。

但是古诗的质量却会不如人意：第一行的字数会明显的更长，整体的句子字数看起来并不工整。

加上“<START>”作为输入的话，生成的古诗的工整度会好很多。

虽然原因我用语言阐释不清，但大概的意思应该在于，模型训练时，古诗的开头总会有一个“<START>”，标志这首诗歌的开始。如果输入不加入“<START>”的话，会影响模型的预测能力。

3.2 输入的两个报错

input=torch.tensor(top\_index).view(1,1).long().to(self.device)

关于这个输入的处理语句，我有过两次棘手bug，特此记录

3.2.1 view(1,1)

一个是没有加上“.view(1,1)”时，会在class PoetryModel(nn.module)的forward函数中的

seq\_len, batch\_size = input.size()

这一行报错，报错的说明是expected 2 个元素但我给了3个，导致赋值错误。

3.2.2 to(self.device)

另一个是，没有添加“to(self.device)”，导致在GPU上运行时，模型与数据异地。

3.3 acrostic\_test()模型训练不够

当我用在CPU上只用前1000个首诗跑了90个epoch的模型测试，生成的藏头诗总为一串由“</s>”组成的字符串。

后来查明，不是代码的bug，而单纯地是模型训练不够的缘故，模型只学会了“</s>”，道理就像年轻人模仿韩国人说话只会在中文后面加“思密达”一样。

4.GPU训练手记

4.1训练耗时

4.2.1在batch\_size=128，加载前1000首诗的情况下

每7个batch，跳动1个epoch

训练100个epoch，耗时约6分钟

训练1万个epoch，，耗时约10个小时

4.2.2 在batch\_size=128，加载前2万首诗的情况下

每157个batch，跳动1个epoch

训练了10个epoch，耗时共13分钟

4.2.3在batch\_size=128，加载前5万首诗的情况下

每393个batch，跳动1个epoch

训练了200个epoch，耗时13个小时

4.2 训练速度

综上，在GPU平台上，batch\_size=128的条件下，训练速度约为0.5秒/batch

4.3 测试样例

清风徐来花，夕色摇阴上。

吞门得正棱，雪为菰翠午。

芳创自自娱，幽物终有客。

谁知公子来，亦有千里者。

萧萧洒木泉，古木虫鸣宿。

主人樵夫人，涕叟嗟我辱。

顾我守渊市，得予事贱倒。

人间我我师，我亦何况汝。

况复哀子山，其也不敢酒。

哀心在君日，犹是金石路。

但觉雨声游，纵横抱新厅。

芭苔谢秋中，寒食无一斑。

此意若得谢，此意若可叹。

清风徐来

清光何可二，二十十二数。

风如如我去，万古如不足。

徐岂无故宫，岂可顾贪宅。

来子不得其，一笑成绝迹。