# 实验三 自动写诗

## 一、实验目的

1.理解和掌握循环神经网络概念及在深度学习框架中的实现。

2.掌握使用深度学习框架进行文本生成任务的基本流程：如数据读取、构造网络、训练和预测等。

## 二、实验要求

1.基于Python语言和任意一种深度学习框架（实验指导书中使用Pytorch框架进行介绍），完成数据读取、网络设计、网络构建、模型训练和模型测试等过程，最终实现一个可以自动写诗的程序。网络结构设计要有自己的方案，不能与实验指导书完全相同。

2.随意给出首句，如给定“湖光秋月两相和”，输出模型续写的诗句。也可以根据自己的兴趣，进一步实现写藏头诗（不做要求）。要求输出的诗句尽可能地满足汉语语法和表达习惯。实验提供预处理后的唐诗数据集，包含57580首唐诗（在课程网站下载），也可以使用其他唐诗数据集。

3.按规定时间在课程网站提交实验报告、代码以及PPT。

## 三、实验原理

实验使用到的网络结构主要有Embedding、LSTM以及全连接层。在pytorch中分别定义如下：

1. Class torch**.**nn**.**Embedding**(**num\_embeddings**,** embedding\_dim**,** padding\_idx**=None,** max\_norm**=None,** norm\_type**=**2.0**,** scale\_grad\_by\_freq**=False,** sparse**=False,** \_weight**=None)**

参数介绍：

num\_embeddings: 词汇表的大小。

embedding\_dim: 每个嵌入向量的维度。

padding\_idx: 如果提供的话，输出遇到此下标时用零填充。

max\_norm: 如果提供的话，会对词嵌入进行归一化，使它们的范数小于提供的值。

norm\_type: 对于max\_norm选项计算p范数时的p。

scale\_grad\_by\_freq: 如果提供的话，会根据mini-batch中单词频率缩放梯度。

2. Class torch**.**nn**.**LSTM**(**input\_size**,** hidden\_size**,** num\_layers**=**1**,** bias**=True,** batch\_first**=False,** dropout**=**0**,** bidirectional**=False)**

参数介绍：

input\_size: 输入的特征数目。

hidden\_size: 隐状态的特征数目。

num\_layers: LSTM的层数。

bias:是否使用偏置。

batch\_first: 是否将batch放在第一维。如果为True，那么输入输出的Tensor形式都是(batch, seq\_len, num\_features)。

dropout: 如果非零，则除了最后一层，每层的输出都加个Dropout层。

bidirectional: 是否使用双向LSTM。

输入：(注: 以下均考虑batch\_first为False)

intput, 形式为(seq\_len, batch, input\_size)。

h\_0, 形式为(num\_layers \* num\_directions, batch, hidden\_size)。

c\_0, 形式为(num\_layers \* num\_directions, batch, hidden\_size)。

输出：

output，形式为(seq\_len, batch, num\_directions \* hidden\_size)。

h\_n, 形式为(num\_layers \* num\_directions, batch, hidden\_size)。

c\_n, 形式为(num\_layers \* num\_directions, batch, hidden\_size)。

3. Class torch**.**nn**.**Linear**(**in\_features**,** out\_features**,** bias**=True)**

参数：

in\_features: 每个输入样本的大小。

out\_features: 每个输出样本的大小。

bias: 默认值True，若设置为False，这层不会学习偏置。

## 四、实验所用数据集及工具

实验提供预处理过的数据集，含有57580首唐诗，每首诗限定在125词，不足125词的以</s>填充。数据集以npz文件形式保存，包含三个部分：

（1） data: 诗词数据，将诗词中的字转化为其在字典中的序号表示。

（2） ix2word: 序号到字的映射

（3） word2ix: 字到序号的映射

除了此数据集，也可自由选择其他唐诗数据集进行训练，根据需要自行预处理，如：<https://github.com/chinese-poetry/chinese-poetry>。

GPU: Tesla V100-SXM2

## 五、实验步骤与方法

实验可简单地划分为**数据准备**、**模型构建**、**模型训练**和**模型预测**四个步骤。

0. 设置一些参数

为了使诗的长度句子凑整，我们设置生成的句子最大长度8×16=128，而非训练时的125。

# 0.Set some parameters

Batch\_size **=** 16 # batch size

learning\_rate **=** 5e-3 # learn rate

embedding\_dim **=** 128 # embedding layer dimension

hidden\_dim **=** 256 # hidden layer dimension

epochs **=** 4 # epochs to train

verbose **=** **True** # print training process

device **=** torch**.**device**(**'cuda:0' **if** torch**.**cuda**.**is\_available**()** **else** 'cpu'**)** # use GPU first

pre\_trained\_model\_path **=** **None** # pre\_trained model path

trained\_model\_path **=** 'model.pth' # trained model path

start\_words **=** '湖光秋月两相和' # the first sentence of poetry

start\_words\_acrostic **=** '轻舟已过万重山' # the 'head' the the genrated acrostic

max\_gen\_len **=** 128 # the max length of generated poetry

1. 数据准备

# 2.Load data from tang.npz

**def** prepareData**():**

# Load Tang poetry data including 3 parts: data, ix2word, word2ix

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

data **=** datas**[**'data'**]**

ix2word **=** datas**[**'ix2word'**].**item**()**

word2ix **=** datas**[**'word2ix'**].**item**()**

# Translate data from np to torch.Tensor & generate dataloader

data **=** torch**.**from\_numpy**(**data**)**

**print(**data**.**shape**)** # [57580, 125]

dataloader **=** DataLoader**(**data**,**

batch\_size **=** Batch\_size**,**

shuffle **=** **True,**

num\_workers **=** 2**)**

**print(len(**dataloader**))** # 3599

**return** dataloader**,** ix2word**,** word2ix

2. 模型构建

模型构建时，需要创建一个继承自nn.Module的类，在这个类的\_init\_( )中定义网络结构，在forward中定义前向传播过程。

# 3.Define PoetryModel class

**class** **PoetryModel(**nn**.**Module**):**

**def** \_\_init\_\_**(**self**,** num\_embeddings**,** embedding\_dim**,** hidden\_dim**):**

**super(**PoetryModel**,** self**).**\_\_init\_\_**()**

self**.**embedding **=** nn**.**Embedding**(**num\_embeddings**,** embedding\_dim**)**

self**.**hidden\_dim **=** hidden\_dim

self**.**lstm **=** nn**.**LSTM**(**embedding\_dim**,** self**.**hidden\_dim**,** num\_layers**=**2**)**

self**.**linear **=** nn**.**Linear**(**self**.**hidden\_dim**,** num\_embeddings**)**

**def** forward**(**self**,** **input,** hidden **=** **None):**

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

**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

embeds **=** self**.**embedding**(input)**

output**,** hidden **=** self**.**lstm**(**embeds**,** **(**h\_0**,** c\_0**))**

output **=** self**.**linear**(**output**.**view**(**seq\_len **\*** batch\_size**,** **-**1**))**

**return** output**,** hidden

3. 模型训练

模型训练包括定义模型、设置优化器和损失函数、获取模型输出、计算误差、误差反向传播等步骤。

# 3.Define train function

**def** train**(**dataloader**,** ix2word**,** word2ix**):**

# print(len(dataloader)) # 3599, so all len=16\*3598+12\*3599=57580, shape=57580\*125

# config model & load pre-trained model or not

model **=** PoetryModel**(len(**word2ix**),** embedding\_dim**,** hidden\_dim**)**

**if** pre\_trained\_model\_path**:**

model**.**load\_state\_dict**(**torch**.**load**(**model\_path**))**

model**.**to**(**device**)**

# set optimizer & loss

optimizer **=** torch**.**optim**.**Adam**(**model**.**parameters**(),** lr **=** learning\_rate**)**

criterion **=** nn**.**CrossEntropyLoss**()**

# circuit train

**for** epoch **in** **range(**epochs**):**

**for** batch\_idx**,** data **in** **enumerate(**dataloader**):**

data **=** data**.**long**().**transpose**(**1**,** 0**).**contiguous**()**

data **=** data**.**to**(**device**)**

**input,** target **=** data**[:-**1**,** **:],** data**[**1**:,** **:]**

output**,** \_ **=** model**(input)**

loss **=** criterion**(**output**,** target**.**view**(-**1**))**

**if** **(**batch\_idx**+**1**)** **%** 899 **==** 0 **&** verbose**:**

# print(data.shape) # [125,16]

**print(**'Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'**.format(**

epoch**+**1**,** **(**batch\_idx**+**1**)** **\*** Batch\_size**,** **len(**dataloader**.**dataset**),**

100. **\*** **(**batch\_idx**+**1**)** **/** **len(**dataloader**),** loss**.**item**()))**

# if batch\_idx==3598:

# print(data.shape) # [125,12]

optimizer**.**zero\_grad**()**

loss**.**backward**()**

optimizer**.**step**()**

# save model

torch**.**save**(**model**.**state\_dict**(),** 'model.pth'**)**

train(dataloader, ix2word, word2ix)

训练输出如下：

Train Epoch: 1 [14384/57580 (25%)] Loss: 2.048638

Train Epoch: 1 [28768/57580 (50%)] Loss: 2.725324

Train Epoch: 1 [43152/57580 (75%)] Loss: 2.032657

Train Epoch: 1 [57536/57580 (100%)] Loss: 2.049974

Train Epoch: 2 [14384/57580 (25%)] Loss: 1.910754

Train Epoch: 2 [28768/57580 (50%)] Loss: 2.663542

Train Epoch: 2 [43152/57580 (75%)] Loss: 2.669175

Train Epoch: 2 [57536/57580 (100%)] Loss: 1.686005

Train Epoch: 3 [14384/57580 (25%)] Loss: 1.997194

Train Epoch: 3 [28768/57580 (50%)] Loss: 1.950005

Train Epoch: 3 [43152/57580 (75%)] Loss: 1.966352

Train Epoch: 3 [57536/57580 (100%)] Loss: 2.264234

Train Epoch: 4 [14384/57580 (25%)] Loss: 1.941731

Train Epoch: 4 [28768/57580 (50%)] Loss: 2.044527

Train Epoch: 4 [43152/57580 (75%)] Loss: 1.607541

Train Epoch: 4 [57536/57580 (100%)] Loss: 1.713295

4. 自动写诗模型预测

# 4.Define generate poetry function

**def** generate**(**start\_words**,** ix2word**,** word2ix**):**

# load trained\_model from trained\_model\_path

model **=** PoetryModel**(len(**word2ix**),** embedding\_dim**,** hidden\_dim**)**

model**.**load\_state\_dict**(**torch**.**load**(**trained\_model\_path**))**

model**.**to**(**device**)**

# list the start sentence

results **=** **list(**start\_words**)**

start\_word\_len **=** **len(**start\_words**)**

# set the first word as <START>

**input** **=** torch**.**Tensor**([**word2ix**[**'<START>'**]]).**view**(**1**,** 1**).**long**()**

**input** **=** **input.**to**(**device**)**

hidden **=** **None**

# generate poetry in the range of max\_gen\_len

**for** i **in** **range(**max\_gen\_len**):**

output**,** hidden **=** model**(input,** hidden**)**

# print(len(output[0]),len(hidden[0]),len(hidden[1])) # 1\*8293, 2\*2

# load start\_words as the first sentence

**if** i **<** start\_word\_len**:**

w **=** results**[**i**]**

**input** **=** **input.**data**.**new**([**word2ix**[**w**]]).**view**(**1**,** 1**)**

# generate other sentences

**else:**

top\_index **=** output**.**data**[**0**].**topk**(**1**)[**1**][**0**].**item**()**

w **=** ix2word**[**top\_index**]**

results**.**append**(**w**)**

**input** **=** **input.**data**.**new**([**top\_index**]).**view**(**1**,** 1**)**

# end label '<EOP>'

**if** w **==** '<EOP>'**:**

**del** results**[-**1**]**

**break**

**return** results

results **=** generate**(**start\_words**,** ix2word**,** word2ix**)**

**print(**results**)**

**print(len(**results**))**

自动写诗输出如下：

['湖', '光', '秋', '月', '两', '相', '和', '，', '一', '片', '云', '山', '无', '一', '声', '。',

'一', '朝', '不', '见', '青', '山', '曲', '，', '不', '见', '人', '间', '无', '一', '人', '。',

'一', '朝', '不', '见', '青', '山', '曲', '，', '不', '见', '东', '风', '吹', '白', '云', '。',

'一', '朝', '不', '见', '青', '山', '曲', '，', '不', '见', '东', '风', '吹', '白', '云', '。',

'一', '朝', '不', '见', '青', '山', '曲', '，', '不', '见', '东', '风', '吹', '白', '云', '。',

'一', '朝', '不', '见', '青', '山', '曲', '，', '不', '见', '青', '山', '不', '可', '见', '。',

'一', '朝', '不', '见', '青', '山', '人', '，', '不', '见', '春', '风', '吹', '白', '雪', '。',

'一', '朝', '不', '见', '春', '风', '起', '，', '一', '曲', '花', '前', '花', '下', '来', '。']

128

5. 藏头诗模型预测

# 5.Define generate acrostic function

**def** gen\_acrostic**(**start\_words**,** ix2word**,** word2ix**):**

# load trained\_model from trained\_model\_path

model **=** PoetryModel**(len(**word2ix**),** embedding\_dim**,** hidden\_dim**)**

model**.**load\_state\_dict**(**torch**.**load**(**trained\_model\_path**))**

model**.**to**(**device**)**

# load the 'head' of the acrostic

results **=** **[]**

start\_word\_len **=** **len(**start\_words**)**

# set the first word as <START>

**input** **=** **(**torch**.**Tensor**([**word2ix**[**'<START>'**]]).**view**(**1**,** 1**).**long**())**

**input** **=** **input.**to**(**device**)**

hidden **=** **None**

index **=** 0 # index of the character in start\_words

pre\_word **=** '<START>' # pre\_word

# generate acrostic in the range of max\_gen\_len

**for** i **in** **range(**max\_gen\_len**):**

output**,** hidden **=** model**(input,** hidden**)**

top\_index **=** output**.**data**[**0**].**topk**(**1**)[**1**][**0**].**item**()**

w **=** ix2word**[**top\_index**]**

# if the pre\_word is end or start label, set the next character in start\_words as the next word

**if** **(**pre\_word **in** **{**u'。'**,** u'！'**,** '<START>'**}):**

# condition of end

**if** index **==** start\_word\_len**:**

**break**

# feed the next character as head

**else:**

w **=** start\_words**[**index**]**

index **+=** 1

**input** **=** **(input.**data**.**new**([**word2ix**[**w**]])).**view**(**1**,** 1**)**

# otherwise, set the next prediction as the next word

**else:**

**input** **=** **(input.**data**.**new**([**word2ix**[**w**]])).**view**(**1**,** 1**)**

results**.**append**(**w**)**

pre\_word **=** w

**return** results

results\_acrostic **=** gen\_acrostic**(**start\_words\_acrostic**,** ix2word**,** word2ix**)**

**print(**results\_acrostic**)**

写藏头诗输出如下：

['轻', '生', '不', '得', '意', '，', '不', '得', '不', '得', '知', '。',

'舟', '中', '有', '奇', '气', '，', '不', '得', '不', '得', '持', '。',

'已', '闻', '天', '上', '来', '，', '不', '得', '不', '得', '宁', '。',

'过', '此', '不', '可', '见', '，', '不', '知', '何', '处', '期', '。',

'万', '里', '不', '可', '见', '，', '一', '朝', '无', '人', '知', '。',

'重', '阳', '不', '可', '见', '，', '一', '日', '不', '可', '攀', '。',

'山', '川', '有', '高', '树', '，', '山', '水', '无', '人', '舟', '。']

从实验中可以看到，训练损失逐渐变小，但是写出来的诗句重复较多，而且没有太强的韵律特点，甚至没有含义。还需要提取更加丰富的特征来训练网络。