Assignment 8

Dr. Nwala

Nathaniel Everett

April 8, 2018

Contents:

1. Create two datasets; the first called Testing, the second called Training.

The Training dataset should:

a. consist of 10 text documents for email messages you consider spam (from your spam folder)

b. consist of 10 text documents for email messages you consider not spam (from your inbox)

The Testing dataset should:

a. consist of 10 text documents for email messages you consider spam (from your spam folder)

b. consist of 10 text documents for email messages you consider not spam (from your inbox)

Upload your datasets on github

The datasets are named accordingly and are actually directories in the Assignment 8 directory. The email address I used was my own ODU email (never004@odu.edu) and I had at least 20 spam and nonspam messages for a total of 40 messages to use for each directory respectively. Each message in the Training directory had its text copied and pasted into an individual text file, labeled spam#.txt or notspam#.txt, where # is the number from 1 to 10. The Testing directory contains 20 different messages, each with email# with # being between 1 and 20.

---

2. Using the PCI book modified docclass.py code and test.py (see Slack assignment-8 channel)
Use your Training dataset to train the Naive Bayes classifier (e.g., docclass.spamTrain())
Use your Testing dataset to test (test.py) the Naive Bayes classifier and report the classification results.

I copied both the docclass.py and test.py code for modification and use of this part of the assignment. The python files I used were NaiveBayesClassifier.py and ClassifierTest.py, respectively. Most of the code in these two scripts remained unmodified. The only parts of the code that were changed were the ones involving docclass's spamTrain() function, and I also removed a number of unnecessary functions for the classifier function. Here is a code snippet from the spamTrain() function:

```
def spamTrain(c)
  f1 = open('Training\\notspam1.txt')
  tr1 = f1.read()
  c.train(tr1, 'Not Spam')
```

This opens one of the 20 files, reads it, and categorizes it as either 'Spam' or 'Not Spam'. The remainder of spamTrain() has the same code snippet as above, except for the other 19 txt files. Note that there is no output from NaiveBayesClassifier.py, but the code works properly. Now for ClassifierTest.py, again most of the code is unmodified from test.py apart from the setdb() and check_output functions, which I modified for my own use (neverett.db). The first time I ran this, I commented out this line:

```
check_output(['rm', 'neverett.db'])
```

as it would not detect the file in question the first time. After I run it, the program worked properly and the proper output file was produced. Note that my db file is full of 4-bit values.

The spamTrain() function is called from NaiveBayesClassifier.py, this time we are taking the txt

files from the Testing Directory. After calling it, I open the 20 files in the Testing directory; an example code snippet is given for the first one:

```
f1 = open('Testing\\email1.txt')
e1 = f1.read()
print(c.classify(e1))
```

with the number changing as it goes along. The console output of the python script is given below, and it shows ten spam and ten non-spam files:

```
Not Spam
Not Spam
Not Spam
Spam
Spam
Not Spam
Not Spam
Not Spam
Not Spam
Not Spam
Spam
Spam
Spam
Spam
Not Spam
Not Spam
Spam
Spam
Spam
Spam
```

Files included:
Assignment8Report.pdf – this file
Training – directory containing ten spam and ten non-spam email messages from my account
Testing - directory containing ten spam and ten non-spam email messages from my account
docclass.py – used for reference and modified for NaiveBayesClassifier.py
test.py – used for reference and modified for ClassifierTest.py
NaiveBayesClassifier – modified docclass.py for Problem 2
ClassifierTest.py – modified test.py for Problem 2
neverett.db – db output from ClassifierTest.py

Files included: