Assignment 8

Joshua Graham

April 10, 2018

Abstract

All Scripts used in the assignment can be found in the A8 folder, if needed.

0.1 Problem 1

Question 1 was pretty much a manual question. Grab 20 spam emails and 20 regular emails. Divide them up into testing and training sets. This was about as hard as question 2. This is becuase ODU email accounts automatically delete spam emails after 30 days. I had 6 emails, 3 of which aren't usable. So if you examine the emails, you may notice references to a Karen and/or Scott Graham. These are my parents, I asked to use their email for spam messages and they gave the okay. With these and a few from one of my other gmail accounts, I managed to scrape together 20 spam emails. To keep the data set related, I also grabbed a few of their emails from the not spam sections to keep the data set from being skewed. There was no automation in this question so I have no code to show. **However,** during question 2, 19 out of 20 emails were listed as not spam, so instead of recreating the data, I swapped half the files from each data set which produced a more reasonable dataset.

0.2 Problem 2

Question 2 had code. Namely, I had to correctly implement test.py and change the code slightly in docclass.py. Starting with docclass.py, it was relativley simple. I used the glob import to find all the .txt files under each respective training directory, then put them into a variable. This is what I passed to cl.train. The changed method is shown below.

Training

```
def spamTrain(cl):
    path = "Training/NotSpam/*.txt"
    for filename in glob.glob(path):
        with open(filename, 'r') as myfile:
            print("Training file: ", filename)
            data = myfile.read()
            cl.train(data, 'not spam')
    path = "Training/Spam/*.txt"
    for filename in glob.glob(path):
        with open(filename, 'r') as myfile:
            print("Training file: ", filename)
            data = myfile.read()
            cl.train(data, 'spam')
```

After changing docclass.py I had to correctly implement test.py. This wasn't a challenge, all I had to do was point at the testing files at the right time. I again used the glob import, for no reason other than I like using wildcard and it got the job done. test.py can be shown below.

test.py

```
import docclass
import glob
from subprocess import check_output

cl = docclass.naivebayes(docclass.getwords)
try:
  check_output(['rm', 'spam.dm'])
except:
  print("No file")
#remove previous db file
cl.setdb('spam.db')
docclass.spamTrain(cl)
spampath="Testing/Spam/*.txt"
notspampath="Testing/NotSpam/*.txt"

print("Testing against spam emails.")
for filename in glob.glob(spampath):
    with open(filename, 'r') as myfile:
        data = myfile.read()
        print("Filename: ", filename, "Status: ", cl.classify(data))
print("Testing against regular emails.")
for filename in glob.glob(notspampath):
    with open(filename, 'r') as myfile:
        data = myfile.read()
        print("Filename: ", filename, "Status: ", cl.classify(data))
```

All in all, I think I'm happy with how it turned out. After running the code, this was the output:

test.py

```
1 Filename:  Testing/Spam/18.txt Status:  not spam
2 Filename:  Testing/Spam/20.txt Status:  spam
3 Filename:  Testing/Spam/3.txt Status:  spam
4 Filename:  Testing/Spam/7.txt Status:  spam
5 Filename:  Testing/Spam/9.txt Status:  spam
6 Filename:  Testing/Spam/12.txt Status:  not spam
7 Filename:  Testing/Spam/5.txt Status:  spam
8 Filename:  Testing/Spam/14.txt Status:  spam
9 Filename:  Testing/Spam/16.txt Status:  spam
10 Filename:  Testing/Spam/1.txt Status:  spam
11 Testing against regular emails.
12 Filename:  Testing/NotSpam/18.txt Status:  not spam
13 Filename:  Testing/NotSpam/20.txt Status:  not spam
14 Filename:  Testing/NotSpam/3.txt Status:  not spam
15 Filename:  Testing/NotSpam/7.txt Status:  not spam
16 Filename:  Testing/NotSpam/9.txt Status:  not spam
17 Filename:  Testing/NotSpam/12.txt Status:  not spam
18 Filename:  Testing/NotSpam/5.txt Status:  not spam
19 Filename:  Testing/NotSpam/14.txt Status:  not spam
20 Filename:  Testing/NotSpam/16.txt Status:  not spam
21 Filename:  Testing/NotSpam/1.txt Status:  not spam
```

I think there was a bit of skew because of using someone elses emails. But it correctly represented all but 2 emails. On a larger data set, the error would probably be larger. However, for the given example, it appears to work decently well.

0.3 Problem 3

Problem 3 was to create a confusion matrix for the results. The matrix assumes that correctly returning spam is a true positive, and correctly returning not spam is a true negative. It can be shown below:

	Returned Positive	Returned Negative
Expected Positive	8 or 80%	2 or 20%
Expected Negative	0 or 0%	10 or 100%