CS432 Spring 2018

Assignment 6

Jonathan Kruszewski

04/09/18

CS 432/532 Web Science
Spring 2018
http://anwala.github.io/lectures/cs532-s18/

Assignment #8
Due: 11:59pm April 8

Support your answer: include all relevant discussion, assumptions, examples, etc.(10 points)

(Spam classification using Naive Bayes classifier)
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

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.

==========
========Each question below is for 3 points extra credit==========
==========

3. Draw a confusion matrix for your classification results
(see: https://en.wikipedia.org/wiki/Confusion_matrix)

4. Report the precision and accuracy scores of your classification results
(see: https://en.wikipedia.org/wiki/Precision_and_recall)

Part 1:

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 files used for this task can be found in two folders, “Training” and “Testing”, in each folder there are 20 text files, 10 spam and 10 nonspam, they are titled “<0-9>Spam.txt” and “<0-9>NotSpam.txt”

Example of non-spam file:

Your account Jonathan.Kruszewski@gmail.com is listed as the recovery email for aod43254@gmail.com. Don't recognize this account? Click here
New sign-in to your linked account
aod43254@gmail.com
Your Google Account was just signed in to from a new Linux device. You're getting this email to make sure it was you.
CHECK ACTIVITY
You received this email to let you know about important changes to your Google Account and services.
© 2018 Google Inc.,1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

example of spam file:

WANT TO DATE RUSSIAN AND UKRAINIAN BEAUTIES?

Part 2:

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.

This task was accomplished by using the docclass.py and test.py files provided in Slack. The docclass.py file was modified by adding a new function emailTrain, seen below, which calls the train function for each of the training files.

```
for x in range(10):
    with open('/home/jonathan/PycharmProjects/Assignment8/Training/NotSpam'+str(x)+'.txt') as
nSpamIn:
        cl.train(nSpamIn.read(), 'Not_Spam')
    with open('/home/jonathan/PycharmProjects/Assignment8/Training/Spam'+str(x)+'.txt') as spamIn:
        cl.train(spamIn.read(), 'Spam')
```

test.py then calls the classify function for each of the test documents, in a similar way as the code above for emailTrain. test.py then outputs the results of classify to a text file, “Output.txt”

Part 3/4

3. Draw a confusion matrix for your classification results
(see: https://en.wikipedia.org/wiki/Confusion_matrix)
4. Report the precision and accuracy scores of your classification results
(see: https://en.wikipedia.org/wiki/Precision_and_recall)

To accomish these tasks a python program “CheckOutput.py” was used. “CheckOutput.py” reads the “Output.txt” file and checks each result and adds up all of the false/true positive/negatives. The formula found on the wikipedia page for precision and recall are then used to find the precision and recall of the results.

	Neg	Pos
True:	9	5
False:	5	1

Precision: 0.8333333333333334
Recall: 0.5