

**模式识别大作业**

题 目 垃圾短信分类问题

学 院 信息科学与工程

专 业 控制科学与工程

姓 名 朱思敏

学 号 Y30180700

指导教师 赵海涛

**完成日期： 2018 年 10 月25日**

**模式识别大作业报告—垃圾短信分类问题**

姓名：朱思敏

学号：Y30180700

**一、问题描述：**

本次大作业所作的题目是lintcode上提供的十一道AI练习题中所选的垃圾短信分类问题。

本题提供了一个数据集train.csv,它包括了5574条英文短信，每条短信内容由几个长短不一的句子组成。每条短信都标注好了是否为垃圾短信(ham或spam)，通过对该训练集进行训练得到一个分类器，进而预测本题提供的测试集test.csv中的短信内容是否为垃圾短信，将测试好的结果存入submission.csv文件中，进行上传，用准确率来评价分类的结果。

**二、问题解决思路：**

1. TF-IDF方式来构建词向量

首先需要统计数据集中的单词出现的次数，可以使用分词工具，然后将句子转换成词向量，建立词表。简单的，可以通过统计词汇表中词汇出现的次数构建向量。但是上述这种简单地构建向量的方式，会导致每个词的重要过于均衡，无法体现词向量中词汇的重要性，导致分类结果不理想。改进中，可以采用TF-IDF方式来构建词向量。

TF-IDF（term frequency–inverse document frequency）是一种用于信息检索与数据挖掘的常用加权技术。TF的意思是词频(Term Frequency)，IDF的意思是逆文本频率指数(Inverse Document Frequency) 。某一特定词语的IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到。某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的TF-IDF。因此，TF-IDF倾向于过滤掉常见的词语，保留重要的词语。

词的IDF公式为：

其中代表语料库中文本的总数，代表语料库中包含词的文本总数。

在一些特殊的情况下上面的公式会存在一些小的问题，比如某一个生僻词在语料库中没有，则分母变为0，IDF就没有意义了，所以常用的IDF需要做一些平滑，使得语料库中没有出现的词也可以得到一个合适的IDF值，平滑的方法有很多种，小修之前为大家介绍过，最常见的IDF平滑公式：

接着就可以计算一个词的TF-IDF值：

其中是指词在当前文本的词频。这个数字是对词数的归一化，以防止它偏向长的文件，（同一个词语在长文件里可能会比短文件有更高的词数，而不管该词语重要与否）。对于某一特定文件里的词语来说，它的重要性可表示为：

在本次大作业中，采用调用scikit-learn包中的TfidfTransformer函数进行TF-IDF操作。

2. 朴素贝叶斯实现分类

当完成数据的预处理工作后，就该考虑采用何种方式进行分类。分类问题可以看作一种决策，即我们可以根据观测结果对待测样本做出相应归属哪一类的决策。

贝叶斯决策是常用的一种分类器，概率论中的贝叶斯公式为

其中是先验概率，是关于的似然函数（likelihood），称为证据因子（evidence），它是一个标量因子，保证各类的后验概率和为1。

假设的不同维度相互独立，即分类问题的各个特征是相互独立的，则似然函数可以表示为下式

此时我们将这样的分类成为朴素贝叶斯(Naïve Bayes)分类。

**三、程序说明以及程序实现：**

1. 导入所需调用的包

import pandas as pd #导入pandas包用于数据集的读取

import numpy as np #导入numpy用于矩阵计算

import nltk

from nltk.tokenize import word\_tokenize

from sklearn.feature\_extraction.text import TfidfVectorizer

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

# train\_test\_split函数将数据集中的数据按一定比例划分为训练集和校验集

import re #导入正则表达式模块

2. 数据导入

#数据的导入

data\_train=pd.read\_csv('train.csv',names=['Label', 'Text']) #读取训练数据

data\_test=pd.read\_csv('test.csv',names=['Id', 'Text']) #读取测试数据

#加上names可以解决读进来错乱的问题

#读出来的数据会多第一行Label、Text，用header=None也没办法解决，只能下来再删掉第一行

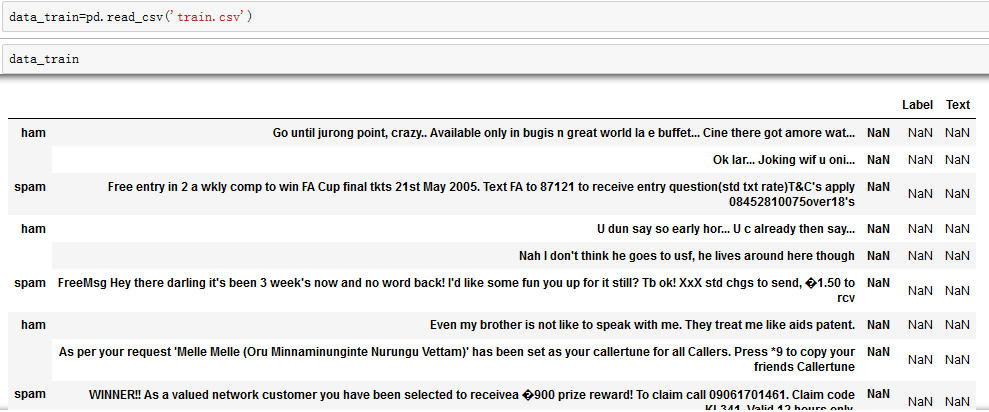
data\_train=data\_train.drop([0])

data\_test=data\_test.drop([0])

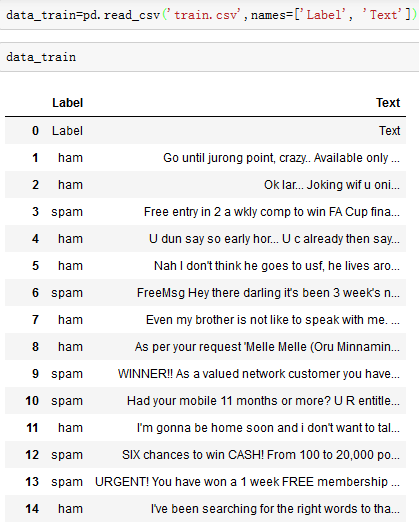
X\_total=pd.concat([data\_train,data\_test])

#将训练测试集合并起来，统一进行数据处理

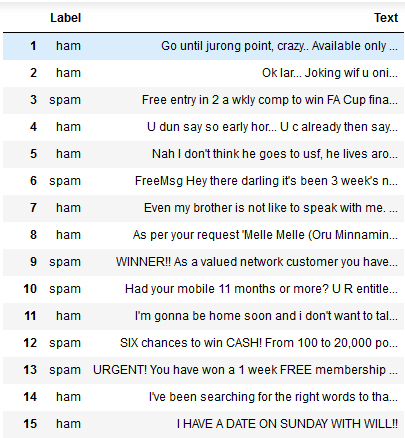
本次实验数据导入的操作主要使用了Pandas模块，可将数据集读取为DataFrame的形式。在读取的过程中使用pd.read\_csv('train.csv')命令进行操作时，读出的数据产生了下图的问题：



这时考虑对pd.read\_csv中的参数进行设置，加入names设置列名，用命令行data\_train=pd.read\_csv('train.csv',names=['Label', 'Text']) 进行数据导入，得到下图：



可以发现多了第一行列名，这时用data\_train=data\_train.drop([0]) 命令删去第一行，得到正确导入的数据，如下图：



3. 数据预处理

数据集中的短信不能直接拿来测试，需要经过一定的预处理，因为其中可能包含很多训练用不到的数据，例如标点符号、数字，并且我们还需要将英文的大写转换成小写，缩写转换为非缩写的形式等。定义如下的数据清洗函数clean\_text()：

def clean\_text(message\_text):

message\_list=[]

s = nltk.stem.snowball.EnglishStemmer() # 词干提取

for text in message\_text:

text = re.sub('���', "'", text)

text = re.sub('��', "'", text)

text = re.sub('�', "'", text)

text = text.lower() # 小写转换

# 删除非字母、数字字符

text = re.sub(r"[^a-z']", " ", text)

# 恢复常见的简写

text = re.sub(r"what's", "what is ", text)

text = re.sub(r"\'s", " ", text)

text = re.sub(r"\'ve", " have ", text)

text = re.sub(r"can't", "can not ", text)

text = re.sub(r"cannot", "can not ", text)

text = re.sub(r"n't", " not ", text)

text = re.sub(r"\'m", " am ", text)

text = re.sub(r"\'re", " are ", text)

text = re.sub(r"\'d", " will ", text)

text = re.sub(r"ain\'t", " are not ", text)

text = re.sub(r"aren't", " are not ", text)

text = re.sub(r"couldn\'t", " can not ", text)

text = re.sub(r"didn't", " do not ", text)

text = re.sub(r"doesn't", " do not ", text)

text = re.sub(r"don't", " do not ", text)

text = re.sub(r"hadn't", " have not ", text)

text = re.sub(r"hasn't", " have not ", text)

text = re.sub(r"\'ll", " will ", text)

new\_text = ""

for word in word\_tokenize(text):

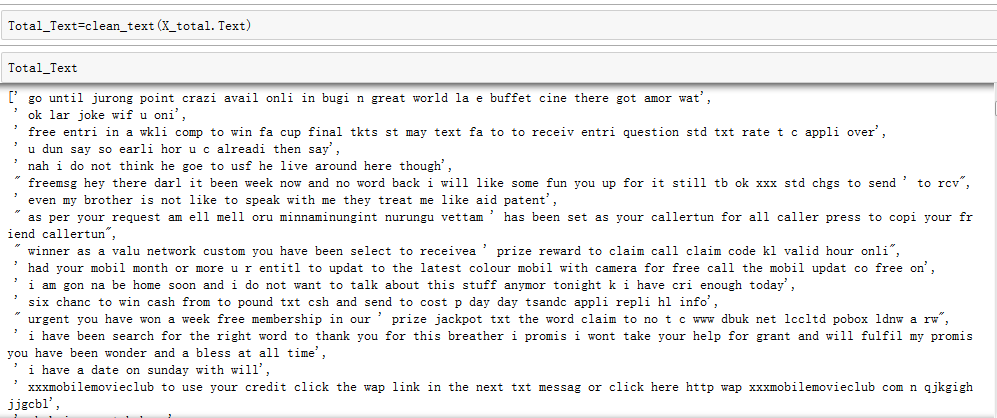
new\_text = new\_text + " " + s.stem(word)

message\_list.append(new\_text)

return message\_list

Total\_Text=clean\_text(X\_total.Text)

清洗后的短信：



x\_total=np.array(Total\_Text)

y=data\_train.Label.map({'ham':0, 'spam':1}) #令 ham=0,spam=1

数据集中的Label是用ham表示非垃圾短信，spam表示垃圾短信，为了训练模型，需要将Label用数字进行表示，令 ham=0,spam=1。

# 数据的TF-IDF信息计算

total\_text\_list = list(Total\_Text)

text\_vector = TfidfVectorizer(sublinear\_tf=True, strip\_accents='unicode',token\_pattern=r'\w{1,}',

max\_features=5000, ngram\_range=(1, 1), analyzer='word')

text\_vector.fit(total\_text\_list)

x\_total = text\_vector.transform(Total\_Text)

x\_total=x\_total.toarray()

x=x\_total[0:5572]

test\_X=x\_total[5572:6687]

4.朴素贝叶斯分类

def NaiveBayes(x,y):

x=astype('float64')

N,M=x.shape #N为短信数，M为所有短信中出现的单词数

P\_prior=sum(y)/N #计算训练集中短信为垃圾短信的 先验概率

P\_0\_total=1

#用P\_0\_total记录每条非垃圾短信的总词汇数，初始化为1，避免分母为0

P\_1\_total=1

#用P\_1\_total记录每条垃圾短信的总词汇数

#将所有词的出现次数初始化为1

P\_0\_count=np.ones(M)

P\_1\_count=np.ones(M)

for i in range(N):

if y[i]==1:

#y[i]=1时为垃圾短信，记录短信中单词的词频

P\_1\_count+=np.array(x[i])

P\_1\_total+=sum(x[i])

else:

P\_0\_count+=np.array(x[i])

#y[i]=0时为非垃圾短信，记录短信中单词的词频

P\_0\_ total +=sum(x[i])

#计算likelihood

P\_0\_condition=P\_0\_count/P\_0\_total

P\_1\_condition=P\_1\_count/P\_1\_total

#为保证概率不会过小，取对数

P\_0\_condition=np.log(P\_0\_condition)

P\_1\_condition=np.log(P\_1\_condition)

return P\_prior,P\_0\_condition,P\_1\_condition

P\_p,P\_0\_c,P\_1\_c=NaiveBayes(recon\_train\_X,train\_Y)

# 贝叶斯分类器

def predict(x,P\_prior,P\_0\_condition,P\_1\_condition):

pred = []

for j in x:

P\_0=sum(j\* P\_0\_condition)+np.log(1-P\_prior)

P\_1=sum(j\* P\_1\_condition)+np.log(P\_prior)

if P\_1> P\_0:

pred.append(1)

else:

pred.append(0)

return np.array(pred)

5．模型的训练和测试：

# 将数据分为训练集和校验集

train\_X, valid\_X, train\_Y, valid\_Y = train\_test\_split(x, y.values, test\_size=0.25)

Y\_pred=predict(valid\_X,P\_p,P\_0\_c,P\_1\_c)

from sklearn.metrics import accuracy\_score

print('Accuracy score: ', format(accuracy\_score(valid\_Y, Y\_pred)))

#计算校验集的准确率

Y\_test=predict(test\_X,P\_p,P\_0\_c,P\_1\_c) #对测试集进行分类

6. 将训练好的数据存成文件：

data\_test['Label']=''

data\_test.drop(['Text'],axis=1,inplace=True)

for index in range(len(data\_test.Id)):

if Y\_test[index]==1:

data\_test.loc[index,'Label']='spam'

else:

data\_test.loc[index,'Label']='ham'

Id=data\_test.Id

dataset=list(zip(Id,data\_test.Label))

df=pd.DataFrame(data=dataset,columns=('SmsId','Label'))

#将数据写入csv文件中，不需要索引列

df.to\_csv('submission.csv',index=False)

**四、作业心得：**

通过本次大作业，使我对模式识别学习中的主成分分析和朴素贝叶斯算法有了更为深刻的理解，更让我体会到理论和程序的编写是分不开的，我们应先打好扎实的理论基础，并动手动脑编写程序得以应用才能对知识有更深刻的体会。