IOSR Journal of Engineering (IOSRJEN) www.iosrjen.org
ISSN (e): 2250-3021, ISSN (p): 2278-8719 Special Issue || September 2021 || PP 66-70
7th National Conference On Advancements in Communication, Computing and Electronics Technology-ACCET 2021

Email Spam Detection and Filtering Using Naïve Bayes Algorithm

Aparna Burhade KomalNagul Uttara Divte
Electronics and Telecommunication Telecommunication
M.E.S.C.O.EM.E.S.C.O.EM.E.S.C.O.E

Prof.M.M.Dhanvijay
Electronics and Telecommunication Department M.E.S.C.O.E

Abstract-*Many techniques have been proposed to combat the upsurge in spam. All the proposed techniques have the same target, trying to avoid the spam entering our inboxes. Spammers avoid the filter by different tricks and each of them needs to be analyzed to determine what facility the filters need to have for overcoming the tricks and not allowing spammers to full our inbox. Different tricks give rise to different techniques. This work surveys spam phenomena from all sides, containing definitions, spam tricks, anti spam techniques, data set, etc. Finally, we discus the data sets which researchers use in experimental evaluation of their articles to show the accuracy of their ideas. Machine learning methods of recent are being used to successfully detect and filter spam emails. We present a systematic review of some of the popular machine learning based email spam filtering approaches. Our review covers survey of the important concepts, attempts, efficiency, and the research trend in spam filtering. The preliminary discussion in the study background examines the applications of machine learning techniques to the email spam filtering process of the leading internet service providers (ISPs) like Gmail, Yahoo and Outlook emails spam filters*

Keywords -*Techniques, Emails,* Spam*filtering, Machine Learning, Algorithms*

I. INTRODUCTION

For sharing of important and official information, email is used as a default medium of communication. Most of the institutions and companies prefer to use emails over all other mediums as it is one of the cheapest, easy to use, easily accessible, most official and reliable way of sharing information. It is used widely as it also provides the confidentiality of the data shared. But with the pros also comes the cons, as many people misuse this reliable and easy. way of communication by sending unwanted and useless bulk messages for their own personal benefits. These unwanted emails affect the normal user to face the problems like flooding of the mail box with unwanted emails making it harder to look for the useful once, even sometimes one may skip through important and useful emails because of all these unwanted emails. So, this gives rise to a need of a strong email spam detector which can filter maximum amount of spam emails with a greater accuracy so that a genuine email does not get filtered as spam.

II. II.IMPLEMENTATION

The methodology that is used for the filtering method is machine learning techniques that divide by three phases. The methodology is used for the process of e-mail spam filtering based on Na ̈ive Bayes algorithm
Step 1:
Consider a random email from the ling spam dataset for experimentation.
Step 2:
The considered email is in raw form. To perform the feature extraction/selection and classification procedure, initially email is needed to pre-process. Pre-processing involves the steps of tokenization, stemming and stop word removal.
2.1. Initially, tokenize the email into individual keywords. Tokenization split each individual word into different tokens.
2.2. Remove the stop words from the obtained tokens.
2.3. Perform stemming on the tokens obtained from the previous step. Stemming process reduces the size of the word to its root word. For stemming, a predefined list of possible words with their respective stem words is considered.
2.3.1. For stemming, a list of suffix words is stored into an array with their respective root words.

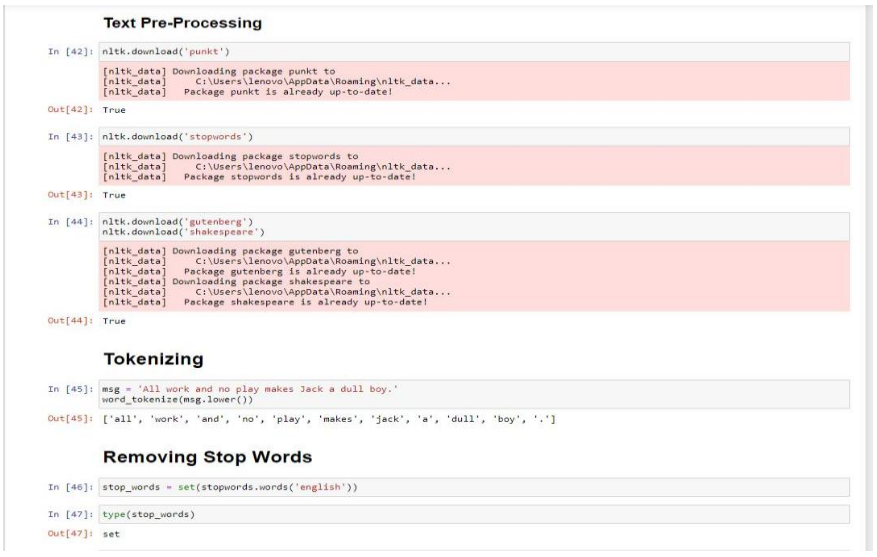
2.3.2. Consider check_token = availability in considered array of root word 2.3.3. If suffix of check_token = true, stem the word to its respective root word from the list of arrays.
2.3.4. Else, there is no need of stemming. Word is already in its root word format. Move to the next token.

Step 3:
Apply Correlation based feature selection approach to select the useful feature words from the pre-processed data. Correlation based feature selection method only selects the feature set which are most related to the particular class. If f is the feature set with k number of features and c is number of classes then CFS can be applied as mentioned in Equation 2.
Where, rcf is the average of feature-class correlation,rcf is the average of feature-class correlation, rff is the average of feature-feature correlation.
Step 4:
Calculate the probability distribution of the tokens along with selected features using NB approach. The formulation for the probability distribution is presented in Equation 3.
Where, f is any feature vector set (f1, f2, f3,....fn) and y are the class variables with m possible outcomes (y1, y2, y3,....yn). P(y|x) stands for posterior probability, P(x|y) is any particular class on which P(y|x) is dependent. P(x) is evidence depending on the known feature variables, P(y) is the prior probability.
Step 5:
Apply PSO approach to optimize the parameters of NB approach.
5.1. All the tokens are considered as the particles. Initially these particles randomly fly and search for the food sources in the form of the best feature match for tokens. Then search for the local and global solution.
5.2. The performance of each particle depends upon the similarity with the features that has to optimize.
5.3. Each particle flies over the n- dimensional outer search space and keep updating the following information:
Xi – current position of particle
Pi – the personal best position of particle x
Vi – the current velocity of particle
5.4. The velocity updates in PSO can be calculated using the formula given below by Equation
(4)
Now, Vi is the new velocity. So, the position of the particle updates with the velocity as defined with Equation
(5)
5.5. Update the positions for each particle and store the global best solutions.
Step 6:
Based on the evaluated feature similarity using PSO, classification of tokens is declared as spam ornon spam.
Step 7:
Further, final classification is performed by evaluating the probability of spam or non-spam tokens in a sentence. 7.1. If the probability value of spam tokens is more, then email is considered as spam email. 7.2. Else, email is considered as non-spam email.
Step 8:
Store the email as spam or non-spam and repeat the process for all the emails.

III. RESULTS

Pre-processing

Pre processing includes:
First, converting the words to lower case, second, Tokenising the words, third, Removing all the stop words, fourth, stemming the words, fifth, removing punctuations, and lastly, removing the HTML tags.
After successfully training and test our data set we have implemented our Naive Bayes classifier and got results.



Text Pre-Processing

```
In [42]: nltk.download('punkt')

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!

Out[42]: True

In [43]: nltk.download('stopwords')

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

Out[43]: True

In [44]: nltk.download('gutenberg')
         nltk.download('shakespeare')

[nltk_data] Downloading package gutenberg to
[nltk_data]     C:\Users\lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package shakespeare to
[nltk_data]     C:\Users\lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Package shakespeare is already up-to-date!

Out[44]: True
```

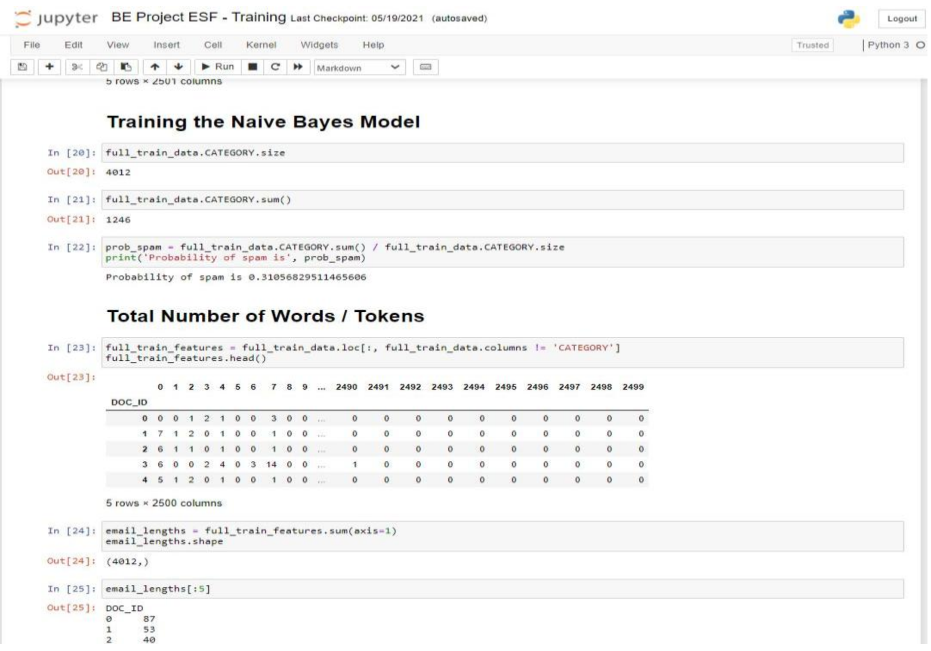
Tokenizing

```
In [45]: msg = 'All work and no play makes Jack a dull boy.'
         word_tokenize(msg.lower())

Out[45]: ['all', 'work', 'and', 'no', 'play', 'makes', 'jack', 'a', 'dull', 'boy', '.']
```

Removing Stop Words

```
In [46]: stop_words = set(stopwords.words('english'))

In [47]: type(stop_words)
Out[47]: set
```

Training a Naïve Bayes Classifier
What is a Training data set?
We use training dataset and test dataset in supervised.
1) The purpose of training data set is to discover the productive relationship by using a model.
2) With training data set, the model can learn the behavior of the data and tweak itself.
3) The model is built based on the data it discovers in the training model



5 rows × 2501 columns

Training the Naive Bayes Model

```
In [20]: full_train_data.CATEGORY.size
Out[20]: 4012

In [21]: full_train_data.CATEGORY.sum()
Out[21]: 1246

In [22]: prob_spam = full_train_data.CATEGORY.sum() / full_train_data.CATEGORY.size
         print('Probability of spam is', prob_spam)

         Probability of spam is 0.31056829511465606
```

Total Number of Words / Tokens

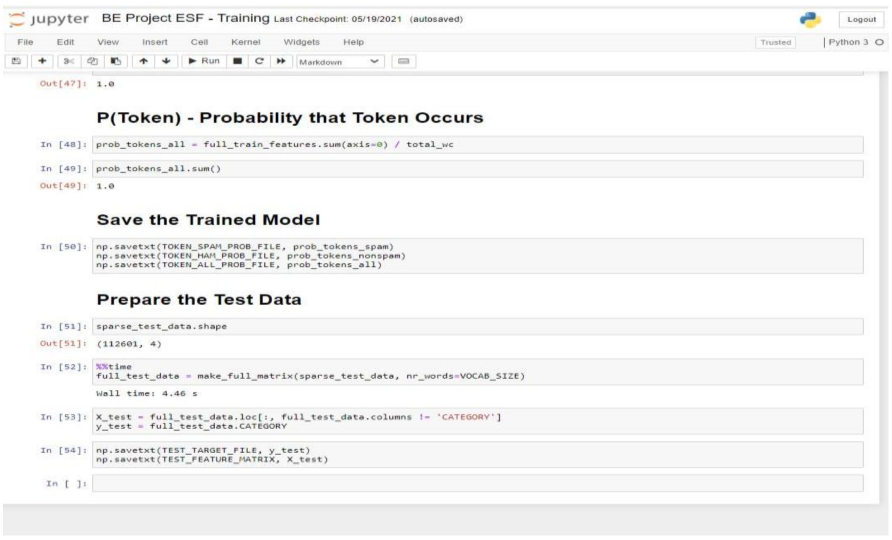
```
In [23]: full_train_features = full_train_data.loc[:, full_train_data.columns != 'CATEGORY']
         full_train_features.head()
Out[23]:
```

DOC_ID	0	1	2	3	4	5	6	7	8	9	...	2490	2491	2492	2493	2494	2495	2496	2497	2498	2499
0	0	0	1	2	1	0	0	3	0	0	...	0	0	0	0	0	0	0	0	0	0
1	7	1	2	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
2	6	1	1	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
3	6	0	0	2	4	0	3	14	0	0	...	1	0	0	0	0	0	0	0	0	0
4	5	1	2	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 2500 columns

```
In [24]: email_lengths = full_train_features.sum(axis=1)
         email_lengths.shape
Out[24]: (4012,)

In [25]: email_lengths[:5]
Out[25]: DOC_ID
         0     87
         1     53
         2     40
```

Testing
1) Once the model is trained we use test data set to get tye accuracy of the model.
2) Since the model has not seen the test data set during the training phase, it gives us impartial results.
3) Therefore it is very important to keep test dataset separate from training dataset.



```
Out[47]: 1.0
```

P(Token) - Probability that Token Occurs

```
In [48]: prob_tokens_all = full_train_features.sum(axis=0) / total_wc

In [49]: prob_tokens_all.sum()
Out[49]: 1.0
```

Save the Trained Model

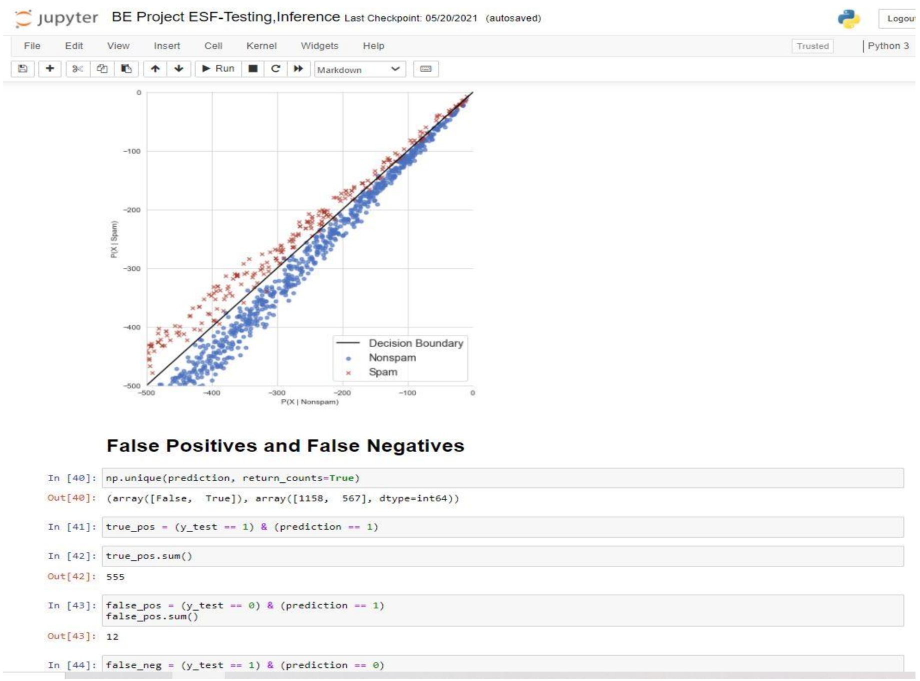
```
In [50]: np.savetxt(TOKEN_SPAM_PROB_FILE, prob_tokens_spam)
         np.savetxt(TOKEN_HAM_PROB_FILE, prob_tokens_nonspam)
         np.savetxt(TOKEN_ALL_PROB_FILE, prob_tokens_all)
```

Prepare the Test Data

```
In [51]: sparse_test_data.shape
Out[51]: (112601, 4)

In [52]: %%time
         full_test_data = make_full_matrix(sparse_test_data, nr_words=VOCAB_SIZE)

         Wall time: 4.46 s

In [53]: X_test = full_test_data.loc[:, full_test_data.columns != 'CATEGORY']
         y_test = full_test_data.CATEGORY

In [54]: np.savetxt(TEST_TARGET_FILE, y_test)
         np.savetxt(TEST_FEATURE_MATRIX, X_test)

In [ ]:
```

Data Visualisation

1)The data can be seen visualized as spam and ham on a graph.

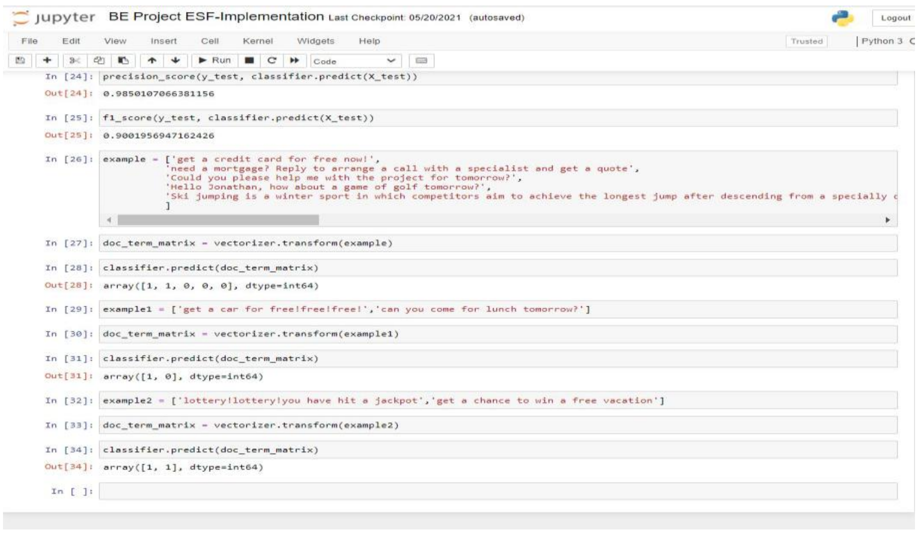


False Positives and False Negatives

```
In [40]: np.unique(prediction, return_counts=True)
Out[40]: (array([False,  True]), array([1158,  567], dtype=int64))

In [41]: true_pos = (y_test == 1) & (prediction == 1)

In [42]: true_pos.sum()
Out[42]: 555

In [43]: false_pos = (y_test == 0) & (prediction == 1)
         false_pos.sum()
Out[43]: 12

In [44]: false_neg = (y_test == 1) & (prediction == 0)
```

Implementation

1)After successfully training and testing our data set we have implemented our Naive Bayes classifier and got results


jupyter
BE Project ESF-Implementation Last Checkpoint: 05/20/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [24]: precision_score(y_test, classifier.predict(X_test))
Out[24]: 0.9850107066381156

In [25]: f1_score(y_test, classifier.predict(X_test))
Out[25]: 0.9001956947162426

In [26]: example = ['get a credit card for free now!',
                    'need a mortgage? Reply to arrange a call with a specialist and get a quote',
                    'Could you please help me with the project for tomorrow?',
                    'Hello Jonathan, how about a game of golf tomorrow?',
                    'Ski jumping is a winter sport in which competitors aim to achieve the longest jump after descending from a specially
                    ]

In [27]: doc_term_matrix = vectorizer.transform(example)

In [28]: classifier.predict(doc_term_matrix)
Out[28]: array([1, 1, 0, 0, 0], dtype=int64)

In [29]: example1 = ['get a car for free!free!free!','can you come for lunch tomorrow?']

In [30]: doc_term_matrix = vectorizer.transform(example1)

In [31]: classifier.predict(doc_term_matrix)
Out[31]: array([1, 0], dtype=int64)

In [32]: example2 = ['lottery!lottery!you have hit a jackpot','get a chance to win a free vacation']

In [33]: doc_term_matrix = vectorizer.transform(example2)

In [34]: classifier.predict(doc_term_matrix)
Out[34]: array([1, 1], dtype=int64)

In [ ]:
```

IV. CONCLUSION

E-mail spam filtering is an important issue in the network security and machine learning techniques; Na¨ıve Bayes classifier that used has a very important role in this process of filtering email spam. The quality of performance Na¨ıve Bayes classifier is also based on datasets that used. As can see, dataset that have fewer instances of e-mails and attributes can give good performance for Na¨ıve Bayes classifier. Na¨ıve Bayes classifier also can get highest precision that give highest percentage spam message manage to block if the dataset collect from single e-mail accounts. So we can see, why performance of Na¨ıve Bayes classifier is good when used SPAMBASE dataset. There are many models of spam classification and filtration. Many of them are based on using single classifier. But in recent times,the use of multiple classifiers is advocated and research is done by using a combination of statistical techniques during different phase of the spam filtration. In future, research can be conducted to further improving the classification accuracy.

REFERENCES

[1]. https://in.springboard.com/blog/email-spam-filtering-using-naive-bayes-classifier/
[2]. https://www.ijsdr.org/papers/IJSDR1906001.pdf
[3]. https://towardsdatascience.com/na%C3%AFve-bayes-spam-filter-from-scratch-12970ad3dae7
[4]. https://medium.com/@ar3441/spam-filtering-using-naive-bayes-98a341224038