[**http://blog.csdn.net/churximi/article/details/61210129**](http://blog.csdn.net/churximi/article/details/61210129)

[**https://github.com/churximi/Keras**](https://github.com/churximi/Keras)

**Keras + LSTM + 词向量 情感分类/情感分析实验**

原创 2017年03月10日 23:26:40

* 5859
* 3
* 2

背景简介

本人是深度学习入门的菜菜菜鸟一枚…   
利用LSTM + word2vec词向量进行文本情感分类/情感分析实验，吸收了网上的资源和代码并尝试转化为自己的东西~

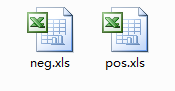
实验环境

* win7 64位系统
* Anaconda 4.3.0 , Python 2.7 version
* Pycharm开发环境
* python包：keras，gensim，numpy等

实验数据

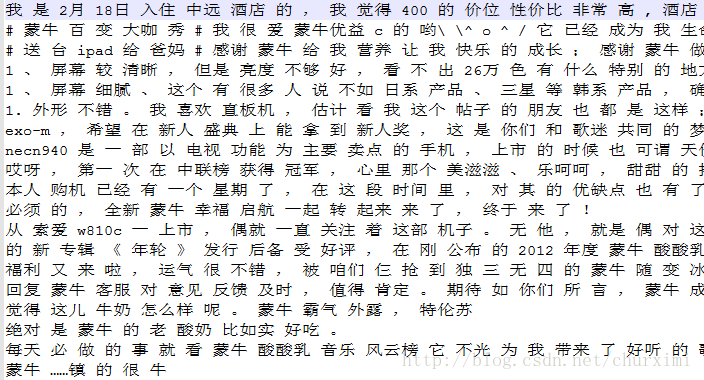
本文的实验数据是来自网上的中文标注语料，涉及书籍、酒店、计算机、牛奶、手机、热水器六个方面的购物评论数据，具体介绍参见该文：[购物评论情感分析](http://buptldy.github.io/2016/07/20/2016-07-20-sentiment%20analysis/)。

数据处理

上面提到的数据在网上见到的次数比较多，原始格式是两个excel文件，如图：   


对，就是这两个…估计来到本文的小伙伴也见过。一些代码就是直接从这两个excel里读取数据、分词、处理…不过我表示自己习惯从txt文本里获取数据，因此本人将数据合并、去重（原数据里有不少重复的评论）、分词（用的是哈工大LTP分词）之后存为一份txt文本，保留的数据情况如下：

正面评价个数：8680个   
负面评价个数：8000个

文本如图所示：   


然后人工生成一份【语料类别】文本，用1表示正面评价，用0表示负面评价，与评论数据一一对应。

文本数据下载地址：

<https://github.com/churximi/Keras>

生成词语的索引字典、词向量字典

利用上述文本语料生成词语的索引字典和词向量字典。   
注意：当Word2vec词频阈值设置为5时，词频小于5的词语将不会生成索引，也不会生成词向量数据。

工具：gensim里的Word2vec，Dictionary

代码

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

"""

功能：利用大语料生成词语的索引字典、词向量，然后保存为pkl文件

时间：2017年3月8日 13:19:40

"""

import pickle

import logging

import tkFileDialog

import numpy as np

np.random.seed(1337) # For Reproducibility

from Functions.TextSta import TextSta

from gensim.models.word2vec import Word2Vec

from gensim.corpora.dictionary import Dictionary

# 创建词语字典，并返回word2vec模型中词语的索引，词向量

def create\_dictionaries(p\_model):

gensim\_dict = Dictionary()

gensim\_dict.doc2bow(p\_model.vocab.keys(), allow\_update=True)

w2indx = {v: k + 1 for k, v in gensim\_dict.items()} # 词语的索引，从1开始编号

w2vec = {word: model[word] for word in w2indx.keys()} # 词语的词向量

return w2indx, w2vec

# 主程序

logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)

print u"请选择大语料的分词文本..."

T = TextSta(tkFileDialog.askopenfilename(title=u"选择文件"))

sentences = T.sen() # 获取句子列表，每个句子又是词汇的列表

print u'训练Word2vec模型（可尝试修改参数）...'

model = Word2Vec(sentences,

size=100, # 词向量维度

min\_count=5, # 词频阈值

window=5) # 窗口大小

model\_name = raw\_input(u"请输入保存的模型文件名...\n").decode("utf-8")

model.save(model\_name + u'.model') # 保存模型

# 索引字典、词向量字典

index\_dict, word\_vectors= create\_dictionaries(model)

# 存储为pkl文件

pkl\_name = raw\_input(u"请输入保存的pkl文件名...\n").decode("utf-8")

output = open(pkl\_name + u".pkl", 'wb')

pickle.dump(index\_dict, output) # 索引字典

pickle.dump(word\_vectors, output) # 词向量字典

output.close()

if \_\_name\_\_ == "\_\_main\_\_":

pass

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55

其中，

T = TextSta(tkFileDialog.askopenfilename(title=u"选择文件"))

sentences = T.sen() # 获取句子列表，每个句子又是词汇的列表

* 1
* 2

TextSta是我自己写的一个类，读取语料文本后，sentences = T.sen()将文本里的每一行生成一个列表，每个列表又是词汇的列表。（这个类原来是用作句子分类的，每行是一个句子；这里每行其实是一个评论若干个句子…我就不改代码变量名了…）

TextSta类部分代码：

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

"""

功能：一个类，执行文本转换

输入：分词文本

输出：句子列表，全文的词汇列表，TF，DF

时间：2016年5月17日 19:08:34

"""

import codecs

import re

import tkFileDialog

class TextSta:

# 定义基本属性，分词文本的全路径

filename = ""

# 定义构造方法

def \_\_init\_\_(self, path): # 参数path，赋给filename

self.filename = path

def sen(self): # 获取句子列表

f1 = codecs.open(self.filename, "r", encoding="utf-8")

print u"已经打开文本：", self.filename

# 获得句子列表，其中每个句子又是词汇的列表

sentences\_list = []

for line in f1:

single\_sen\_list = line.strip().split(" ")

while "" in single\_sen\_list:

single\_sen\_list.remove("")

sentences\_list.append(single\_sen\_list)

print u"句子总数：", len(sentences\_list)

f1.close()

return sentences\_list

if \_\_name\_\_ == "\_\_main\_\_":

pass

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41

总之，sentences的格式如下：

[[我, 是, 2月, …], [#, 蒙牛, 百, …], …]

所有的评论文本存为一个列表，每个评论文本又是词汇的列表。   
sentences列表的长度就是文本的行数：len(sentences) = 16680

利用Keras + LSTM进行文本分类

工具：Keras深度学习库

代码：

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

"""

功能：利用词向量+LSTM进行文本分类

时间：2017年3月10日 21:18:34

"""

import numpy as np

np.random.seed(1337) # For Reproducibility

import pickle

from keras.preprocessing import sequence

from keras.models import Sequential

from keras.layers.embeddings import Embedding

from keras.layers.recurrent import LSTM

from keras.layers.core import Dense, Dropout, Activation

from sklearn.cross\_validation import train\_test\_split

from Functions import GetLineList

from Functions.TextSta import TextSta

# 参数设置

vocab\_dim = 100 # 向量维度

maxlen = 140 # 文本保留的最大长度

batch\_size = 32

n\_epoch = 5

input\_length = 140

def text\_to\_index\_array(p\_new\_dic, p\_sen): # 文本转为索引数字模式

new\_sentences = []

for sen in p\_sen:

new\_sen = []

for word in sen:

try:

new\_sen.append(p\_new\_dic[word]) # 单词转索引数字

except:

new\_sen.append(0) # 索引字典里没有的词转为数字0

new\_sentences.append(new\_sen)

return np.array(new\_sentences)

# 定义网络结构

def train\_lstm(p\_n\_symbols, p\_embedding\_weights, p\_X\_train, p\_y\_train, p\_X\_test, p\_y\_test):

print u'创建模型...'

model = Sequential()

model.add(Embedding(output\_dim=vocab\_dim,

input\_dim=p\_n\_symbols,

mask\_zero=True,

weights=[p\_embedding\_weights],

input\_length=input\_length))

model.add(LSTM(output\_dim=50,

activation='sigmoid',

inner\_activation='hard\_sigmoid'))

model.add(Dropout(0.5))

model.add(Dense(1))

model.add(Activation('sigmoid'))

print u'编译模型...'

model.compile(loss='binary\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

print u"训练..."

model.fit(p\_X\_train, p\_y\_train, batch\_size=batch\_size, nb\_epoch=n\_epoch,

validation\_data=(p\_X\_test, p\_y\_test))

print u"评估..."

score, acc = model.evaluate(p\_X\_test, p\_y\_test, batch\_size=batch\_size)

print 'Test score:', score

print 'Test accuracy:', acc

# 读取大语料文本

f = open(u"评价语料索引及词向量.pkl", 'rb') # 预先训练好的

index\_dict = pickle.load(f) # 索引字典，{单词: 索引数字}

word\_vectors = pickle.load(f) # 词向量, {单词: 词向量(100维长的数组)}

new\_dic = index\_dict

print u"Setting up Arrays for Keras Embedding Layer..."

n\_symbols = len(index\_dict) + 1 # 索引数字的个数，因为有的词语索引为0，所以+1

embedding\_weights = np.zeros((n\_symbols, 100)) # 创建一个n\_symbols \* 100的0矩阵

for w, index in index\_dict.items(): # 从索引为1的词语开始，用词向量填充矩阵

embedding\_weights[index, :] = word\_vectors[w] # 词向量矩阵，第一行是0向量（没有索引为0的词语，未被填充）

# 读取语料分词文本，转为句子列表（句子为词汇的列表）

print u"请选择语料的分词文本..."

T1 = TextSta(u"评价语料\_分词后.txt")

allsentences = T1.sen()

# 读取语料类别标签

print u"请选择语料的类别文本...（用0，1分别表示消极、积极情感）"

labels = GetLineList.main()

# 划分训练集和测试集，此时都是list列表

X\_train\_l, X\_test\_l, y\_train\_l, y\_test\_l = train\_test\_split(allsentences, labels, test\_size=0.2)

# 转为数字索引形式

X\_train = text\_to\_index\_array(new\_dic, X\_train\_l)

X\_test = text\_to\_index\_array(new\_dic, X\_test\_l)

print u"训练集shape： ", X\_train.shape

print u"测试集shape： ", X\_test.shape

y\_train = np.array(y\_train\_l) # 转numpy数组

y\_test = np.array(y\_test\_l)

# 将句子截取相同的长度maxlen，不够的补0

print('Pad sequences (samples x time)')

X\_train = sequence.pad\_sequences(X\_train, maxlen=maxlen)

X\_test = sequence.pad\_sequences(X\_test, maxlen=maxlen)

print('X\_train shape:', X\_train.shape)

print('X\_test shape:', X\_test.shape)

train\_lstm(n\_symbols, embedding\_weights, X\_train, y\_train, X\_test, y\_test)

if \_\_name\_\_ == "\_\_main\_\_":

pass

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74
* 75
* 76
* 77
* 78
* 79
* 80
* 81
* 82
* 83
* 84
* 85
* 86
* 87
* 88
* 89
* 90
* 91
* 92
* 93
* 94
* 95
* 96
* 97
* 98
* 99
* 100
* 101
* 102
* 103
* 104
* 105
* 106
* 107
* 108
* 109
* 110
* 111
* 112
* 113
* 114
* 115
* 116
* 117
* 118
* 119
* 120
* 121
* 122
* 123

其中，

from Functions import GetLineList

* 1

GetLineList是自定义模块，用于获取文本的类别（存为列表），代码如下：

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

"""

功能：文本转列表，常用于读取词典（停用词，特征词等）

使用：给定一个文本，将文本按行转换为列表，每行对应列表里的一个元素

时间：2016年5月15日 22:45:23

"""

import codecs

import tkFileDialog

def main():

# 打开文件

file\_path = tkFileDialog.askopenfilename(title=u"选择文件")

f1 = codecs.open(file\_path, "r", encoding="utf-8")

print u"已经打开文本：", file\_path

# 转为列表

line\_list = []

for line in f1:

line\_list.append(line.strip())

print u"列表里的元素个数：", len(line\_list)

f1.close()

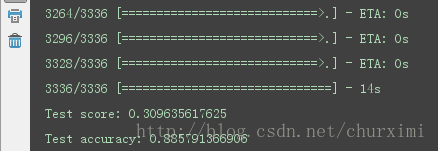
return line\_list

if \_\_name\_\_ == "\_\_main\_\_":

pass

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30

实验结果



参考文献:

<http://buptldy.github.io/2016/07/20/2016-07-20-sentiment%20analysis/>   
<https://github.com/BUPTLdy/Sentiment-Analysis>