***MAT 345: Assignment 2***

# Spam Filter Report

## By: Bradley Esparza, Rahil Momin

In this report, we used a dataset(A) from http://spamassassin.apache.org/old/publiccorpus/ to write a spam filter.

This entire report was done in Python, using Juptyer Notebook as the editor.

For our spam filter, we decided to remove any non-unicode character as we wanted to only deal with utf-8 format.

We chose to label spam if the probability exceeded or was equal to 0.50.

Top 5 "spammiest" words:

1. your
2. the
3. for
4. you
5. a

Top 5 "hammiest" words:

1. re
2. the
3. for
4. of
5. to

Accuracy: 0.88

Precision: 0.55

Recall: 0.75

Conclusion: Our spam filter works! It is fairly accurate for predicting Ham, though pretty bad when it comes to spam. With a precision of .55, we should definitely look into improving how we can correctly predict more spam. Currently we do not exclude stop words, so an improvement would be to remove these. We could also look to messing with the smoothing variables.

## Spam Filter Code:

In [1]:
```
import os
import re
import numpy as np
import string
import re
import operator
import math
```

In [2]:
```
def parse_folder_to_list(folder):
    list_of_subjects = []
    for filename in os.listdir(folder):
        with open(os.path.join(os.path.dirname(folder), filename), 'rb',
0) as search:
            for line in search:
                line = line.rstrip()
                if re.search(br'Subject:', line):
                    try:
                        list_of_subjects.append(line[8:].decode().transl
ate(str.maketrans('', '', string.punctuation)).lower())
                    except Exception:
                        pass
                        #print("File: ", filename, " --- Failed to decod
e: ", line)
                pass
        pass
    return list_of_subjects
```

In [3]:
```
def CreateArrayFromEmail(email, dictionary):
    dictionary2 = dictionary.fromkeys(dictionary, 0)

    #get tokens
    words = re.findall(r"[\w']+", email)

    #set word to 1 if in subject
    for word in words:
        if word in dictionary.keys():
            dictionary2[word] = 1
        pass
    return dictionary2
```

In [4]:
```
#Createa a function that caluclates the prob of being spam
def prob_of_word_given_spam(word, spamDic, totalSpam):
    try:
        value = spamDic[word]
    except Exception:
        value = 0
    return (value + alpha)/ (totalSpam + beta)

def prob_of_word_given_ham(word, hamDic, totalHam):
    try:
        value = hamDic[word]
    except Exception:
        value = 0
    return (value + alpha)/ (totalHam + beta)
```

In [5]:
```
def TestForSpam(emailArray, y, y_0, z, z_0, probSpam, probHam):
    num = math.exp(np.dot(emailArray, y) + y_0) * probSpam
    denom = num + (math.exp(np.dot(emailArray, z) + z_0) * probHam)
    return num / denom
```

Our first step is to take the downloaded data parse out all the subject headers.

1. First we loop through each raw email file.
2. Then we check if the raw email data contains "Subject:"
3. We then convert the data to 'utf-8' string format, removing any non-complient headers.
4. We also remove all punctuation and set each string to lower-case for easy comparisons.

In [6]:
```
spam = parse_folder_to_list('./spam/')

easy_ham = parse_folder_to_list('./easy_ham/')
hard_ham = parse_folder_to_list('./hard_ham/')
ham = easy_ham + hard_ham
```

Next, we split our data into 75% Training and 25% Testing. This will be used to train and test our spam filter.

In [7]:
```
from sklearn.model_selection import train_test_split
spam_train, spam_test = train_test_split(spam, test_size=0.25)
ham_train, ham_test = train_test_split(ham, test_size=0.25)
print("Spam/Ham Training data size:", len(spam_train), "/", len(ham_trai
n))
print("Spam/Ham Training data size:", len(spam_test), "/", len(ham_test
))
```

```
Spam/Ham Training data size: 373 / 2268
Spam/Ham Training data size: 125 / 756
```

Now we take the Training data and create a list of unique words to be used as a dictionary.

In [8]:
```
spam_dic = {}
ham_dic = {}

totalSpam = 0.0
totalHam = 0.0

for subjectHeader in spam_train:
    for x in subjectHeader.split():
        try:
            spam_dic[x] += 1
        except Exception:
            spam_dic[x] = 1
        pass
        totalSpam += 1
    pass

for subjectHeader in ham_train:
    for x in subjectHeader.split():
        try:
            ham_dic[x] += 1
        except Exception:
            ham_dic[x] = 1
        pass
        totalHam += 1
    pass

total_dic = dict(spam_dic);
total_dic.update(ham_dic)

totalWords = totalHam + totalSpam

print("Total unique words: ", len(total_dic))
print("Total Spam Words: ", totalSpam)
print("Total Ham Words: ", totalHam)
print("Total Words: ", totalWords)
```

```
Total unique words:  3951
Total Spam Words:  2558.0
Total Ham Words:  12831.0
Total Words:  15389.0
```

Now we may begin working on our Spam Filter!

In [9]:
```
#lets define our constants:
prob_spam = totalSpam / totalWords
prob_ham = totalHam / totalWords
alpha = 1
beta = 2
```

In [10]:
```
#more constants

y = []
y_0 = 0.0
z = []
z_0 = 0.0

spammiest = {}
hammiest = {}

for word in total_dic:
    #compute y
    Pks = prob_of_word_given_spam(word, spam_dic, totalSpam)
    y.append(math.log(Pks / (1 - Pks)))

    #compute y_0
    y_0 += math.log(1 - Pks)

    #compute z
    Pkh = prob_of_word_given_ham(word, ham_dic, totalHam)
    z.append(math.log(Pkh / (1 - Pkh)))

    #compute z_0
    z_0 += math.log(1 - Pkh)


    #compute spammiest and hammiest
    spammiest[word] = Pks
    hammiest[word] = Pkh
    pass

sortedSpam = sorted(spammiest.items(), key=operator.itemgetter(1))
sortedSpam.reverse()
print("Spammiest: ", sortedSpam[:5])
sortedHam = sorted(hammiest.items(), key=operator.itemgetter(1))
sortedHam.reverse()
print("Hammiest: ", sortedHam[:5])
```

```
Spammiest:  [('the', 0.0296875), ('of', 0.0234375), ('your', 0.02304687
5), ('for', 0.017578125), ('a', 0.01640625)]
Hammiest:  [('re', 0.08782046286916544), ('the', 0.02781890438712694),
('for', 0.0190914049715577), ('to', 0.014260110652224734), ('of', 0.013
402945531052755)]
```

In [11]:
```
correctSpam = 0.0

for everyEmail in spam_test:
    # Generate the binary array [value is in dictionary 1, else 0]
    binArray = np.array(list(CreateArrayFromEmail(everyEmail, total_dic)
.values()))

    # Test for spam
    result = TestForSpam(binArray, y, y_0, z, z_0, prob_spam, prob_ham)

    if(result >= 0.5):
        correctSpam += 1
    pass

correctHam = 0.0

for everyEmail in ham_test:
    # Generate the binary array [value is in dictionary 1, else 0]
    binArray = np.array(list(CreateArrayFromEmail(everyEmail, total_dic)
.values()))

    # Test for spam
    result = TestForSpam(binArray, y, y_0, z, z_0, prob_spam, prob_ham)

    if(result < 0.5):
        correctHam += 1
    pass

print("Correctly predicted Spam: ", correctSpam)
print("Correctly predicted Ham: ", correctHam)
print("Falsely predicted Spam: ", len(ham_test) - correctHam)
print("Falsely predicted Ham: ", len(spam_test) - correctSpam)

totalTestedEmails = len(ham_test) + len(spam_test)
print("Total Emails Tested: ", totalTestedEmails)
```

```
Correctly predicted Spam:  96.0
Correctly predicted Ham:  679.0
Falsely predicted Spam:  77.0
Falsely predicted Ham:  29.0
Total Emails Tested:  881
```

In [12]:
```
Accuracy = (correctSpam + correctHam) / totalTestedEmails
Precision = correctSpam / ((len(ham_test) - correctHam) + correctSpam)
Recall = correctSpam / len(spam_test)
print("Accuracy: ", Accuracy)
print("Precision: ", Precision)
print("Recall: ", Recall)
```

```
Accuracy:  0.8796821793416572
Precision:  0.5549132947976878
Recall:  0.768
```