HW9 - Email Classification

Logan Bartels
December 13, 2020

Q1

Answer

The topic I've chosen to classify emails on is shopping. To be more specific, I define shopping emails as emails with discounts from businesses I regularly do business with. For this assignment, those business include Best Buy, Old Navy, and Amazon. I must state that I only have 1 email from Amazon in my training set, as it was the only Amazon email I could extract the plain-text from. I must also state that I used my personal email for shopping emails, and my ODU email for "not shopping" emails. For all of the emails I consider to be shopping emails (including test emails), I extracted the text using JusText. I also used JusText to extract the first 5 training "not shopping" emails. For the rest of the "not shopping" emails, I just copy-pasted the text from emails from my ODU account into text files. I did this because JusText was inserting non-plaintext elements into the output file. That's why when you look in the repository you will only see 30 *.eml files instead of 50.

As for the naming convention of the text files, each file is named to represent their purpose. All training emails are saved with "train" as the prefix, followed by "On" or "Off" to represent whether they are on or off topic (shopping or not shopping). Each training email is concluded with a number to keep track of the files. So a training email that is on topic would be saved as "trainOn1.txt." All testing emails have the prefix "test," and the rest of the name shares the same conventions as the training files. So a test email that is not on topic would be saved as "testOff1.txt."

Note:

Questions 2 and 3 were completed using a copy of the week 13 Colab notebook. The notebook can be found in the repository, and here: `https://colab.research.google.com/drive/16ht-67nTMMN8m4pPBI3v6icdzbGAL2pM?usp=sharing`

Modifications to the notebook are documented below.

Q2

Answer

```python
# Read in files for training
# emails that are shopping
file = open("trainOn1.txt", "r")
trainOn1 = file.read()
file = open("trainOn2.txt", "r")
trainOn2 = file.read()
file = open("trainOn3.txt", "r")
trainOn3 = file.read()
file = open("trainOn4.txt", "r")
trainOn4 = file.read()
file = open("trainOn5.txt", "r")
trainOn5 = file.read()
file = open("trainOn6.txt", "r")
trainOn6 = file.read()
file = open("trainOn7.txt", "r")
trainOn7 = file.read()
file = open("trainOn8.txt", "r")
trainOn8 = file.read()
file = open("trainOn9.txt", "r")
trainOn9 = file.read()
file = open("trainOn10.txt", "r")
trainOn10 = file.read()
file = open("trainOn11.txt", "r")
trainOn11 = file.read()
file = open("trainOn12.txt", "r")
trainOn12 = file.read()
file = open("trainOn13.txt", "r")
trainOn13 = file.read()
file = open("trainOn14.txt", "r")
trainOn14 = file.read()
file = open("trainOn15.txt", "r")
trainOn15 = file.read()
file = open("trainOn16.txt", "r")
trainOn16 = file.read()
file = open("trainOn17.txt", "r")
trainOn17 = file.read()
file = open("trainOn18.txt", "r")
trainOn18 = file.read()
file = open("trainOn19.txt", "r")
trainOn19 = file.read()
file = open("trainOn20.txt", "r")
trainOn20 = file.read()
```

```
# emails that are not shopping
file = open("trainOff1.txt" , "r")
trainOff1 = file.read()
file = open("trainOff2.txt" , "r")
trainOff2 = file.read()
file = open("trainOff3.txt" , "r")
trainOff3 = file.read()
file = open("trainOff4.txt" , "r")
trainOff4 = file.read()
file = open("trainOff5.txt" , "r")
trainOff5 = file.read()
file = open("trainOff6.txt" , "r")
trainOff6 = file.read()
file = open("trainOff7.txt" , "r")
trainOff7 = file.read()
file = open("trainOff8.txt" , "r")
trainOff8 = file.read()
file = open("trainOff9.txt" , "r")
trainOff9 = file.read()
file = open("trainOff10.txt" , "r")
trainOff10 = file.read()
file = open("trainOff11.txt" , "r")
trainOff11 = file.read()
file = open("trainOff12.txt" , "r")
trainOff12 = file.read()
file = open("trainOff13.txt" , "r")
trainOff13 = file.read()
file = open("trainOff14.txt" , "r")
trainOff14 = file.read()
file = open("trainOff15.txt" , "r")
trainOff15 = file.read()
file = open("trainOff16.txt" , "r")
trainOff16 = file.read()
file = open("trainOff17.txt" , "r")
trainOff17 = file.read()
file = open("trainOff18.txt" , "r")
trainOff18 = file.read()
file = open("trainOff19.txt" , "r")
trainOff19 = file.read()
file = open("trainOff20.txt" , "r")
trainOff20 = file.read()
# Read in files for testing
# Test emails that should be shopping
file = open("testOn1.txt", "r")
testOn1 = file.read()
file = open("testOn2.txt", "r")
testOn2 = file.read()
```

```python
file = open("testOn3.txt", "r")
testOn3 = file.read()
file = open("testOn4.txt", "r")
testOn4 = file.read()
file = open("testOn5.txt", "r")
testOn5 = file.read()
# Test emails that should not be shopping
file = open("testOff1.txt", "r")
testOff1 = file.read()
file = open("testOff2.txt", "r")
testOff2 = file.read()
file = open("testOff3.txt", "r")
testOff3 = file.read()
file = open("testOff4.txt", "r")
testOff4 = file.read()
file = open("testOff5.txt", "r")
testOff5 = file.read()
```

Listing 1: A single cell in the notebook dedicated to reading in the text-file version of emails.

```python
def spamTrain(cl):
  # emails that are shopping
  cl.train(trainOn1, 'shopping')
  cl.train(trainOn2, 'shopping')
  cl.train(trainOn3, 'shopping')
  cl.train(trainOn4, 'shopping')
  cl.train(trainOn5, 'shopping')
  cl.train(trainOn6, 'shopping')
  cl.train(trainOn7, 'shopping')
  cl.train(trainOn8, 'shopping')
  cl.train(trainOn9, 'shopping')
  cl.train(trainOn10, 'shopping')
  cl.train(trainOn11, 'shopping')
  cl.train(trainOn12, 'shopping')
  cl.train(trainOn13, 'shopping')
  cl.train(trainOn14, 'shopping')
  cl.train(trainOn15, 'shopping')
  cl.train(trainOn16, 'shopping')
  cl.train(trainOn17, 'shopping')
  cl.train(trainOn18, 'shopping')
  cl.train(trainOn19, 'shopping')
  cl.train(trainOn20, 'shopping')
  # emails that are not shopping
  cl.train(trainOff1, 'not shopping')
  cl.train(trainOff2, 'not shopping')
  cl.train(trainOff3, 'not shopping')
  cl.train(trainOff4, 'not shopping')
  cl.train(trainOff5, 'not shopping')
```

```
    cl.train(trainOff6, 'not shopping')
    cl.train(trainOff7, 'not shopping')
    cl.train(trainOff8, 'not shopping')
    cl.train(trainOff9, 'not shopping')
    cl.train(trainOff10, 'not shopping')
    cl.train(trainOff11, 'not shopping')
    cl.train(trainOff12, 'not shopping')
    cl.train(trainOff13, 'not shopping')
    cl.train(trainOff14, 'not shopping')
    cl.train(trainOff15, 'not shopping')
    cl.train(trainOff16, 'not shopping')
    cl.train(trainOff17, 'not shopping')
    cl.train(trainOff18, 'not shopping')
    cl.train(trainOff19, 'not shopping')
    cl.train(trainOff20, 'not shopping')
```

Listing 2: Modified spamTrain function to train the classifier.

```
# HW9 test
cl = naivebayes(getwords)
cl.setdb('test.db')
spamTrain(cl)
```

Listing 3: Code cell to call the classifier set the database and call spamTrain

Note: Classification results were obtained by copy-pasting the following cell in the notebook into other cells and changing the variable name passed in as needed. This can be seen in the cells beneath the cell shown in listing 3.

```
cl.classify(testOn1, default='unknown')
```

Listing 4: Code to classify test variable.

Table 1: Classification results.

Test file	Classification
testOn1.txt	shopping
testOn2.txt	shopping
testOn3.txt	shopping
testOn4.txt	shopping
testOn5.txt	shopping
testOff1.txt	not shopping
testOff2.txt	not shopping
testOff3.txt	not shopping
testOff4.txt	not shopping
testOff5.txt	shopping

Discussion

I should start with stating that I used the Bayesian classifier with SQL (the classifier class was passed into the naivebayes class) to obtain my classifications. The code in listing 1 reads in each of the text files an stores their text in variables named after the file. I know that there are better, more concise methods of doing this, but in the interest of time, I opted for the brute-force-copy-paste approach. The code in listing 2 is a modified version of the example notebook's spamTrain function. It takes each variable that a file's text was saved in and uses it to train the classifier. Variables named "trainOnx" are classified as shopping, and files named "trainOffx" are classified as "not shopping." The code in listing 3 calls the naivebays classifier and sets the database to "test.db." Then it calls the spamtrain function to feed in the training files/variables. After that, I simply call the classify function for each test variable/file, and set the default classification to unknown. This is shown in listing 4. Most of the classifications were correct, the keyword being "most." I will elaborate more in question 3.

Q3

Answer

Table 2: Confusion Matrix

		Actual	
		Shopping	Not Shopping
Predicted	Shopping	5	1
	Not Shopping	0	4

```
Greetings fellow Monarchs,

Our next College of Sciences Lunch and Learn will take place on Tuesday, November 17th from 12:20 - 1:20pm and feature Lynn Clements, former Executive Director of the Virginia Aquarium and Marine Science Center, and Greg Bockheim, Executive Director of the Virginia Zoo.  You will learn how they got into the field of animal conservation, the challenges and rewards of working with animals, and how you can enter a career at an accredited zoo or aquarium. Please join us.

Join Zoom Meeting
https://odu.zoom.us/j/94632175154?pwd=dzl6bnlPeU5HUGdJQ3ZrWFNmN3lidz09

Meeting ID: 946 3217 5154
Passcode: 805735
```

Listing 5: Text of testOff5.txt

Discussion

As you can see in table 2, I had only 1 false positive classification. According to table 1, that false positive was testOff5.txt. The text for that file is shown in listing 5. I am perplexed as to why this email was classified as a shopping email. If anything, this goes to show that classification algorithms are not perfect. I would bet money on the average person getting an email in their inbox where it should have gone somewhere else at least once in their lifetime.

Extra Credit

Q4

Answer

$$Precision = \frac{\sum TruePositive}{\sum TestOutcomePositive} \tag{1}$$

$$Precision = \frac{5}{6} \tag{2}$$

$$Precision = 0.833 \tag{3}$$

$$Accuracy = \frac{\sum TruePositive + \sum TrueNegative}{\sum TotalPopulation} \tag{4}$$

$$Accuracy = \frac{5+4}{10} \tag{5}$$

$$Accuracy = 0.9 \tag{6}$$

Discussion

The formulas in equations 1 and 4 were taken from slide 43 of the Week 13 slides. For the precision, the confusion matrix shown in figure 2 shows 5 true positives and 1 false positive. It follows that the numerator of the fraction is 5 and the denominator is 6 (5 true positives plus 1 false positive). The final value for precision is 0.833.

For accuracy the confusion matrix shows 5 true positives and 4 true negatives. We know from the assignment that the total test population is 10. It follows that the numerator is the sum of 5 and 4, and the denominator is 10. The final value for accuracy is 0.9. Both values were calculated mentally and checked by the Windows 10 calculator application.

Q5

Answer

Table 3: Updated Confusion Matrix

		Actual	
		Shopping	Not Shopping
Predicted	Shopping	5	0
	Not Shopping	0	5

Discussion

For this question, I improved precision and accuracy by turning the 1 false positive email (indicated in table 2) into a true negative. I accomplished this by going to the classifier class, and adjusting the weight and assumed probability (ap) in the weightedprob function. I started by lowering the weight to 0.5. Then I proceeded to lower the assumed probability by increments of ten percent (0.1) after every unsuccessful attempt to convert my false positive to a true negative. I also re-ran the classifications for all other files to make sure nothing else changed that I didn't want to change. I should note that each time I updated the weightedprob function, I re-ran the classifier class, the naivebayes class, and (because I used the classifier with SQL) deleted the database file for retraining. I continued to lower the assumed probability by ten percent each time I classified every email to no avail, until I lowered it to 0.2. When I classified each email with an assumed probability of 0.2 my false positive turned into an "unknown." This was progress. I lowered the assumed probability once more to 0.1. This is when my false positive finally turned into a true negative (not shopping). The updated confusion matrix is shown in table 3. Every other email's classification was unchanged.

References

- Week 13 slides, `https://docs.google.com/presentation/d/1TgSeYh7gjpxl8f_9-FKG7MnXG1HI_iwH_KDFyUgKlWU/edit?usp=sharing`
- Week 13 Colab notebook, `https://colab.research.google.com/github/cs432-websci-fall20/assignments/blob/master/432_PCI_Ch06.ipynb`
- JusText repository, `https://github.com/miso-belica/jusText`