Charles Vanleuvan

IST 664 Final Project

NLP Investigation: Spam Email Classification

Background

In this NLP investigation, I will be diving into the email dataset from the Enron public email corpus. The purpose of this investigation is to tune a NLP model and feature extraction program to see the varying accuracies (and other evaluation metrics) that can be achieved. This dataset is provided to us in the form of a zipped folder, which contains two more nested folders, labeled 'ham' and 'spam'. The 'ham' email labels indicate a good, valid email, while the 'spam' label is just that, spam. In this case, spam is referring to the uselessness or irrelevant contents of the email (such as an unwanted pharmaceutical advertisement, or an advertisement in another language, or contained nonsensical language). There are 3,672 emails in the 'ham' folder, and 1,500 emails in the 'spam' folder. The feature extraction python program will process a number of emails from each collection, then extract designed features from each email, and then run a Naïve Bayes classification algorithm from NLTK to view how accurate the model is at predicting a spam or ham label.

```
Subject: meter 1431 - nov 1999
aimee ,
sitara deal 92943 for meter 1431 has expired on oct 31 , 1999 . settlements
is unable to draft an invoice for this deal . this deal either needs to be
extended or a new deal needs to be set up . please let me know when this is
resolved . we need it resolved by friday , dec 17 .
hc
```

Figure 1: An example of a ham email

```
Subject: your prescription is ready . . oxwq s f e
low cost prescription medications
soma , ultram , adipex , vicodin many more
prescribed online and shipped
overnight to your door ! !
one of our us licensed physicians will write an
fda approved prescription for you and ship your
order overnight via a us licensed pharmacy direct
to your doorstep . . . . fast and secure ! !
click here !
no thanks , please take me off your list
ogrg z
lqlokeolnq
lnu
```

Figure 2: An example of a spam email

Document Collection

The program developed to process the email data set was created in python, then executed via a command line prompt, where the only input needed from the user was a suggestion of how many emails to test with (this value was variable, but was kept to 1000 emails for this investigation for time conservation). When broken down, this python file has 7 key steps:

1. Identify the number of emails to collect from the spam and ham collections
2. Loop through the spam and ham collections and read in each line of text in each email and append to a spam list or a ham list. This list is a collection of documents.
3. Tokenize the words in each document in the two collections and add a label of 'ham' or 'spam', corresponding to the collection each one is in.
4. Randomly shuffle the documents so that there is not order to the collection of labels.
5. Create a word list from both collections of documents that will be used for the next steps of feature extraction. Commonly known as a Unigram feature set, or Bag of Words feature set.
6. Use a custom defined feature extraction function to prepare feature sets (a python dictionary) for each text, using the above word list.
7. Run 10-fold cross validation, where the training and test documents are randomized in each run of the Naïve bayes model. At the end of the k-fold cross validation, the evaluation metrics are displayed in the terminal.

Feature Extraction

A crucial component of this investigation was the development of 5 feature extraction functions, along with 2 separate functions tasked with calculating evaluation measures and cross validation accuracy. In the NLTK version of a Naïve Bayes algorithm, the feature set must be supplied as a python tuple, where the first element is a feature set dictionary, and the second element is the classification. All feature extraction functions took two arguments to the function call, a document, and a set of words. The term frequency – inverse document frequency functions were slightly different, and that will be explained later in this investigation. Each function also looped over the collection of tokenized texts to create the feature dictionary. The 5 feature extraction functions are:

1. Unigram features (aka Bag of Word features)
    a. This function took a set of words from the document and looped through each word. The word was cross referenced with the words in the top 1,500 most frequent words in all the texts, and a Boolean True or False was inserted into the unigram feature dictionary as the value for that word's key. The result of this function is a dictionary with keywords as the keys, and a True or False Boolean as the value.
2. Bigram features
    a. This process is very similar to the unigram feature extraction. The slight difference is that the bigram function from NLTK is utilized here to create keys in the features dictionary that represent a Boolean True or False if that Bigram is included in the document.
3. Part-Of-Speech features
    a. This function runs the NLTK part of speech tagging function on the tokenized text in the email document, and then calculates a cumulative sum of each tagged part of speech in

the text. The result for each document is the total number of tags for nouns, verbs, etc. that are in that document.

4. Punctuation frequency features
    a. This function loops through each word in the top 1,500 most common words from all documents and calculates a frequency of that word in each document. The resulting dictionary has keywords as keys and the frequency value in each document as the dictionary value.
5. Term frequency – Inverse Document Frequency features
    a. Term frequency feature extractor
        i. This function loops through each document, then calculates the number of words in that document. Then for each unique word in the 1,500 most common words, the frequency is calculated by dividing the count by the total number of words.
    b. Inverse Document Frequency extractor
        i. This function finds the number of documents in total (2,000 since 1,000 ham and 1,000 spam emails are selected when this program runs), and then loops through the 1,500 most common words and records in a dictionary which documents contain that word. The cumulative sum for each word is calculated as well (i.e., how many documents that contain word *w*). Then the log of the total number of documents / number of documents containing word *w* is calculated and stored in a dictionary for each unique word as the key. A dictionary of the inverse document frequencies is returned
    c. Combining Term Frequency and Inverse Document Frequency
        i. Finally, a function combines the term frequency and the inverse document frequency. This simply is the multiplication of the TF by IDF for each keyword, resulting in another dictionary that will finally be used as the feature set in the NLTK naïve Bayes classifier.
6. Cross Validation Accuracy
    a. This function takes a user input for the number of folds to run in cross validation, as well as the feature set being used in this round of testing. A subset size is then calculated from the number of folds value entered by the user. This feature set is then split into a training and test data set based on the subset size. Next, the naïve bayes classifier is run on the training set. A gold list and a predicted list is created from the test data set and running the classifier on the test set, respectively. Then, the accuracy for this round is calculated by evaluating the predicted list correct answers against the gold list true answers. Each round's accuracy is stored in a list. This process is repeated as many times as the user desires, and when it is complete, the output is an average accuracy value from all the rounds' testing.
7. Evaluation Measures
    a. The evaluation measures of precision, recall, and F score were combined into one function that would print the scores for each round of testing in k-fold cross validation. Using the gold list and predicted list from the cross-validation function, the number of True Positives, True Negatives, False Positives, and False Negatives can be calculated. Then, the calculation of recall, precision, and F measure are quite simple.

Experiments

In the experiment phase of this investigation, the Unigram, bigram, and Part of Speech tagger feature sets were compared with and without stop word filtering. The bigram feature set was also combined with the bigram feature set. A new feature that was decided to investigate was the punctuation frequency within the email, entirely omitting any alphanumerical characters. Lastly, the advanced task chosen was the term frequency- Inverse document frequency of common words, both with and without stop word filtering.

With the Unigram feature set, no stop word filtering, and 10-fold cross validation, the resulting average accuracy was 93.95%.

```
Number of spam files: 1000
Number of ham files: 1000
Each fold size:  200
Accuracies:
0 0.945
        Precision       Recall          F
ham           0.884      1.000      0.939
spam          1.000      0.905      0.950
1 0.94
        Precision       Recall          F
ham           0.880      1.000      0.936
spam          1.000      0.893      0.943
2 0.92
        Precision       Recall          F
ham           0.832      1.000      0.908
spam          1.000      0.868      0.929
3 0.935
        Precision       Recall          F
ham           0.863      1.000      0.927
spam          1.000      0.890      0.942
4 0.94
        Precision       Recall          F
ham           0.876      1.000      0.934
spam          1.000      0.896      0.945
5 0.95
        Precision       Recall          F
ham           0.910      1.000      0.953
spam          1.000      0.899      0.947
6 0.93
        Precision       Recall          F
ham           0.867      1.000      0.929
spam          1.000      0.872      0.931
7 0.955
        Precision       Recall          F
ham           0.913      1.000      0.955
spam          1.000      0.914      0.955
8 0.92
        Precision       Recall          F
ham           0.846      1.000      0.917
spam          1.000      0.857      0.923
9 0.96
        Precision       Recall          F
ham           0.926      0.989      0.956
spam          0.991      0.938      0.963
Average Accuracy from k-fold CV:  0.9395
```

Figure 3: Unigram feature set. No stop word filtering

When filtering out English stop words using NLTK's corpus, the average accuracy improves around 1% when using the unigram feature set, to 94.9%.

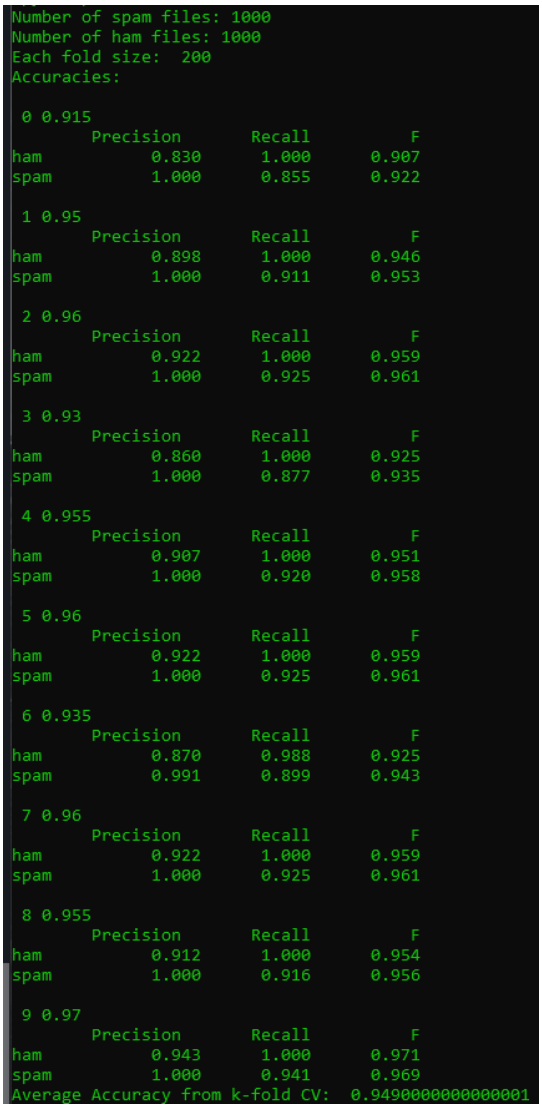


Figure 4: Unigram feature set with stop word filtering

With the basic bigram feature set, the average accuracy after 10 rounds of CV is 93.85% accuracy.

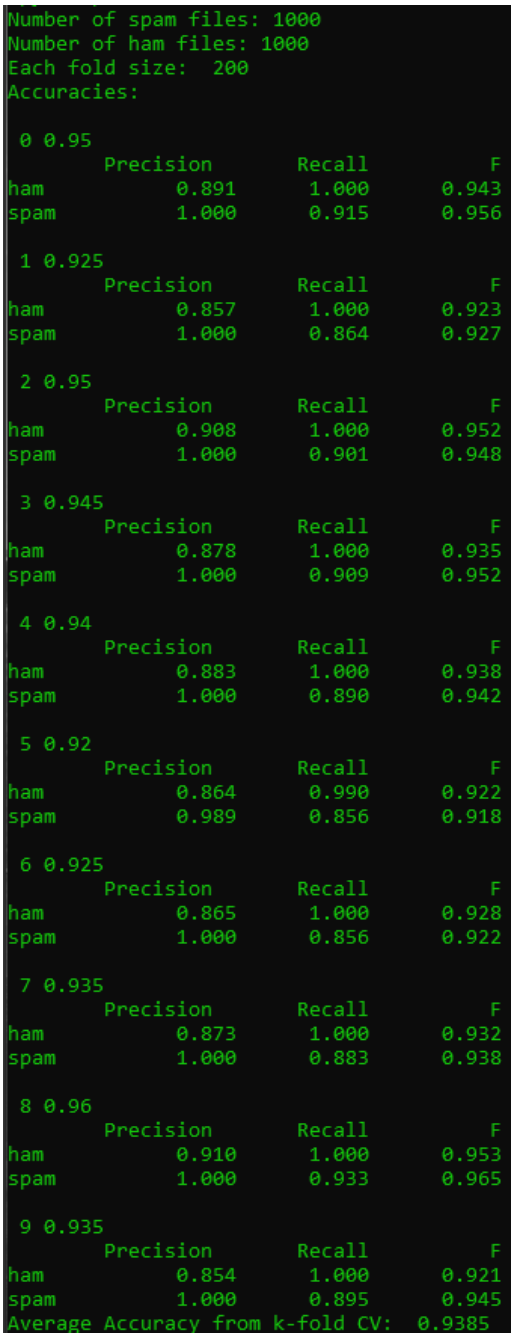


Figure 5: Bigram feature set. No stop word filtering

An experiment with the bigram feature set was to include unigram features as well in the dictionary and top it off with filtering out English stop words. When combined, the average accuracy increases to 95%

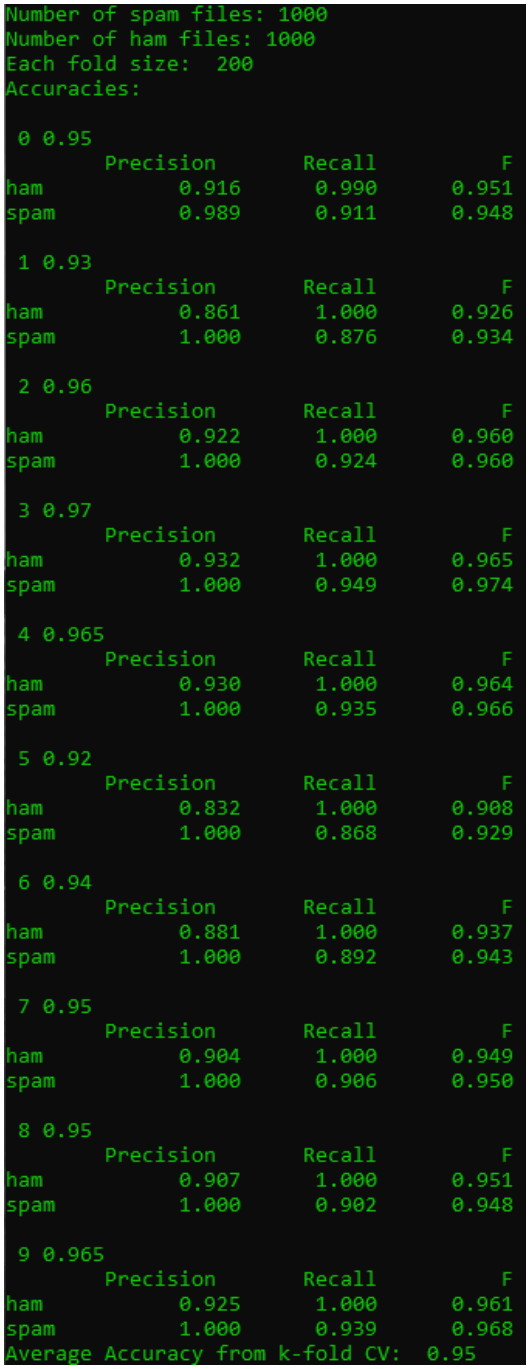


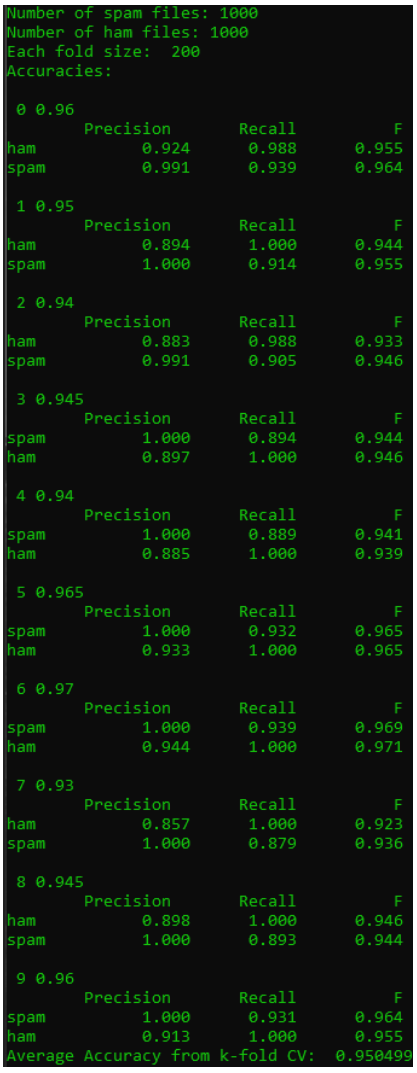
Figure 6: Bigram feature set combined with unigram features. Filters out stop words

The POS tagged feature set produced an average accuracy of 94.2%

```
Number of spam files: 1000
Number of ham files: 1000
Each fold size:  200
Accuracies:

 0 0.92
       Precision       Recall          F
ham           0.846      1.000      0.917
spam          1.000      0.857      0.923

 1 0.96
       Precision       Recall          F
ham           0.918      1.000      0.957
spam          1.000      0.927      0.962

 2 0.96
       Precision       Recall          F
ham           0.909      1.000      0.952
spam          1.000      0.933      0.966

 3 0.965
       Precision       Recall          F
ham           0.938      1.000      0.968
spam          1.000      0.926      0.962

 4 0.925
       Precision       Recall          F
ham           0.833      1.000      0.909
spam          1.000      0.880      0.936

 5 0.93
       Precision       Recall          F
ham           0.882      1.000      0.938
spam          1.000      0.853      0.920

 6 0.94
       Precision       Recall          F
ham           0.872      1.000      0.932
spam          1.000      0.898      0.946

 7 0.945
       Precision       Recall          F
ham           0.892      0.988      0.938
spam          0.991      0.914      0.951

 8 0.965
       Precision       Recall          F
ham           0.929      1.000      0.963
spam          1.000      0.935      0.967

 9 0.91
       Precision       Recall          F
ham           0.825      1.000      0.904
spam          1.000      0.843      0.915
Average Accuracy from k-fold CV:  0.942
```

Figure 7: POS feature set. No stop word filtering

When stop word filtering is applied to the POS Feature set, an average accuracy of 95.05% is achieved, slightly better than the bigram+unigram feature set.



```
Number of spam files: 1000
Number of ham files: 1000
Each fold size:  200
Accuracies:

 0 0.96
       Precision       Recall          F
ham           0.924      0.988      0.955
spam          0.991      0.939      0.964

 1 0.95
       Precision       Recall          F
ham           0.894      1.000      0.944
spam          1.000      0.914      0.955

 2 0.94
       Precision       Recall          F
ham           0.883      0.988      0.933
spam          0.991      0.905      0.946

 3 0.945
       Precision       Recall          F
spam          1.000      0.894      0.944
ham           0.897      1.000      0.946

 4 0.94
       Precision       Recall          F
spam          1.000      0.889      0.941
ham           0.885      1.000      0.939

 5 0.965
       Precision       Recall          F
spam          1.000      0.932      0.965
ham           0.933      1.000      0.965

 6 0.97
       Precision       Recall          F
spam          1.000      0.939      0.969
ham           0.944      1.000      0.971

 7 0.93
       Precision       Recall          F
ham           0.857      1.000      0.923
spam          1.000      0.879      0.936

 8 0.945
       Precision       Recall          F
ham           0.898      1.000      0.946
spam          1.000      0.893      0.944

 9 0.96
       Precision       Recall          F
spam          1.000      0.931      0.964
ham           0.913      1.000      0.955
Average Accuracy from k-fold CV:  0.950499
```

Figure 8: POS feature set with stop word filtering

Strictly looking at punctuation frequency was not explicitly looked at in this course, but I hypothesized that spam emails would contain a rather obscene amount of random, scattered punctuation within them, thus it could be a sturdy indicator of ham or spam. The results were lackluster, as the average accuracy after ten rounds of CV came to be 47%, which is slightly lower than random guessing. A discussion on why this might have performed so poorly will follow in the next section.

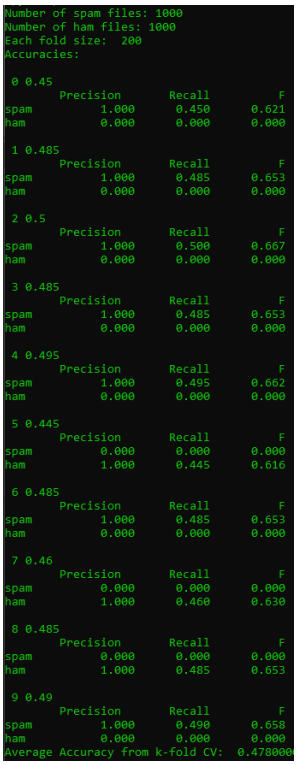


Figure 9: Punctuation frequency

One of the advanced ideas used was the term frequency – inverse document frequency feature set. When filtering out the stop words, this feature set provided an accuracy of 67.3%

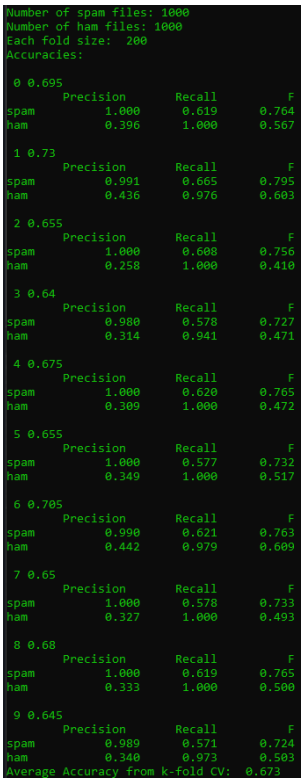


Figure 10: TF-IDF feature set with stop word filtering

This feature set was tested again, but instead of using word TF-IDF, I used punctuation TF-IDF (the previous punctuation experiment was just raw frequency, not IDF). The accuracy of this feature set was improved but still performed much worse than the Bag of words feature set, at 72.2%

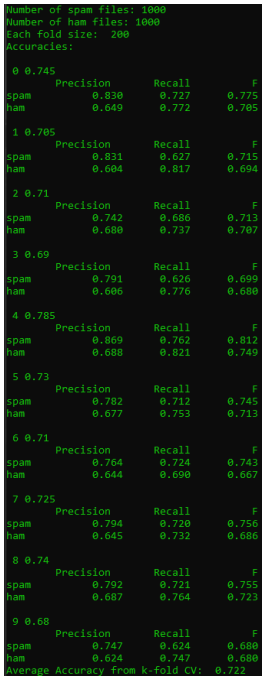


Figure 11: TF-IDF feature set using punctuation only

Discussion on Results

The best performing feature set was the POS features combined with filtering out stop words. At 95.05% it was slightly more accurate then the combined bigram and unigram feature set. The precision for spam detection using these features was nearly perfect, almost 100% each round. Just to recap on the formulas for the evaluation measures:

$$Accuracy = \frac{TP + TN}{\#\ of\ Guesses}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F\ Measure = \frac{2 \times Precision\ \times Recall}{Precision + Recall}$$

Since the value for precision was 1 (and not 0), that indicates that there were 0 false positives when detecting spam emails with the POS feature set. In other words, the naïve bayes classifier did not incorrectly predict a text to be spam for 8 of the 10 rounds of testing. My reasoning for this particular feature set success is that the spam texts likely do not comprise of syntactically complete sentences, therefore the ratio of nouns:pronouns:verbs:adverbs would be off.

It was clear that the unigram and bigram models also worked better than the TF-IDF feature set. This could be due in part to a distinction in diction between the ham and spam emails. When reading through example ham and spam emails, it is evident that this data set was designed to try and trick any model trying to predict labels (even the ham emails have a lot of special characters or funky writing). If there is sufficient frequency of terms in both ham and spam emails, then that feature would be a poor indicator of the label of other emails. However, I believe that further investigation would show that the most common words in ham emails are not also the most common words in the spam emails.

Just as interesting to observe is the weak performance of the punctuation feature set. Again, this was strictly looking at raw frequency of punctuation and non-alphanumerical characters. This was based on the observation that many of the spam texts include special characters. However, this performance warranted a second look into more spam and ham texts. Upon a second look, I saw that many ham emails did indeed have special characters scattered throughout their texts. As mentioned above, this appears to be the data set creator's attempt at deceiving any model's prediction efforts (which is good for practice and study). Ultimately, the punctuation feature set would only contain at most 30 or 32 features in total, too little to create a rich and diverse feature set. This set pales in comparison to the unigram feature set which has 1,500 words as features.

In the future, more feature engineering can be done to extract better feature sets, and potentially combine part of speech tags with bigrams or even trigrams to create the most accurate NB classification model.