ML Algorithm based Email Processing Model

by

Karteek Pradyumna Bulusu

A Project Report Submitted
In
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Computer Science
Supervised by

Prof Dr. Carol Romanowski

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

December 2016
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Serial Number</th>
<th>Topic</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>List of Figures</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Acknowledgement</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Abstract</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Problem Statement</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>Motivation</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Description</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Existing Model</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>Previous Work</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>Design and Implementation</td>
<td>9</td>
</tr>
<tr>
<td>9.1</td>
<td>Collection of data</td>
<td>9</td>
</tr>
<tr>
<td>9.2</td>
<td>Text Preprocessing</td>
<td>11</td>
</tr>
<tr>
<td>9.3</td>
<td>TF and Tri-gram</td>
<td>12</td>
</tr>
<tr>
<td>9.4</td>
<td>Multinomial Naïve Bayes algorithm</td>
<td>14</td>
</tr>
<tr>
<td>9.5</td>
<td>Testing over My Mails</td>
<td>15</td>
</tr>
<tr>
<td>9.6</td>
<td>Optimizing Running-time.</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>Results</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>Advantages</td>
<td>19</td>
</tr>
<tr>
<td>12</td>
<td>Conclusion</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>Future Work</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>References</td>
<td>20</td>
</tr>
</tbody>
</table>
## 1. LIST OF FIGURES

<table>
<thead>
<tr>
<th>Serial Number</th>
<th>Title</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sample data set after collection process</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Gmail rules option for email filtering</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Architecture of the model</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>After scrapping the data. Prior to initial preprocessing</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>After removing HTML tags and other unwanted data from each mail</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Stop words in the sample mail are marked.</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>The highlighted content are special character</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>Tokenized words</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>Result after performing TF over the tokens</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>Sample list of features</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>Snippet for Text preprocessing in Python</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>Sample algorithm of Classification Model</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>Test result for Spam mail</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td>Running-time for prediction and total algorithm</td>
<td>17</td>
</tr>
<tr>
<td>15</td>
<td>Running-time for prediction and total algorithm</td>
<td>17</td>
</tr>
<tr>
<td>16</td>
<td>Word cloud of all my features</td>
<td>18</td>
</tr>
<tr>
<td>17</td>
<td>Word cloud of Inbox mails</td>
<td>19</td>
</tr>
<tr>
<td>18</td>
<td>Word cloud of Spam mails</td>
<td>19</td>
</tr>
</tbody>
</table>
2. ACKNOWLEDGEMENT

With a sense of gratitude, I acknowledge the support of my advisor for this capstone, Prof Dr. Carol Romanowski for being there and helping with all issues and suggesting changes and ideas. I utilize this platform to also be grateful to my instructor Prof. Dr. Leonid Reznik who provided me with inputs regarding the milestone presentations as well as the poster.

I would always be grateful to the researchers in this domain of Machine learning whose research papers, tutorials and ideas have proven to be of a great advantage for this project. Those articles helped me understand the deep concepts pertaining to text analysis and implement in my capstone.
3. ABSTRACT

There is an importance of utilizing Machine learning algorithms in real life scenarios. Email systems are always been a rule based engine and there are few flaws but are effective. With a notion of overcoming these flaws and make the existing model work for any type of mail, I am working on developing a Machine learning based email preprocessing system which would analyze the emails already existing and correlate with its section and would be used to predict and classify the new mails. This is a case of text understanding and classification. The initial stages would include preprocessing of the emails. From all the emails I am parsing and considering only the ‘Subject’ and the ‘Body’ of the email. This is later preprocessed using the NLTK packages and techniques such as stemming, tokenization, POS tagging and obtaining the term frequency of each word with respect to the document. Using Multinomial Naïve Bayes, term frequency and tri-gram concept, I am training my model where all the mails are tagged with their respective labels. One of the flaws in existing model I am dealing with this is inappropriate classifying of the mail. There are instances in existing model where certain emails from one particular user goes to inbox but few from the same user goes to the Spam section. Also there are few cases when we find few important emails in the spam folder. This can be really harmful when the user is particular about its mail. Machine learning approach can resolve these types of situations if trained appropriately with good relevant data.

4. PROBLEM STATEMENT

I aimed to solve the issue of misclassification of emails that occur in the present rule-based email processing system and propose a Machine learning based email processing system which analyzes the subject and body of the content, understand the context of the mail and its importance and classify the mail to Inbox or spam folder. I am taking into consideration few factors that affect the classification. With this model, I aim to classify the emails accurately and with minimal delay in receiving the mail. This work is a different technique in which email processing system can be built.

5. MOTIVATION

After identifying demerits of knowledge based engines in large scale usage, organizations are shifting domains towards other means. One such alternate is usage of Machine and Deep learning algorithms. Recently, Google implemented Robotics arm using Artificial Neural network. This adding to my immense interest towards data science has intrigued me to implement email processing system using Machine Learning algorithms and see how the results and behavior of mails are towards ML algorithms and the results are impressive and proved to be one of the good alternatives for rule-based engine.

I believe this is step one for implementing new simpler technique for managing mails which plays one of the most crucial role in many individuals in the present era. And this step would prove to be of great advantage in other related endeavors.
6. DESCRIPTION

Machine learning algorithms have largely evolved and there is continuous research happening in this field both in the domain of image science as well as data science. With this advancement in this domain, research of its implementation in various established domain is been tested. Recently, Google robotics arm which was based on rule-based engine is implemented using Machine learning algorithms.

This fascinated to try implementing this in email processing system. Gmail as well as Yahoo mails have some misclassification rate where few of the important mails sometime go to the spam folder. This is a serious issue since false negative aspect in mails is very risky since emails play a very crucial role in the present era for every individual.

For example, if you book for an event of place an order, in the confirmation sometimes they say “Do check your spam mails as well for the confirmation.”. Why do they mention that? Because, they know that there is a chance of misclassification because existing though is accurate and fast but do have high rate of false negative.

The rules in a rule-based system are to be in regular based updated and lot of maintenance is need for it. Machine learning algorithms can help avoid this problem and produce the results accurately. If we continuously train the mails as and when they come based on their label, this type of misclassification can be easily avoided. This simple thought has triggered me work on this domain.

<table>
<thead>
<tr>
<th>Body</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>security alert confirm sky financial group sky bank member email inform sky bank account access account may</td>
<td>Spam</td>
</tr>
<tr>
<td>pg e cash formatted pg e cash report let know updates thank</td>
<td>Inbox</td>
</tr>
<tr>
<td>start date hour start date hour ancillary schedules awarded variances detected log messages parsing file sched</td>
<td>Inbox</td>
</tr>
<tr>
<td>th annual calendar make plans join th annual boil chimney rock jambalaya beer margaritas advance door comp</td>
<td>Inbox</td>
</tr>
<tr>
<td>n search lo search com</td>
<td>Spam</td>
</tr>
<tr>
<td>would like make money read information advertisement click hereto removed mailing list click</td>
<td>Spam</td>
</tr>
<tr>
<td>failure send program net afraid able deliver message following addresses permanent error given sorry work sc</td>
<td>Spam</td>
</tr>
<tr>
<td>launched website one visits submitting website search engines may sales dramatically invested time money w</td>
<td>Spam</td>
</tr>
<tr>
<td>start date hour start date hour ancillary schedules awarded variances detected log messages parsing file sched</td>
<td>Inbox</td>
</tr>
</tbody>
</table>

Figure 1: Sample data set after collection process

Figure above (Figure 1) displays the sample dataset I considered for the project prior to any textual preprocessing. It has “Body” which holds the mails and Class which is bi-split class “Spam” or “Inbox”. The data is equally split data and is not biased since those factors would affect the classification results.

As seen, there is lot of preprocessing need to be done such as spell correction and removal of stop words which were done in the later stages.
Following are the main tasks of the project:

- Scrapping mails from my inbox as well collect data from online reliable resources. Removing the HTML tags.
- Preprocessing the mails and correcting the spelling. Also, I removed send to:, BCC: as well as CC: tags and email ids since I am dealing with only body and subject of the mails and not particular about the email address.
- Calculate the term frequency and perform tri-gram over the mails in order to obtain the context of the mails.
- Once context is known, data is trained with Multinomial Naïve Bayes model and tested over my mails.
- Work on improving the running time of the algorithm and make it as minimal as possible since receiving of the mails shouldn't be affected due to this.

7. EXISTING MODEL

We already have email processing techniques that all the email systems like Gmail, Yahoo mail etc use but these email processing techniques are rule-based engines. Rule-based engines are used to understand the situation and interpret information based on it. The system which Gmail [2] uses is a Rule-based engine where all the mails are passed through this filter which checks for all the rules in the rule engine and based on the results specified in the rules, the mail gets classified. [1] This model provides over 98% accuracy but there are major instances where the mails were misclassified. For this reason, sometimes when the user book a ticket or order something online, there is a disclaimer which asks the user to check the spam folder also for confirmation.

Rule-based engines are good to use when the rules are limited to certain extent. Rule-based engine with thousands of rules will increase the space occupancy, manage the rules as well as time to process through all the rules. For this reason, all the domains in industry are deviating from rule-based engine to machine learning domain or some other domain.

Recently, robotics domain at Google Inc. is deviated from rule-based engine to Deep learning domain. My capstone is a similar attempt to build an email processing system using Machine Learning algorithms.

If you notice any misclassification, you have to make your own rule that avoid the misclassification next time. But one could not write a rule for each misclassification. Following is the image of a Gmail rules section.

In another research paper [2], authors implement spam filter for messaging system where talk about how to determine the context and find if that particular message is relevant or not and this can be totally related to the email system where I would be working on finding the context and determining and removing unwanted emails from inbox and pushing them to the spam folder.
As mentioned in Figure 2: for every misclassification we can create a filter a misclassification but we can’t do for every mail. To avoid this delimitation, I attempted to implement email processing system using Machine learning algorithms.

8. PREVIOUS WORK

A lot of research has happened in the domain of Email processing has taken place previously. In one of the research paper I referred to, authors talked about the implementation of Naïve Bayes and Support Vector Machine in the domain of email classification. [4] Authors consider whole email as such with email address and hyperlinks mentioned in the emails. They performed text preprocessing over the emails such as removal of stop words, correcting the spellings before including them in the features list and analyzing.

The features list consisted to the words and not the word count. Naïve Bayes algorithm was implemented over this features list of the words and based on the context and occurrence of word independent of the frequency of it is trained and later tested on the mails. Further, Support vector machine was implemented on it since SVM [5] works well for bi-split class since implementing linear margin is easier considering the space of the features. Contrast to the present system, I am also taking into account the occurrence of the word by implementing TF (Term frequency) over each mail. Further, to determine the context of occurrence, I am implementing tri-gram prior to performing analysis stage.
9. DESIGN AND IMPLEMENTATION

The figure below (Figure 3) describes the basic flow and architecture of the model. The prime focus of this model is obtain highly accurate classification results in minimal delay time possible.

![Diagram showing the flow and architecture of the model.]

**Figure 3:** Architecture of the model

### 9.1 COLLECTION OF DATA:

Since few of the data sources are unreliable since it might add noise to the model, some part of the data was obtained by scrapping from my mails. Used BeautifulSoup package in Python to scrap my mails and write to the text file.

After that is been performed, I read through the files and removed unwanted features such as email addresses (send to, cc, bcc etc.) and kept only the subject as well as the body of the emails. But major portion of the data was collected from Enron dataset [3].

The data consisted of various attributes such as delivered by, data of delivery and signature of the mail which is not needed to determine if the mail is Spam or Inbox mail and hence those tags were removed. These were removed with simple regex function which would check for ‘<’ and ‘>’ and remove all the content that holds between that. This way, all the HTML tags were also removed since they too are irrelevant for the analysis.
HTML tags are common in both Spam and Inbox mails and doesn't affect the accuracy of the model in any manner. The figure below displays the sample data after collection process.

From: "Clarence Puckett" <c_puckett_ie@cic.ca>
To: rait@bruce-guenter.dyndns.org
Subject: V, icodin, limited supply available
Date: Tue, 24 Aug 2004 04:44:17 +0000
MIME-Version: 1.0
Content-Type: text/html;
   charset="us-ascii"
<html>
Looking for V1C0DL1N and/or HYDROCODONE? <p>
Only place you can get it <b>without prescription</b>: <a href="http://oqobrlbuostq.thepillslive.com/?x8=k7m256">Med-Network</a></p><br>
Now with same-day shipping, we are unbeatable. <p>
Under-priced deals on other products like PHENTERM1NE too. <p>
<a href="http://tauBBkbbnda.thepillslive.com/?x8=k7m256">DEALS WONT LAST - VISIT US NOW!</a></p>
<br><br><br><br><br><a href="http://dvuvzvbpbbugr.znnyoua.com/c.html">No thanks</a>
</html>

**Figure 4**: After scrapping the data. Prior to initial preprocessing.

Subject: gas master
david:
pursuant to our discussion, attached is the revised gas master which includes the collateral annex as exhibit d. sorry for the previous confusion.
jeff

**Figure 5**: After removing HTML tags and other unwanted data from each mail.

Even though I have performed initial preprocessing and eliminated all the unwanted data from the mail and stored as a text file, there is still lot of noise in the data which would affect the analysis results in a negative way.

Also, I had utilized data from Enron dataset [3] obtained online. This data was free of HTML tags and other unwanted content which were removed using Regex operations. I performed this using Python where I read each line and remove all the content that lies between the square brackets since it doesn't help me during my
analysis phase. The result of this is depicted in Figure 5. This isn't completely preprocessed as we can notice we need to remove special characters and other noise elements from the data which was done in the later stages.

9.2 TEXT PREPROCESSING:

Preprocessing of the data plays the most crucial role in any analysis. Here since I was dealing with textual data and these words would be my features, I should be particular to what all I should add to my features list and what I shouldn’t and if added would affect my analysis results.

I performed text processing and analysis using Python. Python NLTK provided packages for performing different types of preprocessing which helped me clean my data to major extent. With cleaning process, the order in which different cleaning techniques performed matters since it avoids each technique to deal with large amount of unwanted elements.

a. Stop word removal:

Stop word removal was first cleaning technique I implemented since this would discard all the unwanted words from my emails hence reducing number of words to deal with in later stages. This was done using nltk.corpus [6] package which contains list of stop words. I converted all the words into lower case and performed this since Python ignores stop word in uppercase.

For example: If the input was ['I am capstoning my capstone and appreciate my seniors who have capstoned successfully! '], then the output was ['Capstoning my capstone appreciate my seniors who capstoned successfully']

Figure 6: Stop words in the sample mail are marked.

b. Removal of special character [7]:

All the special characters such as [ “!”, “,”, “.”,”/”,”?”,“\”,”|”, “”,”`” ] etc. are removed from the resultant text data since these doesn’t add on to the analysis results instead harm the analysis. Post removal of special characters, I took care of punctuations and removed them from the data. I used ‘string’ package in python

we appreciate the pressure and long hours people in houston are putting into getting the site up and running . we applaud their efforts and do not wish to add to their burden . i tell you this only to make you aware of our situation , should any problems arise because of the unedited text . hopefully , if anything does arise , problems can be easily and quickly corrected .
which hold a list of punctuation words which are matched with the data and if exists, removed from the corpus.

**Figure 7:** The highlighted content are special character

c. Tokenization:

Now we have received all the features from each mail. In my analysis phase, since I am dealing with each word individually and not as a sentence or phrase, I need to split the words from the sentences/phrases. And tokenization helps me achieve this task very easily. Tokenization reads the line and considers each word as a separate token and resultant is list of words separated by comma. I used nltk.tokenize [8] package which splits the string. Sample output would be something like: ['Capstoning', 'my', 'capstone', 'appreciate', 'my', 'seniors', 'who', 'capstoned', 'successfully'].

**Figure 8:** Tokenized words.

d. Spell check:

I ran an English dictionary spell checker on each token to make sure no feature in my features list is grammatically incorrect. I used ‘Enchant’ package [9] in Python 3 which I would run over each token to see if the word exists in the English dictionary. If it doesn’t, it corrects the spelling and replaces with corrected word. This provides me cleaned data to run my algorithm on.

9.3 TERM FREQUENCY AND TRI-GRAM:

I stored all the mails from each text file and into a CSV file with the label pertaining to each label. I maintained an equal balance of the count of mails for both the classes ‘Spam’ and ‘Inbox’ otherwise it might act like a weak learner data and provide over fitted results.

With this data, next milestone was to calculate the term frequency of each token in the mail dataset since I wanted to analyze the usage of each word in both Spam and Inbox mails and assign weight to that word based on it. I used TF method which
considers each token and calculates its occurrence in whole document and stores as key value pair.

For example, [‘mail’, 720]. Additionally I am checking how many times each word occur with each particular label such as how many times the word ‘mail’ occur in spam mails and how many times it occur in Inbox mails. I am using ‘sklearn.feature_extraction.text’ [10] package in order to perform this task. Sample output for this task would be like the following image:

```
[‘mail’, 720], [‘greeting’, 679], [‘capstone’, 681], [‘(com)’, 590], [‘greeting’, 541], [‘year’], 542], [‘many’], 542], [‘inform’, 510], [‘(tax)’, 499], [‘service’], 471], [‘(sen)’, 469],
```

**Figure 9:** Result after performing TF over the tokens.

After I have come to know the occurrence of each word in the features list, I should write an algorithm to find the context of the usage of the word. There are many techniques such as POS tagging (Part of Speech tagging) which would see the usage of each word in terms of English grammar and find context. But for my dataset I felt usage of Tri-gram is a better option that POS Tagging since tri-gram would help my model understand the repetition of words and usage in better way. Since I am dealing with emails, there is a little margin of difference in context in mails when it comes to Inbox and Spam mails.

Tri-gram comes from root n-gram technique which takes the input a sentence and considers three cases. In first case, it considers each token in the line as an individual entity. In the second case, it take pairs of subsequent token and analyze. In the third case, it takes group of three subsequent characters.

Example: If the line is ['Capstoning my capstone appreciate my seniors who capstoned successfully'].


```
[works, sap, team, work, us, issues, list, items, laurel, reviewed, forwarded, glover, laurel, glover, action, list, items, focusing, along, names, working, specific, problem, keep, posted, items, aware, everything]
```

**Figure 10:** Sample list of features
The figure above (Figure 10) displays list of features. As shown, all the features are comma separated and this is prior to finding term frequency over them. These features are the combination of spam mails as well as inbox mails. These features are later labelled with their labels as well as their term frequencies.

```python
for file in glob.glob("*.txt"):
    openfile = open(file, "r").read()
    filelist = []
    lower = openfile.lower()

    lower = lower.replace("\n","
    lower = re.sub(r'\d+',',lower)
    lower = re.sub(r'[^\w]','', lower)
    lower = re.sub(r'\https:\/:\/\/.*.\',', lower, flags= re.MULTILINE)
    lower = lower.replace(punctuation,"

    phase_one = lower.replace('subject re ','').replace('subject fw ','').replace('enron','').replace('www','').replace('com','').join([word for word in phase_one.split() if word not in stopwords.words("english")]
```

**Figure 11**: Snippet for Text preprocessing in Python.

The above picture, figure 11 depicts the snippet for textual preprocessing where initially all the words are converted into lower case since no NLTK package considers upper case for preprocessing. Later I am getting rid of punctuations and replacing few of the most commonly occurring irrelevant words and removing stop words as well.

### 9.4 MULTINOMIAL NAÏVE BAYES ALGORITHM:

Now we have extracted features and have understood the context of the model, next step is to build a training model of the same. This in layman terms means is to make the machine understand the context and match labels to each mail.

I am using Multinomial Naïve Bayes algorithm to build my training model. Multinomial Naïve Bayes implements same probability calculation which Naïve Bayes does but this considers the term frequency which building model where as simple Naïve Bayes takes the word as such and trains the data.

While training the data, all the tokens and n-gram pairs get the label tagged with it depicting that the machine has learnt the label for each mail and also know the context for each label and label. This helps during classification when new mail is to be checked with this model. During this situation, features of that mail are been matched with this trained data and context is identified and new mail is been labelled with same label the trained data of that context is labelled with. I used Python sklearn.naive_bayes [12] package to perform this task. After this step, my model is been trained and is ready for testing with new mails.
Naïve Bayes implements Bayesian probability calculation. This is widely used in textual analysis since Naïve Bayes has the property of taking each word as an independent feature and doesn't create any bias or overfitting. This is the primary reason I opted for this algorithm.

The functioning of the Multinomial Naïve Bayes is more or less similar to the algorithm depicted below.

```python
for each word in test case:
    spamPercentage = Count(Spam(word)) / Count(Spam(word)) + Count(Ham(word))
    hamPercentage = Count(Ham(word)) / Count(Spam(word)) + Count(Ham(word))
    if hamPercentage > spamPercentage:
        countOfHam += 1
    elif hamPercentage == spamPercentage:
        Check for other words
    else:
        countOfSpam += 1
if countOfHam > countOfSpam:
    Class = 'Ham'
elif countOfHam == countOfSpam:
    Class = 'Ham'  # It is better to put the mail on inbox than labelling it Spam. What if it is important.
else:
    Class = 'Spam'
```

**Figure 12**: Sample algorithm of Classification Model.

Figure 12 depicts the sample functioning of the prediction algorithm. So for each word in the new mail given to the algorithm, it calculates the percentage of occurrence of that word with Spam label as well as with “Inbox” label based on the percentage value, count of that particular label is incremented.

At the end when all the words were matched with the features list and we have the count of how many words occurred in Spam section in trained model and how many in Inbox mails, we determine what label the new mail should get based on which count is greater. And if the count of occurrence is equal, it is labelled as Inbox mail since it is better to hold less important mail in Inbox than holding important mail in spam folder.

**9.5 TESTING OVER MY MAILS:**

Now that my model is ready and the “Machine” knows what my mails are, now is the time to test the machine and see how accurate the results are. I am testing my mails on the test data which I had created and also on my mails just to make a comparative analysis.
When I am testing my model with Test dataset, it is predicting the label for each mail and storing in a list. I am considering that list and comparing with list of actual labels for each of the test mails and making a comparison. The accuracy of the model can be determined by the number of matches in both the list. I checked this model on all of my spam mails where I include only the subject and the body of the mail and classified the mails accurately. Though there were spam mails which almost looked like Inbox mails and had word features that resemble inbox mails, few of those were misclassified.

9.6 OPTIMIZING RUNTIME OF ALGORITHM:

Since I am building an email processing model, I should be very careful about how much time it is taking to predict the label of the new mail and it should be as minimal as possible. There are many factors affecting the runtime of the algorithm such as optimality of the code, number of features in trained data, the source new mail is coming from and the machine the model is running it (usually a laptop).

Initially, prediction of the label was taking over more than two minutes since I was using NLTK package for Naïve Bayes. Realizing that I shifted to Multinomial Naïve Bayes which is quicker and the time taken for prediction is about 8 seconds which is comparatively quicker. It is further running in less than 5 seconds if tri-gram wasn’t implemented but that reduces accuracy of the classification. Other alternatives that can be implemented to reduce time is to implement own n-gram over all the features.

10. RESULTS

The results are in the form of prediction and accuracy of the model. As mentioned above, accuracy is been calculated by matching the predicted labels with actual labels. I am achieving an accuracy of over 96% and running time of the algorithm for prediction is not more than 9 seconds and for whole model to run is around 20 seconds.

I tested over the test data as well as my emails just to perform a comparative analysis. Initially the accuracy was very bad of approximately 51% since the features list had too many inappropriate words and that affected the analysis. This led me repeat my cleaning process again. I then implemented spell checker which corrected all the spellings in my mail and created proper list of features. Below is the result of its working over my email.

The Test mail is: ['Hi Harsh k Pradhan. Still waiting in ATM Queue? Get a CreditCard & Go Cashless +Free Life Time Credit CardOffer']

Target Label: Spam mail.
Predicted Label: ['Spam']
Accuracy of the model is: 95.8996294924 %

Figure 13: Test result for Spam mail
Figure 14: Test result for Inbox mails

Figure 13, 14 are the sample test results obtained after I tested over my emails. As shown in the image, the results are highly accurate and accuracy is approximately 96%

Second observation I wished to highlight in this is understanding of the context. As one can see in each of the test mails, ‘Hi Karteek Pradyumna’ is present. This gives the model is predict both as ‘Inbox’ or both as ‘Spam’ mail. But machine also understands the context and usage of other feature words and compare with label assigned to those features in the trained model and determines the results.

Secondly, one doubt that occurs to any data scientist seeing 96% accuracy is if the model is overfitting. In order to assure that the model isn’t overfitting, I ran this model on my mails and it works well with my mails. Another measure I took to check this is by repeatedly running the model over same mail. If the results vary, it affirms that the model is not “predicting”, its “guessing” and that means that the model overfits. But this was given same and accurate results each time.

In order to see the features and how weighted each word is, I generated the word cloud of my list of features. There are approximately 800,000 feature words in my trained model through which every word of test mail runs.

Figure below displays the running time of the algorithm which is the second primary issue I worked on. The running time calculates how much time it took for the training model to build and new label to get classified and labelled as Inbox or Spam mail.

![Time to predict the label and running time of the algorithm](https://via.placeholder.com/150)

Figure 15: Running-time for prediction and total algorithm.

Running time of the algorithm also matter since one would not wait for the model to run for long time and wait for the mails. Hence, I gave equal importance to how much time the algorithm is taking to perform. The running time of algorithm initially was taking more than two minutes because of lesser optimal code. I fine-tuned the code and implemented different approach for finding the features and classifying the mail. This model as depicted I Figure 15 above takes approximately 8 seconds to predict the label and almost 20 seconds for total runtime of the algorithm.
Figure 16: Word cloud of all my features.

As seen in Figure 16, we can determine that word “please” has the maximum term frequency since it is bigger in size followed by “busy”, “mail” and other. Also it helps me understand how will a new mail gets classified. I built this word cloud using R, another programming language used for data analysis using SnowballC and wordcloud package [13]. This calculates count of its occurrence of each word. Also it has function ‘random’ which randomly orders the word in different axis and color. The image below displays the word cloud of Inbox mails and size is directly proportional to count of occurrence.

Figure 17: Word cloud of Inbox mails.
In my Figure 17 (inbox mails dataset), we notice that words such as “database”, “hour”, “final”, “Schedule” occur more number of times. In Figure 18 (Spam mails), we notice that “free”, “company” occur more number of times. As expected, since most of the spam mails contains these types of noise words.

11. ADVANTAGES

Though the existing model is very powerful and is majorly used in the present era but there research undergoing in this domain since researchers believe there are few advantages using this. Few of the major advantages include very little management of the model. Rule based engine needs huge maintenance and are to be continuously updated or else it will harm its functioning. But with self-learning email processing model using Machine and deep learning, since it learns on its own there is no necessity to maintain it.

Secondly, usage of Machine learning in this domain can give accurate results in quicker time since it doesn’t have to deal with huge rule based engine containing large number of rules.

Though there are few flaws such as dealing with unconventional mails which this model might not function due is inadequate number of trained features but this can be resolved by training appropriately and using strong algorithms such as Artificial Neural Network.
12. CONCLUSION

Machine learning algorithms are used in various domains and this attempt of implementing Machine learning approach in email processing proves that it could be used in this domain as well since results are both accurate and swift which the primary goal of any email processing system is. With advancement in technology and research in this domain, this technique is feasible.

The results state that calculating Term frequency and performing N-gram over this textual data proves to be of heavy advantage since it helps the trained model to understand the context of each mail and classify based on that. Also, we observe that text preprocessing plays the most crucial role in this project since we need to have list of accurate and appropriate list of features in our trained model. These are the features which would add some weight to the final classification results. One can’t afford to have irrelevant features in the list since it would tamper the classification or overfits. NLTK packages have provided required lists and packages which I used in my project to preprocess the data. The results obtained were accurate and with minimal delay in receiving the mail.

13. FUTURE WORK

There are few tasks that can be done over this which would improve its performance as well as add new add-ons to the project. Firstly, I wish to make this model as a self-learning model which would takes each new mail which is classified into the training model after classification process. I wish to include a user intractable interface where can select “useful”/“not useful” if the mail is correctly classified/incorrectly classified respectively. All the mails which are correctly classified and if user selects “useful” for those mails will be appended to the list of features which can be trained. So this way the machine self learns and improves its accuracy and results.

Second add-on I wish to do in future would be train this model for other mail labels as well such as “Updates”, “promotions” (the feature which Gmail provides) by appropriately training the data.

REFERENCES


