Email Spam Detection And Filtering : An Application of Natural Language Processing

Nihit Parikh
16BCP027
B.Tech. (Computer Engineering) student at
Pandit Deendayal Petroleum University, India.

Parth Shah
16BCP028
B.Tech. (Computer Engineering) student at
Pandit Deendayal Petroleum University, India.

1. Introduction

Text mining (deriving information from text) is a wide field which has gained popularity with the huge text data being generated. Automation of a number of applications like sentiment analysis, document classification, topic classification, text summarization, machine translation, etc has been done using machine learning models. Spam filtering is a beginner's example of document classification task which involves classifying an email as spam or non-spam (a.k.a. ham) mail. Spam box in your Gmail account is the best example of this. With ever increasing cybercrime and bullying, emails are one of the easiest way to target innocent people over the internet to scam. This will help users to differentiate from genuine emails which they need v/s spam mails which includes unwanted offers from many corporates and scams. This will also help to keep the inbox clutter free. Spams are used daily by con artist all over the world to scam innocent people over the internet. Spams are also sent to people against their wishes, and the inbox gets cluttered with unwanted offered from many corporates. The purposes of spam and spam filters are diametrically opposed: spam is effective if it evades filters, while a filter is effective if it recognizes spam.

2. Dataset Description

The data is available on the following website:
http://www2.aueb.gr/users/ion/data/enron-spam/

3. Email spam detection & filtering approach

3.1 Prepare the Data:

All the email data is contained in the data folder. This email dataset contains 6 folders. The six folders are further subdivided into two separate subsets — ham and spam emails. Ham indicates the email is genuine and not a spam email.

Total number of ham emails - **16545**

Total number of spam emails - **17171**

The training and testing set is randomly divided into **60:40** respectively. (using K-fold cross validation)

Pre-processing:

Text cleaning is the first step where we remove those words from the document which may not contribute to the information we want to extract. Emails may contain a lot of undesirable characters like punctuation marks, stop words, digits, etc which may not be helpful in detecting the spam email.

a) Removal of stop words – Stop words like “and”, “the”, “of”, etc are very common in all English sentences and are not very meaningful in deciding spam or legitimate status, so these words have been removed from the emails.

b) Lemmatization – It is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. For example, “include”, “includes,” and

“included” would all be represented as “include”. The context of the sentence is also preserved in lemmatization as opposed to stemming

3.2 Generate Dictionary (Vocabulary):

After we got the data prepared we can start creating the dictionary where we are gonna choose the features (words in this case) based on which the algorithm will later decide if given email message is spam or non-spam.

First thing we need to do is to create the dictionary of words that will be used for our model. The dictionary contains words and a boolean value True as Naive bayes expects the input in a particular format

{Word1: True, Word2: True, Word3: True}

as key and the value will be either ham or spam depending from which set it has been taken from.

Code:

```
def createWordFeatures(words):
    my_dict = dict( [ (word, True) for word in words] )
    return my_dict
hamList = []
spamList = []
for directories, subdirs, files in os.walk(data_directory):
    if (os.path.split(directories)[1]  == 'ham'):
        for fileName in files:
            with open(os.path.join(directories, fileName), encoding="latin-1") as f:
```

```
                message = f.read()
                wordList = word_tokenize(message)
                hamList.append((createWordFeatures(wordList), "ham"))
        if (os.path.split(directories)[1]  == 'spam'):
            for fileName in files:
                with open(os.path.join(directories, fileName), encoding="latin-1") as f:
                    message = f.read()
                    wordList = word_tokenize(message)
                    spamList.append((createWordFeatures(wordList), "spam"))
    print(hamList[0])
    print(spamList[0])
```

After the dictionary is created, it looks like this.
({'Subject': True, ':': True, 'christmas': True, 'tree': True, 'farm': True, 'pictures': True}, 'ham')

3.3 Generate ML Model:

We have our data structured and prepared for running it through Naive Bayes algorithm so we can get the prediction model. The Naive Bayes is also called “probabilistic classifier” since it is based on calculating the probability that one item (email in our case) belongs to a particular class (classification).

Its formula is pretty simple. It counts the number of occurrences of a w (in our case one word from the dictionary) in all the c (sum of all occurrences of the dictionary words in either spam or non spam emails depending for which one we are estimating the

probability). V — is the number of words in the dictionary. This probability will be calculated separately first on spam and then on non spam emails.

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c) \right) + |V|}$$

Figure: Naive Bayes using laplacian smoothing

Implementation of above formula:

```
trainingPart = int(len(mergedList) * .6)
trainingData = mergedList[:trainingPart]
testData =  mergedList[trainingPart:]

print("Total Number of files:    {}".format(len(mergedList)))
print("Number of Training files: {}".format(len(trainingData)))
print("Number of Test files:     {}".format(len(testData)))

nb_classifier = NaiveBayesClassifier.train(trainingData)
print("Trained the Naive Bayes classifier in {:,.2f} seconds".format(time.time()-start_time))
accuracy = nltk.classify.util.accuracy(nb_classifier, testData)
print("Accuracy is: ", accuracy * 100)
```

3.4 Test the ML Model:

The below mentioned are the most interesting features:

nb_classifier.show_most_informative_features(20)

Most Informative Features

enron = True	ham : spam	=	3115.4 : 1.0
daren = True	ham : spam	=	447.4 : 1.0
php = True	spam : ham	=	363.0 : 1.0
hpl = True	ham : spam	=	305.3 : 1.0
ect = True	ham : spam	=	219.8 : 1.0
crenshaw = True	ham : spam	=	217.0 : 1.0
corel = True	spam : ham	=	200.7 : 1.0
713 = True	ham : spam	=	188.3 : 1.0
scheduling = True	ham : spam	=	181.1 : 1.0
eol = True	ham : spam	=	179.0 : 1.0
louise = True	ham : spam	=	177.8 : 1.0
parsing = True	ham : spam	=	172.0 : 1.0
sex = True	spam : ham	=	158.7 : 1.0
schedules = True	ham : spam	=	158.3 : 1.0
= True	ham : spam	=	154.2 : 1.0
xp = True	spam : ham	=	146.7 : 1.0
origination = True	ham : spam	=	130.1 : 1.0
medication = True	spam : ham	=	122.7 : 1.0
prescription = True	spam : ham	=	104.7 : 1.0
omit = True	spam : ham	=	104.0 : 1.0

3.4.1 Predicting New Emails:

classifying the below messages as spam or ham.

Break the message into words using *word_tokenzise*

Generate features using *createWordFeatures*

Use the *classify* function

Example of Nigerian Scam email

msg1 = '''Sir, we are honourably seeking your assistance in the following ways.

1) To provide a Bank account where this money would be transferred to.

2) To serve as the guardian of this since I am a girl of 26 years.

Moreover Sir, we are willing to offer you 15% of the sum as compensation for effort input after the successful transfer of this fund to your designate account overseas. please feel free to contact ,me via this email address

wumi1000abdul@yahoo.com Anticipating to hear from you soon.Thanks and God Bless.'"

Note from Professor regarding Python + Tableau for plotting flight pattern

msg2 = '"I've mentioned that Python is widely used for data pre-processing tasks.

Here's a really nice use of Python for preparing flight data for plotting in Tableau.

The same principles can be used for plotting all kinds of geographic related "routes".

Plus, you learn about KML files and working with spatial data.'"

words = word_tokenize(msg1)

features = createWordFeatures(words)

print("Message 1 is" ,nb_classifier.classify(features))

Message 1 is spam

words = word_tokenize(msg2)

features = createWordFeatures(words)

print("Message 2 is" ,nb_classifier.classify(features))

Message 2 is ham.

4. Conclusion

(1) The **accuracy** achieved by our system is **97.6923%.**

(2)

	Spam	Ham
Spam	6680 (TP)	137 (FN)
Ham	225 (FP)	6445 (TN)

Here, TP = True Positive, FN = False Negative, FP = False Positive, TN = True Negative

Testing set contains 13487 instances (emails).

Precision = TP/(TP + FP) = 6680/(6680+225) = 0.967 (aka 96%)

Recall = TP/(TP + FN) = 6680/(6680+137) = 0.9799 (aka 98%)

(3) When we checked the accuracy of the system using 5-fold cross validation (keeping 80% data for training and using 20% data for testing), we achieved an accuracy around **98.5%** for 5 total test-cases.

```
In [25]: X_train, X_test, y_train, y_test = train_test_split(features_matrix, labels, test_size=0.20)
         model = MultinomialNB()
         model.fit(X_train,y_train)
         result = model.predict(X_test)
         print("Accuracy is 1:",result)

         Accuracy is 1:  98.53682649530401
```

5. Future Work

The current trained machine give output as ‘spam’, if a sentence containing totally new words (not a single word belonging to the whole training dataset) is inputted in the testing phase. Thus, the team envisions to solve that challenge.

6. References

[1] https://medium.com/analytics-vidhya/building-a-spam-filter-from-scratch-using-machine-learning-fc58b178ea56

[2] https://appliedmachinelearning.blog/2017/01/23/email-spam-filter-python-scikit-learn/