CS 432: Homework Number Eight

Due on April 13, 2019 at 4:20 PM

Alexander Nwala

Tim Bruce

Problem 1

Create two datasets; the first called Testing, the second called Training.
The Training dataset should:

- Consist of 10 text documents for email messages you consider spam (from your spam folder)
- Consist of 10 text documents for email messages you consider not spam (from your inbox)

The Testing dataset should:

- Consist of 10 text documents for email messages you consider spam (from your spam folder)
- Consist of 10 text documents for email messages you consider not spam (from your inbox)

Upload your datasets on github Please do not include emails that contain sensitive information

Solution
This one is pretty self explanatory. I found the emails and copy pasted them to various text files on my computer organized into four files of ten based on if they were spam. It was hard to find enough non-spam emails that were not too personal to put on GitHub. The dataset does have one notable curveball. Two emails from the same source that talk about very similiar topics, one of which is spam and one of which is not. Specifically, one of them is a daily report from my favorite ski mountain, Sugarloaf. The other email is a notification that they have had a significant amount of snow. I want the snow warning, because it may change driving conditions en-route to the mountain, and skiing conditions will be significantly changed as well (not always for the better). The daily report is garbage (well written, but useless to receive every day)

Problem 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.

Solution
I got the files from the slack, and edited test.py to fit my needs. In order to pull up the dataset, I created the get_emails function to get the dataset.

Listing 1: get_emails function.

```
def get_emails(directory):
    a = [directory + x for x in os.listdir(directory)]
    emails = []
    for email in a:
        f = open(email)
        string = ""
        for line in f.readlines():
            string += line
        emails.append(string)
        f.close()
    return emails
```

Problem 2 continued on next page...

Then, I train the Bayes classifier using the training data set, and classify the other emails. It should be noted that the training set is alternating entry, with one being spam followed by not spam. This is to prevent overloading of the last one to train for. Even though this randomization is in place, for my entire testing dataset, the classifier thinks that it is spam. There could be several causes:

- My testing and training datasets are too disjoint, and the not spam testing values are similar to my spam messages from the training.
- The to from and subject fields are messing up the content, and skewing it towards spam.
- A combination of the above two.

Either way, the classifier is not useful if everything is spam, but it does clean up your inbox really well.

Problem 3

Draw a confusion matrix for your classification results.

Solution
I have completed this. Due to my lack of useful results, it is not extremely interesting, but it does display why the data is not useful in a very easy to read way. It has been a while since I made a table in LaTeX. I remember why I avoid it now.

		Actual Class	
		Spam	Not Spam
Predicted Class	Spam	10 True Positives	10 False Positives
	Not Spam	0 False Negatives	0 True Negatives

As you can see, my classifier is always returning spam, which seems to make it 50% accurate (at least it is not **worse** than randomly guessing), it is in fact just returning the same value every time. This shows the value of a chart like this. Because it shows false negatives and false positives, it is possible to see which result the classifier is biased to.

Problem 4

Report the precision and accuracy scores of your classification results.

Solution
Precision is the same as the positive predictive value, which can be found using the following equation according to the provided Wikipedia article.

$$Precision = \frac{|\{Relevant\ Documents\} \cap \{Retrived\ Documents\}|}{|\{Retrived\ Documents\}|} \tag{1}$$

Which can also be written as the following.

$$Precision = \frac{\sum True\ Positive}{\sum True\ Positive + \sum False\ Positive} \tag{2}$$

Plugging in our values, we get the answer.

$$Precision = \frac{10tp}{10tp + 10fp} = \frac{10}{20} = 0.5 \tag{3}$$

Problem 4 continued on next page. . .

Again, according to the provided Wikipedia page, the equation for accuracy can be found with the following equation.

$$Accuracy = \frac{\sum True\ Positive + \sum True\ Negative}{\sum True\ Positive + \sum True\ Negative + \sum False\ Positive + \sum False\ Negative} \tag{4}$$

By plugging in our values from the spam classifier, we get the following results.

$$Accuracy = \frac{10tp + \sum 0tn}{\sum 10tp + \sum 0tn + \sum 10fp + \sum 0fn} = \frac{10}{20} = 0.5 \tag{5}$$

Thus, our precision is 0.5 and our accuracy is also 0.5. This is because half of our positive results were correct, making precision 0.5, and half were the correct value while the other half were not, making accuracy 0.5.