文本生成模型-参数优化介绍

1、 maxlen:这个是滑动框的大小，新生成的代码由前面 maxlen 个代码词序列预测而来.

```
"""数据预处理-字符串序列向量化"""

import numpy as np
import keras

maxlen = 50       #提取50个代码词组成的序列
step = 3        #每10个代码词采样一个新序列
sentences = []        #保存所提取的序列
next_words = []        #保存目标代码词

cut_words = list_text        #将列表形式的元数据保存在cut_words中
for i in range(0,len(cut_words) - maxlen,step):
    sentences.append(cut_words[i:i + maxlen])        #将元数据按照步长来存储在每个序列中
    next_words.append(cut_words[i + maxlen])        #将目标代码词存储在next_words中


print('Number of sequences:', len(sentences))
```

2、 step:步长。假设步长为 3，那么每隔 3 个代码词，就生成一个 sentence，作为 x 的一个输入。

```
"""数据预处理-字符串序列向量化"""

import numpy as np
import keras

maxlen = 50        #提取50个代码词组成的序列
step = 3        #每10个代码词采样一个新序列
sentences = []        #保存所提取的序列
next_words = []        #保存目标代码词

cut_words = list_text        #将列表形式的元数据保存在cut_words中
for i in range(0,len(cut_words) - maxlen,step):
    sentences.append(cut_words[i:i + maxlen])        #将元数据按照步长来存储在每个序列中
    next_words.append(cut_words[i + maxlen])        #将目标代码词存储在next_words中


print('Number of sequences:', len(sentences))
```

3、模型能修改的参数就比较多了，比如：

1）Embedding 层的参数个数；

2）LSTM 层的参数个数、dropout 的值；

3）甚至模型的层的叠加可以尝试更多不同的方案，现在只用了一个最简单的 LSTM 模型，肯定是远远不够的；

```
"""模型尝试一：yk_model_local_gpu-0507-01：Embedding + 单层LSTM(加入dropout)"""

import keras
from keras import layers
from keras.layers import LSTM, Dense, Dropout

def create_model(words):        #定义创建模型的函数
    model = keras.models.Sequential()        #模型初始化
    model.add(layers.Embedding(len(words),128))        #模型第一层为embedding层
    model.add(layers.LSTM(128,dropout=0.2,recurrent_dropout=0.2))        #模型第二层为LSTM层，加入dropout减少过拟合
    model.add(layers.Dense(len(words),activation='softmax'))        #模型第三层为全连接层

    optimizer = keras.optimizers.RMSprop(lr=0.003)        #定义优化器
    model.compile(loss='categorical_crossentropy',optimizer=optimizer)        #模型编译

    return model
```

4、关于训练模型部分：

1）训练的 epoch 的轮数随你修改；

2）batch_size 的值也可以修改，但是 batch_size 的值一般为：128,256,512,1024...这几个值都可以试试；

3）temperature 的值我尝试了 0.1、0.4、0.8 这三个，你们也可以尝试其他的；

```
"""训练模型"""

#这一部分的代码我改的比较多，看的时候要稍微耐心点，有看不懂的地方可以随时问我

import random
import sys
import os

strings =''        #将生成的代码保存下来，一轮epoch结束后，将生成的代码写入到文件中

#mark、last_word、start_gen定义的目的是为了让最终的生成的代码符合标准代码的格式要求----简言之就是该空格的地方空格，不该的就不空格
mark = '.,()[]:{}\n'        #将后面不需要空格的元素保存在字符串中
last_word = ''
start_gen=''

for epoch in range(0,50):
    print('\n' + '----------epoch=' + str(epoch) + '----------')
    strings += '\n' + '----------epoch=' + str(epoch) + '----------'

    if os.path.exists(filepath):        #如果模型存在，则从现有模型开始训练
        model.load_weights(filepath)
        print("==========正在从断点开始续训模型==========")
        strings += "==========正在从断点开始续训模型=========="

        model.fit(x,y,batch_size=128,epochs=1,callbacks=[checkpoint])        #开始训练模型

    else:
        model.fit(x,y,batch_size=128,epochs=1,callbacks=[checkpoint])


    for temperature in [0.1,0.4,0.8]:        #定义随时数，随机数越高，文本生成的创造性越强，规则表示越弱
        print('\n' + '------ temperature:' ,str(temperature) +'\n' )
        strings += '\n' + '------ temperature:' + str(temperature) +'\n'
```

5、这两句话很重要：如果下面这两句话不标注掉，那么本模型是按照滑动框（n-gram）来训练的。标注掉以后，每一个新的代码词都是从已生成的代码序列来进行预测的。
所以我建议是，大家把标注掉或者没标注掉的代码都跑一边，稍微比较看看，哪种效果更好。

```
for i in range(50):      #生成一个代码句
    sampled = np.zeros((1,len(generated_text)))      #根据现有代码词长度，初始化一个相同长度的sampled
    for t,word in enumerate(generated_text):      #将已有代码词向量化
        sampled[0,t] = word_indices[word]

    preds = model.predict(sampled,verbose=0)[0]      #预测并生成下一个代码词
    next_index = sample(preds,temperature = 0.3)
    next_word = words[next_index]


    generated_text.append(next_word)      #将生成的代码词加到已生成的代码序列中

    #如果下面这两句话不标注掉，那么本模型是按照滑动框（n-gram）来训练的。标注掉以后，每一个新的代码词都是从
    #if len(generated_text) == maxlen:
    #    generated_text = generated_text[1:]

    if next_word not in mark and last_word not in mark:      #将生成的代码转换成标准代码格式，并打印出来
        sys.stdout.write(' ' + next_word)
        strings += ' ' + next_word
    else:
        sys.stdout.write(next_word)
        strings +=  next_word

    last_word = next_word

    if next_word == '\n':      #如果预测的代码词为\n，那么表示这一句结束
        break
```

总结：参数的优化主要是上面提到的几个，个人感觉比较重要的：

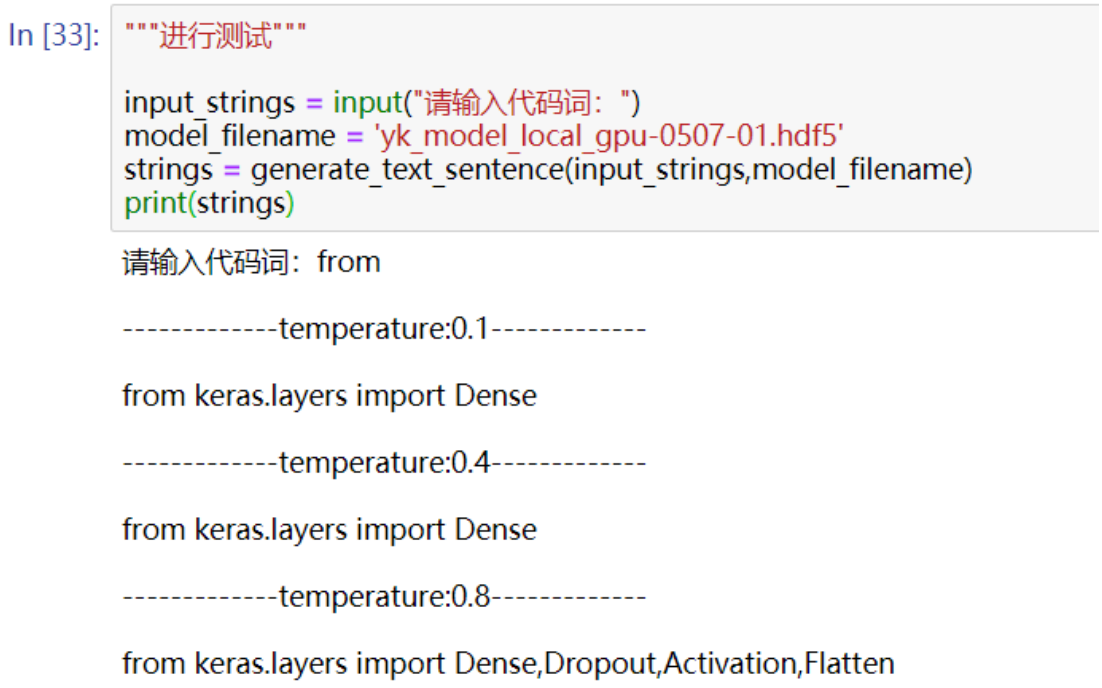
1）模型的构建；
2）step；
3）是否使用滑动框；

第二部分：关于测试代码的使用介绍

1、 代码句的推荐：

1）运行代码句推荐函数

2）进行测试：运行该代码时，会弹出一个提示框，输入 from 之后，会根据 temperate 的值不同，生成不同的结果。



2、代码段的推荐：
代码段的推荐操作同理，输入'def model():'，可以看到 temperature=0.8 的效果意外的好。
非常有意思，推荐大家好好训练下模型，看看最终测试效果如何。

```
In [*]: """进行测试"""

        input_strings = input("请输入代码词：")
        model_filename = 'yk_model_local_gpu-0507-01.hdf5'
        strings = generate_text_paragraph(input_strings,model_filename)
        print(strings)

        请输入代码词：  def model():
```

```
"""进行测试"""

input_strings = input("请输入代码词：")
model_filename = 'yk_model_local_gpu-0507-01.hdf5'
strings = generate_text_paragraph(input_strings,model_filename)
print(strings)
```

```
请输入代码词：def model():
d:\01-software-installation\06-python-3.5.4\lib\site-packages\ipykernel_launcher.p
  """


----------temperature:0.1----------

def model():
model = Sequential()
model.add(LSTM(input_shape =(),return_sequences =))
model.add(Dropout())
model.add(LSTM())
model.add(Dropout())
model.add(Dense())
model.add(Activation())
model.compile(loss =,optimizer =,metrics =[])
if show_summaries:


 ----------temperature:0.4----------

 def model():
 model = Sequential()
 model.add(LSTM(input_dim =,output_dim =,return_sequences =))
 model.add(Dropout())
 model.add(LSTM())
 model.add(Dense())
 model.add(Activation())
 model.compile(loss =,optimizer =,metrics =[])
 model.fit(x_train,y_train,batch_size =,epochs =,batch_size =)
 return model

 ----------temperature:0.8----------

 def model():
 model = Sequential()
 model.add(LSTM(input_shape =(),return_sequences =))
 model.add(LSTM())
 model.add(Dropout())
 model.add(Dense())
 model.add(Activation())
 model.compile(loss =,optimizer =,metrics =[])
 model.fit(x_train,y_train,batch_size =,epochs =)
 return model
```