古诗生成器（创意版）

基于循环神经网络（Keras+LSTM-RNN），采用了浙江大学人工智能研究所提供的古诗词库，并且在其提供的AI学习平台上训练完成。本案例非原创，原来的代码用class来实现，并且重构了model的方法。为了方便初学者理解，重新调整了代码和参数，并且用jupyterlab写了一个完整的文档。

虽然模型已经训练完成，但要应用这个模型，还需要提供原来用于训练的语料，即古诗词库。训练和应用使用的语料要保持一致。

运行这个案例，需要安装多个库，还需要pydot、graphviz的支持。虚谷号教育版上已经提供了所有相关的支持库。需要说明的是，在虚谷号上训练比较慢，预计至少需要一天时间，才可以看到较好的效果。

除了常规的藏头诗、随机诗外，还可以做哪些好玩的事情？创意版试图做些不一样的工作，供抛砖引玉。

原案例地址：https://github.com/youyuge34/Poems_generator_Keras/blob/master/poem_model.ipynb (https://github.com/youyuge34/Poems_generator_Keras/blob/master/poem_model.ipynb)

模型下载地址（课程汇集/虚谷号内置课程目录/5.机器学习）：https://github.com/vvlink/vvBoard-docs/ (https://github.com/vvlink/vvBoard-docs/)

1.导入必要的库

In [1]:

```
import random
import os
import keras
import numpy as np
from keras.callbacks import LambdaCallback
from keras.models import Input, Model, load_model
from keras.layers import LSTM, Dropout, Dense
from keras.optimizers import Adam
```

```
Using TensorFlow backend.
```

2.参数设置

In [2]:

```
class config(object):
    # 输入的诗词库（语料库）
    poetry_file = 'dataset/8-poetry_zju.txt'
    # 模型名称
    weight_file = 'model/8-model_zju.h5'
    # 输出训练的信息
    fixlog = 'poem_log.txt'
    # 复合训练时，间隔多少次输出一次测试结果
    predict_num = 5
    batch_size = 32
    learning_rate = 0.001
    # 下面参数不能随意修改，改动将影响整个模型的大小
    #根据前六个字预测第七个字，生成的是五言诗（含标点）
    max_len = 6
    # 去除低频率文字（避免生僻字）
    frequence_num = 3
```

3.数据处理

In [3]:

```
def preprocess_file(poetry_file):
    # 语料文本内容
    files_content = ''
    with open(poetry_file, 'r',encoding='UTF-8') as f:
        for line in f:
            x = line.strip() + "]"
            # x = x.split(":")[1]
            if len(x) <= 5 :
                continue
            # 确保导入的诗句，是五言诗
            if x[c.max_len-1] == ', ':
                files_content += x

    words = sorted(list(files_content))
    counted_words = {}
    for word in words:
        if word in counted_words:
            counted_words[word] += 1
        else:
            counted_words[word] = 1

    # 去掉低频的字，这样可以去除一些怪异的字。
    erase = []
    for key in counted_words:
        if counted_words[key] <= c.frequence_num:
            erase.append(key)
    for key in erase:
        del counted_words[key]
    wordPairs = sorted(counted_words.items(), key=lambda x: -x[1])

    words, _ = zip(*wordPairs)
    words += (" ",)
    # word到id的映射
    word2num = dict((c, i) for i, c in enumerate(words))
    num2word = dict((i, c) for i, c in enumerate(words))
    word2numF = lambda x: word2num.get(x, len(words) - 1)
    return word2numF, num2word, words, files_content

# 清洗、准备数据
c=config()
word2numF, num2word, words, files_content = preprocess_file(c.poetry_file)
#分割诗词，记录在poems_num中
poems = files_content.split(']')
poems_num = len(poems)
```

这里返回的四个变量，分别介绍如下：

- word2numF：返回不同汉字对应的字典位置。
- num2word：字典，key是数字，值是字符。
- words：列表。所有的字符表，按照频率排序，先大再小。
- files_content：字符串。所有的诗词，用“]”分开。

按照字符频率，最多的是“，”，其次是“。”。poems为列表，存储所有的诗歌，poems_num为诗歌的数量。

4.应用模型

根据模型能够根据输入的一组字来预测后面的字，那么就可以做出很多应用，如藏头诗，藏字诗等。

4.1 导入模型设置参数

In [45]:

```
# 导入训练好的模型
model = load_model(c.weight_file)

# 热度参数，默认设置为1，正常的参数
temperature=1.0

# 输入字符串，返回字符串
def m_preds(sentence,length = 23,temperature =1):
    '''
    sentence:预测输入值
    lenth:预测出的字符串长度
    供类内部调用，输入max_len长度字符串，返回length长度的预测值字符串
    '''
    sentence = sentence[:c.max_len]
    generate = ''
    for i in range(length):
        pred = m_pred(sentence,temperature)
        generate += pred
        sentence = sentence[1:]+pred
    return generate

# 输入字符串，返回字符
def m_pred(sentence,temperature =1):
    '''内部使用方法，根据一串输入，返回单个预测字符'''
    if len(sentence) < c.max_len:
        print('in def m_pred,length error ')
        return

    sentence = sentence[-c.max_len:]
    x_pred = np.zeros((1, c.max_len, len(words)))
    for t, char in enumerate(sentence):
        x_pred[0, t, word2numF(char)] = 1.
    preds = model.predict(x_pred, verbose=0)[0]
    next_index = sample(preds,temperature=temperature)
    next_char = num2word[next_index]

    return next_char

# 根据输入的矩阵，返回一个数字
def sample(preds, temperature=1.0):
    '''
    当temperature=1.0时，模型输出正常
    当temperature=0.5时，模型输出比较open
    当temperature=1.5时，模型输出比较保守
    在训练的过程中可以看到temperature不同，结果也不同
    就是一个概率分布变换的问题，保守的时候概率大的值变得更大，选择的可能性也更大
    '''
    preds = np.asarray(preds).astype('float64')
    exp_preds = np.power(preds,1./temperature)
    preds = exp_preds / np.sum(exp_preds)
    pro = np.random.choice(range(len(preds)),1,p=preds)
    return int(pro.squeeze())
```

```
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x,
dropout() uses dropout rate instead of keep_prob. Please ensure that t
his is intended.
WARNING:tensorflow:Large dropout rate: 0.6 (>0.5). In TensorFlow 2.x,
dropout() uses dropout rate instead of keep_prob. Please ensure that t
his is intended.
```

热度参数 temperature 说明

一般来说，temperature=1.0输出的比较正常。小于1比较开放，大于1则比较保守。就是一个概率分布变换的问题，保守的时候概率大的值变得更大，选择的可能性也更大。

4.2 藏头诗升级版

思路分析：

总觉得普通的藏头诗效果不太好，于是设计了一种思路：先在诗词库中找到以“藏头字”开头的词语，优先嵌入诗句，让生成的诗句变得有意义。

实现方式：

首先需要分词，对诗词库中的所有诗句进行分词。不选择原始语料，毕竟诗词库中已经整理过，是五言诗句。

这需要安装jieba库。这是一个常用的中文分词库，还支持词云之类的功能。jieba分词原理比较简单，根据固有的词库进行关联度分析，出现频率大的词语进行有效切分，因而产生的词语符合我们大众的组词习惯。

安装命令（使用清华镜像）： pip install -i https://pypi.tuna.tsinghua.edu.cn/simple (https://pypi.tuna.tsinghua.edu.cn/simple) jieba

In [7]:

```
import jieba
txt=files_content  # 导入诗词库
word_object = jieba.lcut(txt,cut_all=False)  # 生成一个列表
```

```
Building prefix dict from the default dictionary ...
Loading model from cache /var/folders/01/hcypgmm52h7fbcggc7sfr1140000g
n/T/jieba.cache
Loading model cost 0.921 seconds.
Prefix dict has been built successfully.
```

In [9]:

```
# 输出分词的个数
print(len(word_object))
```

```
1183798
```

In [10]:

```
#删除重复的元素（分词结果），这个工作需要一定的时间
new_word_object=list(set(word_object))
#合并后的词语个数
print(len(new_word_object))
# 排序
new_word_object=sorted(new_word_object)
# 输出前面几个对象
print(new_word_object[:100])
```

In [32]:

```
# 获得某个字开头的所有词语
def find_str(s,li):
    t=[]
    for i in li:
        if i[0]==s:
            t.append(i)
    #return t
    index = random.randint(0, len(t)-1)
    return str(t[index])

# '''根据给4个字，生成藏头诗五言绝句'''
def predict_hide(text,temperature = 1):
    if len(text)!=4:
        print('藏头诗的输入必须是4个字！')
        return
    islow=''
    for t in text:
        if t not in words:
            islow = t
            break
    if islow:
        print('输入的字存在冷僻字：'+ islow)
        return

    # 在分词库中，随机找出一个词语
    wlist=find_str(text[0],new_word_object)
    #选取随机一首诗的最后max_len字符 + 找到词语作为初始输入
    index = random.randint(0, poems_num)
    sentence = poems[index][len(wlist)-c.max_len:] + wlist
    generate = str(wlist)
    # print('引子 = ',sentence)

    for i in range(5-len(wlist)+1):
        next_char = m_pred(sentence,temperature)
        sentence = sentence[1:] + next_char
        generate+= next_char

    for i in range(3):
        # 在分词库中，随机找出一个词语
        wlist=find_str(text[i+1],new_word_object)
        generate += wlist
        sentence = sentence[len(wlist):] + wlist
        for i in range(5-len(wlist) + 1):
            next_char = m_pred(sentence,temperature)
            sentence = sentence[1:] + next_char
            generate+= next_char

    return generate
```

In [85]:

```
# 执行
for i in range(3):
    #藏头诗
    sen = predict_hide('一杯浊酒',temperature = temperature)
    print(sen)
```

一水遥五寒，杯明自清人。浊气人尽露，酒酣空影连。
一言知请香，杯锡上含伤。浊波中苍落，酒用月道无。
一共林去临，杯重长九高。浊者月营风，酒乡露化长。

In [ ]:

```

```