# Keras-Bert 实现文本分类

目录

[1. 安装Keras、keras-bert、tensorflow](#_Toc28085_WPSOffice_Level1) [1](#_Toc28085_WPSOffice_Level1)

[2. 下载bert模型](#_Toc31575_WPSOffice_Level1) [1](#_Toc31575_WPSOffice_Level1)

[3. 分类](#_Toc27842_WPSOffice_Level1) [1](#_Toc27842_WPSOffice_Level1)

[一、 项目实现](#_Toc154_WPSOffice_Level1) [2](#_Toc154_WPSOffice_Level1)

[1.1. 数据集](#_Toc31575_WPSOffice_Level2) [2](#_Toc31575_WPSOffice_Level2)

[1.2. 流程](#_Toc27842_WPSOffice_Level2) [2](#_Toc27842_WPSOffice_Level2)

[1.2.1. 导入的包和参数](#_Toc31575_WPSOffice_Level3) [2](#_Toc31575_WPSOffice_Level3)

[1.2.2. 得到编码](#_Toc27842_WPSOffice_Level3) [3](#_Toc27842_WPSOffice_Level3)

[1.2.3. 健壮性编码](#_Toc154_WPSOffice_Level3) [3](#_Toc154_WPSOffice_Level3)

[1.2.4. 读取数据](#_Toc9787_WPSOffice_Level3) [4](#_Toc9787_WPSOffice_Level3)

[1.2.5. 将文字进行编码](#_Toc1420_WPSOffice_Level3) [4](#_Toc1420_WPSOffice_Level3)

[1.2.6. 加载Bert模型和构建自己的模型](#_Toc23532_WPSOffice_Level3) [5](#_Toc23532_WPSOffice_Level3)

[1.2.7. 运行](#_Toc3333_WPSOffice_Level3) [6](#_Toc3333_WPSOffice_Level3)

[1.2.8. 结果](#_Toc27007_WPSOffice_Level3) [6](#_Toc27007_WPSOffice_Level3)

[1.2.9. 总结](#_Toc8044_WPSOffice_Level3) [7](#_Toc8044_WPSOffice_Level3)

[参考：](#_Toc9787_WPSOffice_Level1) [7](#_Toc9787_WPSOffice_Level1)

前言：Bert讲解网上很多，就不累赘。使用Keras-bert实现分类很少，故尝试使用。使用过程如下

1. 安装Keras、keras-bert、tensorflow

pip install XXX -i https://pypi.douban.com/simple/

1. 下载bert模型

<https://github.com/google-research/bert/> 该网站下 下载bert模型，可能需要梯子，本项目上传到了[github](https://github.com/Luomiou/post)上，供大家使用。改模型的下载地址如下：

<https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip> 。

将压缩包解压，有如下四个文件：



1. 分类

废话少说，讲点效率的事。将以上模型应用到二分类上。手撕代码。

## 项目实现

项目所需要的 数据集 和 模型 下载地址为：

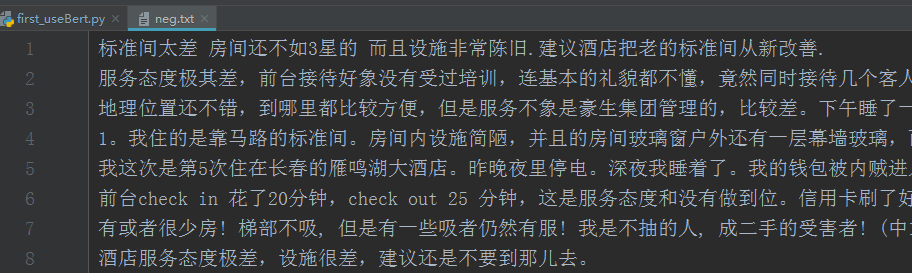
数据集：<https://pan.baidu.com/s/1AvKeyY8_QC_ksXozGxZPlg> 提取码：vk8f

模型chinese\_L-12\_H-768\_A-12：

<https://pan.baidu.com/s/1QFmUqh0ArsrlNCWFt8qGYA> 提取码：g6ro

### 1.1. 数据集

首先看看数据集长什么样。本数据是 谭松波 酒店评论 数据，6000=3000 pos+3000 neg)条,谭的数据集全是 一句话一个txt文本，本人将这些总结到了一个文档数据，在github上



### 1.2. 流程

#### 1.2.1. 导入的包和参数

import codecs  
import os  
import sys  
  
import numpy as np  
from keras import Input, Model, losses  
from keras.layers import Lambda, Dense  
from keras.optimizers import Adam  
from keras.preprocessing import sequence  
from keras\_bert import Tokenizer, load\_trained\_model\_from\_checkpoint  
  
syspath = sys.path[1]+'/hotl'  
os.chdir(syspath)  
# 注意！！！！特别注意！！！此处要慎重，是绝对路径，以下内容是Bert-Pre-training  
config\_path = 'D:\Pycharm\post\hotl\chinese\_L-12\_H-768\_A-12\\bert\_config.json'# 加载配置文件  
checkpoint\_path = 'D:\Pycharm\post\hotl\chinese\_L-12\_H-768\_A-12\\bert\_model.ckpt'  
dict\_path = 'D:\Pycharm\post\hotl\chinese\_L-12\_H-768\_A-12\\vocab.txt'  
# 以上是模型的加载，要慎重，一定要慎重！！！！！  
  
maxlen=100# 句子的最大长度，padding要用的

#### 1.2.2. 得到编码

在喂入bert模型之前，将文本数据进行编码，也就是对每个汉字进行编码，可不是词语哦！！！，Bert模型中有个vocab.txt文件，类似一个大词典。使用get\_token\_dict(vocab.txt)将得到每个字对应的索引。

def get\_token\_dict(dict\_path):  
 *'''* ***:param****: dict\_path: 是bert模型的vocab.txt文件* ***:return****:将文件中字进行编码  
 '''* # 将bert模型中的 字 进行编码  
 # 目的是 喂入模型 的是 这些编码，不是汉字  
 token\_dict = {}  
 with codecs.open(dict\_path, 'r', 'utf-8') as reader:  
 for line in reader:  
 token = line.strip()  
 token\_dict[token] = len(token\_dict)  
 return token\_dict

测试：

token\_dict = get\_token\_dict(dict\_path)  
print(token\_dict)

结果：

{'': 13503, '##right': 11264,................, '慷': 2724, '##盅': 17714, '##剤': 14251}

#### 1.2.3. 健壮性编码

class OurTokenizer(Tokenizer):  
 *'''  
 关键在 Tokenizer 这个类，要实现这个类中的方法，其实不实现也是可以的  
 目的是 扩充 vocab.txt文件的  
 '''* def \_tokenize(self, text):  
 R = []  
 for c in text:  
 if c in self.\_token\_dict:  
 R.append(c)  
 elif self.\_is\_space(c):  
 R.append('[unused1]')  
 else:  
 R.append('[UNK]')  
 return R

#### 1.2.4. 读取数据

def get\_data():  
 *'''  
 读取数据的函数* ***:return****: list 类型的 数据  
 '''* pos = []  
 neg = []  
 with codecs.open('./data/pos.txt','r','utf-8') as reader:  
 for line in reader:  
 pos.append(line.strip())  
 with codecs.open('./data/neg.txt','r','utf-8') as reader:  
 for line in reader:  
 neg.append(line.strip())  
 return pos,neg

#### 1.2.5. 将文字进行编码

# 得到编码  
def get\_encode(pos,neg,token\_dict):  
 *'''* ***:param*** *pos:第一类文本数据* ***:param*** *neg:第二类文本数据* ***:param*** *token\_dict:编码字典* ***:return****:[X1,X2]，其中X1是经过编码后的集合，X2表示第一句和第二句的位置，记录的是位置信息  
 '''* all\_data = pos + neg  
 tokenizer = OurTokenizer(token\_dict)  
 X1 = []  
 X2 = []  
 for line in all\_data:  
 # tokenizer.encode(first,second, maxlen)  
 # 第一句和第二句，最大的长度，  
 # 本数据集是 都是按照第一句，即一行数据即是一句，也就是第一句  
 # 返回的x1,是经过编码过后得到，纯整数集合  
 # 返回的x2,源码中segment\_ids，表示区分第一句和第二句的位置。

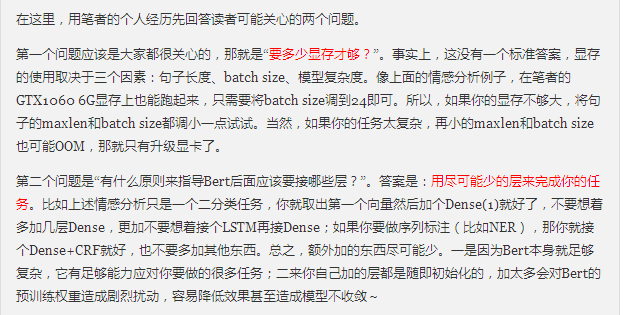
# 结果为：[0]\*first\_len+[1]\*sencond\_len  
 # 本数据集中，全是以字来分割的。  
 # line\_list = line.split('.')  
 # for i in line\_list:  
 # print(i)  
 x1,x2 = tokenizer.encode(first=line)  
 # print(line,'\n')  
 # print(x1,'\n',len(x1),'\n',x2,'\n',len(x2))  
 # break  
 X1.append(x1)  
 X2.append(x2)  
 # 利用Keras API进行对数据集 补齐 操作。  
 # 与word2vec没什么区别，都需要进行补齐  
 X1 = sequence.pad\_sequences(X1,maxlen=maxlen,padding='post',truncating='post')  
 X2 = sequence.pad\_sequences(X2,maxlen=maxlen,padding='post',truncating='post')  
 return [X1,X2]

#### 1.2.6. 加载Bert模型和构建自己的模型

def build\_bert\_model(X1,X2):  
 *'''* ***:param*** *X1:经过编码过后的集合* ***:param*** *X2:经过编码过后的位置集合* ***:return****:模型  
 '''* # ！！！！！！ 非常重要的！！！非常重要的！！！非常重要的！！！  
 # 加载 Google 训练好的模型bert 就一句话，非常完美prefect  
 bert\_model = load\_trained\_model\_from\_checkpoint(config\_path, checkpoint\_path, seq\_len=None)  
 # config\_path 是Bert模型的参数，checkpoint\_path 是Bert模型的最新点，即训练的最新结果  
 # 特别注意的是 加载Bert的路径 问题，  
 # 注：https://storage.googleapis.com/bert\_models/2018\_11\_03/chinese\_L-12\_H-768\_A-12.zip，  
 # 下载完之后，解压得到4个文件，直接放到 项目的路径下，要写上绝对路径，以防出现问题。  
 # 安装 keras-bert：pip install keras-bert  
  
 x1 = Input(shape=(None,))  
 x2 = Input(shape=(None,))  
 wordvec = bert\_model.predict([x1,x2])  
 return wordvec

def build\_model():  
 model =Sequential()  
 model.add(Dense(128,activation=relu))  
 model.add(Dense(1,activation=sigmoid))  
 model.compile(loss=losses.binary\_crossentropy, optimizer=Adam(1e-5), metrics=['accuracy'])  
 model.summary()  
 return model

def train(wordvec,y):  
 model = build\_model()  
 model.fit(wordvec,y,batch\_size=32,epochs=10,validation\_split=0.2)  
 yaml\_string = model.to\_yaml()  
 with open('test\_keras\_bert.yml', 'w') as f:  
 f.write(yaml.dump(yaml\_string, default\_flow\_style=True))  
 model.save\_weights('test\_keras\_bert.h5')

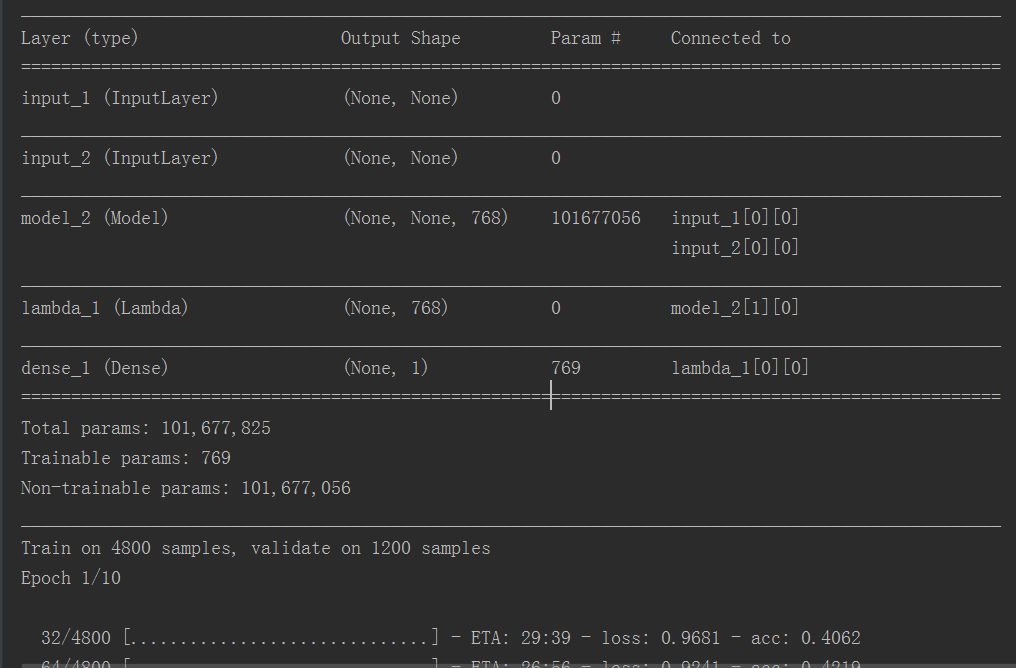


#### 1.2.7. 运行

if \_\_name\_\_ =='\_\_main\_\_':  
 pos,neg = get\_data()  
 token\_dict = get\_token\_dict(dict\_path)  
 # get\_encode()  
 [X1,X2] = get\_encode(pos,neg,token\_dict)  
 wordvec = build\_bert\_model(X1,X2)  
 # 标签类，其中选取3000个积极的文本和3000个消极的文本，将积极的记为1，将消极的记为0  
 y = np.concatenate((np.ones(3000, dtype=int), np.zeros(3000, dtype=int)))  
 # y = keras.utils.to\_categorical(y,num\_classes=2)  
 # p = Dense(2, activation='sigmoid')(x)  
 train(wordvec,y)

#### 1.2.8. 结果

由于训练时间较长，故停止了运行，程序是能跑的。



#### 1.2.9. 总结

使用Bert模型，最重要的还是特征提取，不了解Bert的小伙伴，去网上get，本文只介绍使用。

Bert与Word2vec相比，前者使用了以字为级别的向量，而word2vec则是以词语为级别的向量，而且，word2vec将一个词语的所有的意思都叠加到了一个词语的向量，而Bert只是参考上下文，进行语义提取，故不存在一词多义的问题。

项目开发流程：

1. 首先将以字为级别的数据集，进行编码
2. 将得到的编码喂入到训练好的Bert模型中。
3. 定义自己的模型，切记Bert模型后面不要加入太复杂的模型。

### 参考：

Keras-Bert： <https://kexue.fm/archives/6736#%E6%96%87%E6%9C%AC%E5%88%86%E7%B1%BB>