

Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

Read the guide

langgege-cqu / ml_learn

 Unwatch


1


 Star


0


 Fork


0


 Code


 Issues


 Pull requests


 Actions


 Projects

 Wiki

 Security

 Insights

 Settings

 master

ml_learn / homework_2 / homework2.ipynb

Go to file

...






 fl lab3

Latest commit 30e08d0 1 minute ago

 History

0 contributors

298 lines (298 sloc) | 7.78 KB

  Raw Blame   

In [1]:

得到词典和特征向量

```
In [2]: import os
import numpy as np
from collections import Counter
from numpy import *
from functools import reduce
import matplotlib.pyplot as plt
from sklearn import metrics

def make_Dictionary(train_dir):
    emails = [os.path.join(train_dir, f) for f in os.listdir(train_dir)]
    all_words = []
    for mail in emails:
        with open(mail) as m:
            for i, line in enumerate(m):
                if i == 2:
                    words = line.split()
                    all_words += words
    dictionary = Counter(all_words)
    list_to_remove = [key for key in dictionary.keys()]
    for item in list_to_remove:
        if item.isalpha() == False:
            del dictionary[item]
        elif len(item) == 1:
            del dictionary[item]
    dictionary = dictionary.most_common(3000)
    return dictionary

train_dir, test_dir = 'train-mails', 'test-mails'
dictionary = make_Dictionary(train_dir)

def extract_features(mail_dir):
    files = [os.path.join(mail_dir, fi) for fi in os.listdir(mail_dir)]
    features_matrix = np.zeros((len(files), 3000))
    docID = 0
    for fil in files:
        with open(fil) as fi:
            for i, line in enumerate(fi):
                if i == 2:
                    words = line.split()
                    for word in words:
                        for i, d in enumerate(dictionary):
                            if d[0] == word:
                                wordID = i
                                features_matrix[docID, wordID] = 1

                    docID = docID + 1
    return features_matrix

train_features = extract_features(train_dir)
test_features = extract_features(test_dir)
print('endl')
```

endl

生成训练测试label

```
In [3]: def get_label(mail_dir):
files = [os.path.join(mail_dir, fi) for fi in os.listdir(mail_dir)]
label = [1 for _ in range(len(files))]
for i, file in enumerate(files):
    if 'spam' in file:
        label[i] = 0
    return label

train_label = get_label(train_dir)
test_label = get_label(test_dir)

print('end2')
```

end2

定义训练函数，训练数据集，预测测试集的情况

```
In [6]: def trainNB(trainMatrix, trainClass):
numTrainClass = len(trainClass)
numWords = len(trainMatrix[0])
p0Num = ones(numWords)
p1Num = ones(numWords)
p0Words = 2.0
p1Words = 2.0
for i in range(numTrainClass):
    if trainClass[i] == 1:
        # 数组在对应的位置上相加
        p1Num += trainMatrix[i]
        p1Words += sum(trainMatrix[i])
    else:
        p0Num += trainMatrix[i]
        p0Words += sum(trainMatrix[i])

p0Vec = log(p0Num / p0Words)
p1Vec = log(p1Num / p1Words)
# 计算在类别中1出现的概率，0出现的概率可通过1-p得到
pClass1 = sum(trainClass) / float(numTrainClass)
return p0Vec, p1Vec, pClass1

def classifyNB(testVec, p0Vec, p1Vec, pClass1):
p1 = sum(testVec * p1Vec) + log(pClass1)
p0 = sum(testVec * p0Vec) + log(1 - pClass1)
if p0 > p1:
    return 0
return 1

p0V, p1V, pClass1 = trainNB(train_features, train_label)
pre_test_label = []
for i in range(len(test_label)):
    testClass = classifyNB(test_features[i], p0V, p1V, pClass1)
    pre_test_label.append(testClass)

print('end3')
```

end3

计算准确率、召回率、f1-score

```
In [5]: def get_value(pre_test_label, test_label):
count = 0
for i in range(len(pre_test_label)):
    if pre_test_label[i] == test_label[i]:
        count += 1
accuracy = count / len(test_label)
count1 = 0
for i in range(len(pre_test_label)):
    if pre_test_label[i] == 1 and test_label[i] == 1:
        count1 += 1

count2 = 0
for i in range(len(pre_test_label)):
    if test_label[i] == 0 and pre_test_label[i] == 1:
        count2 += 1
precision = count1 / (count1 + count2)
count3 = 0
for i in range(len(pre_test_label)):
    if test_label[i] == 1 and pre_test_label[i] == 0:
        count3 += 1
recall = count1 / (count1 + count3)
print(precision)
f1_score = 2 * (precision * recall) / (precision + recall)
return accuracy, recall, f1_score

accuracy, recall, f1_score = get_value(pre_test_label, test_label)

print('accuracy:', accuracy, 'recall:', recall, 'f1_score:', f1_score)

0.9416058394160584
accuracy: 0.9653846153846154 recall: 0.9923076923076923 f1_score: 0.9662921348314606
```