HW9 CS432

Ondra Torkilson

December 13, 2020

Q1

I first chose to classify between 'news' and 'notNews' documents, but because I didn't use boilerplate to clean the email documents, I ended up with files that were crashing my program when the classifier tried to getwords from the documents.

I decided to change my categories to 'Weigle' and 'Not Weigle,' and I copy and pasted plain text emails to new documents. This allowed me to move forward with the project. The full list of files is available on the github repo for this assignment.

Q2

Let's look at the code to implement the classifier on my data. First, I defined a sampletrain function which loops through both of my training data sets and trains the classifier on each document. For emails from Dr.Weigle, the label is 'Weigle,' and for emails from anyone else, the label is 'Not Weigle.' After that, I instantiate the naivebayes classifier on the getwords function. Then, I can use my sampletrain function to pass the classifier and train it. From there, I classified all of my test documents. I output the results of the classification for each document to the console. We can see an image of the results on the following page.

```
def sampleTrain(c1):
    with os.scandir('trainWeigle/') as entries:
        for entry in entries:
            filein = open(entry, 'r', encoding='utf-8')
            doc = filein.read()
            filein.close()
            c1.train(doc,'Weigle')

    with os.scandir('trainNotWeigle/') as entries:
        for entry in entries:
            filein = open(entry, 'r', encoding='utf-8')
            doc = filein.read()
            filein.close()
            c1.train(doc,'notWeigle')


c1 = naivebayes(getwords)
sampleTrain(c1)

pp.pprint('Classifying Weigle: ')
with os.scandir('testWeigle/') as entries:
    for entry in entries:
        print('\n')
        filein = open(entry, 'r', encoding='utf-8')
        doc = filein.read()
        filein.close()
        pp.pprint(c1.classify(doc,default='unknown'))

pp.pprint('Classifying Not Weigle')
with os.scandir('testNotWeigle/') as entries:
    for entry in entries:
        print('\n')
        filein = open(entry, 'r', encoding='utf-8')
        doc = filein.read()
        filein.close()
        pp.pprint(c1.classify(doc,default='unknown'))
```

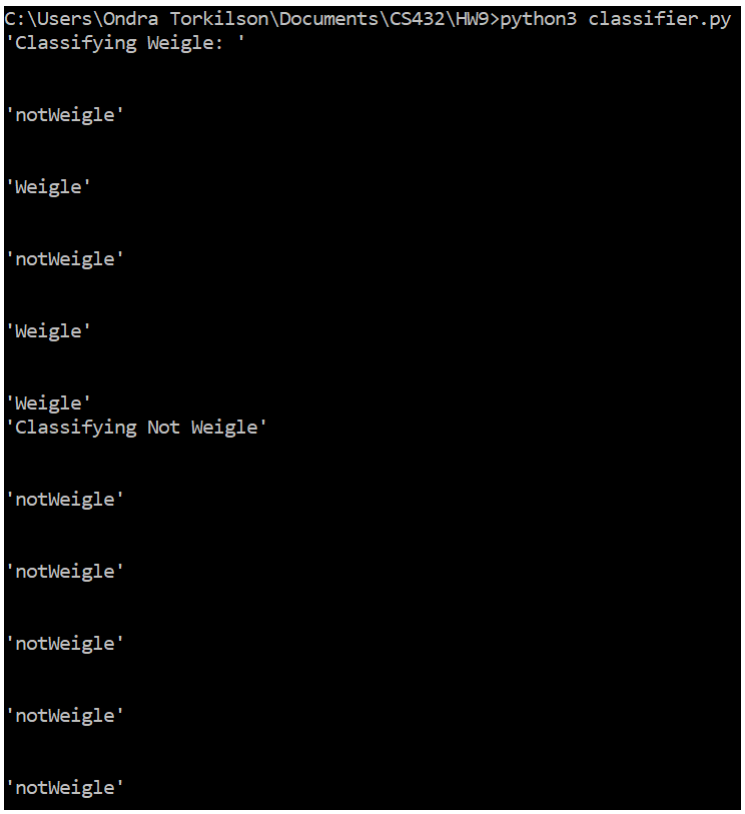

Figure 1: Results of Classifying Ten Documents

From this image we can see that all of the 'Not Weigle' documents were correctly classified. Two of the 'Weigle' documents were misclassified as 'Not Weigle.' So overall, five true negatives, two false negatives, and three true positives. We will explore this further on the next page in the confusion matrix.

Q3 Q4

Table 1: Confusion Matrix

		Predicted: Weigle	Predicted: Not Weigle
Actual	Weigle	3 (TP)	2 (FN)
	Not Weigle	0 (FP)	5 (TN)

For the precision score, we can take the true positive cell divided by the test outcome positives, and we get 3/3 = 1.0 for precision.

Accuracy score is the sum of true positive plus the true negative divided by the total population, which is 3+5/10 = .8 or 80% accuracy.

References

[1] Segaran, T. (2007). Programming Collective Intelligence. O'Reilly Media. Retrieved December 13, 2020, from https://learning.oreilly.com/library/view/programming-collective-intelligence/9780596529321/ch03.htmlsupervised_versus_unsupervised_learning

[2] Weigle, M. (2020). 432-PCI-Ch06.ipynb. Retrieved December 13, 2020, from https://colab.research.google.com/github/cs432-websci-fall20/assignments/blob/master/432_PCI_Ch06.ipynbscrollTo=0hSyFIKnrRly