SPAM CLASSIFICATION USING NATURAL LANGUAGE PROCESSING

Billah Syed Mashkur
Dept. Computer Science
International Islamic University
Gombak, Selangor, Malaysia
mashkurbillah@outlook.com

KM Zubair
Dept. Computer Science
International Islamic University
Gombak, Selangor, Malaysia
contactkmzubair@gmail.com

Tasnim Rafia
Dept. Computer Science
International Islamic University
Gombak, Selangor, Malaysia
tasnim.rafia777@gmail.com

ABSTRACT:
Communication plays a critical role in today's new age of digitalization, whether it be formal or informal communication. It has become more common to use email or other messaging platforms. This growth in electronic contact resulted in an exponential rise in the number of emails that are spammed. Spam constitutes 49.7% of the emails received. The number of global email users will be over 4.1 billion by the end of 2021, with more than half of the world's people using email in January this year. Our objective is to detect unsolicited messages with commercial or pornographic content from the stream of a user while allowing other valid messages to pass through.

KEYWORDS:
Regular Expression, StopWords, Porter Stemmer, Bag of Words, Naive Bayes Classifier, Accuracy Score, Spam Detection Model.

1. INTRODUCTION:

The modern web's social networks allow users to interact with each other by sharing their concerns in the form of email or other text messages. Such online communications, though, could be spam messages that do not have beneficial information. Every day, we get advertising spam messages. The aim of these spam messages is to support low-rated content or services or to promote less well-known items on the market. In this project, we will try to create a model that acts as a spam classifier and will actually classify the spam messages that we usually get in our email or other platforms.

2. RELATED WORK:

In order to detect link-based site spam messages, Davison [1] used the decision trees algorithm. He identified and removed connections between pages that, for purposes other than merit, were present utilizing heuristics to drop internal ties. SVM was used by Drost et. al. [2] to identify site spam with functionality dependent on content. By disclosing the use of groups of intrinsic and relational attributes, they proposed the issue of detecting connection spam and offered a solution for producing training data.

For the identification of spam texts, Mishne et. al. [3] used machine learning strategies and likelihood distributions over strings. The language models used in the blog article, the letter, and the pages connected with the messages are contrasted. The knowledge corpus from the same package was used by Bhattari et. al. [4] to identify spam comments from blogs and pages.

3. TECHNICAL BACKGROUND:

We've used NLTK libraries to create our classifier and the most remarkable ones are mentioned below.

3.1 Understanding the dataset:

Basically, the first column of the dataset named label specifies whether it is spam or ham where ham is just a category that indicates that the message is not spam. In the second column, we have our message. Based on this particular dataset, we will try to create a model and then try to predict whether any message is ham or spam. In our code, we are using 'pandas' for reading this dataset.

3.2 Data Cleaning and Preprocessing:

We didn't have to go through all of the words, instead, we focused on some words that increase the possibility of the message becoming spam. Firstly, we lowered up all the words in a sentence. For data pre-processing, we imported the 're' library that is basically used for regular expression. Then we implemented 'nltk' library, the 'stopWords' library, and the Porter Stemming algorithm.

Regular expression:

Regular expression is an instruction given to a function about whether and how to fit or replace a string sequence. Using 're', we tried to remove all the characters except small a-z and capital A-Z and replace those places with blank.

StopWords Corpus:

Stop words are words that don't bring much meaning to a sentence. They can be easily skipped without losing the essence of the sentence and filtered out before or after the collection of natural language data. For our dataset, words like ‘an’, ‘a’, ‘to’, ‘the’ etc. are of no use and we used ‘stopwords’ to get rid of these irrelevant words as ‘stopwords’ has all these kinds of words.

Porter Stemmer:

The Porter stemming algorithm is a method for extracting the commoner morphological endings or to reduce the inflection in words to their root forms from words in English. We’ve used porter stemmer for stemming to help us find the base root form of the word.

3.3 Bag of Words:

Bag of words is a document matrix with respect to the words. It is a technique for extracting functionality from text records. These characteristics can be used for teaching machine learning algorithms. In our training package, it generated a vocabulary of all the special words that appear in all the texts. We’ve imported the CountVectorizer library which is present inside sklearn.feature extraction and then used fit_transform to our corpus to get the most frequent columns or words.

3.4 Naive Bayes Classifier:

Naive Bayes is a straightforward learning algorithm that uses the Bayes law with a firm assumption that the characteristics are independent events. This model works pretty well for spam classification. Basically, it’s a classification technique that works completely on probability. We’ve used the MultinomialNB library to implement naive bayes which work for multiple classes.

3.5 Confusion Matrix:

The confusion matrix gives us a 2*2-dimension matrix which will tell us how many numbers of elements are correctly predicted. When executed, it gave us a 2*2 matrix and the diagonal elements that will visualize our actual output and our predicted output.

3.6 Accuracy Score:

For assessing classification models, accuracy is one metric. This function measures subset precision in multilabel classification. We’ve imported the accuracy score library from sklearn.metrics to check the accuracy of our model. Being executed, our model got an accuracy of 98% which is great.

4. USED APPROACH:

In our study, we have shown that spam messages can be identified with the right set of features that catch various characteristics of valid messages in order to separate them from spam messages. Our strategy is to eliminate words that contain discontinuous text first, and we refer to words that contain an increased number of punctuation marks, white spaces, capital letters, non-ASCII characters, digits, and new lines through discontinuous text. To implement our Spam Detection Model properly, we've used Regular Expression, Stopwords corpus, Porter Stemmer, Bag of words, Naive Bayes Classifier, and accuracy score.

5. METHODOLOGY:

Spam classification using natural language processing might be narrated with certain principles (e.g. training the root words with algorithm). To explicitly define the process of the project, we started with importing libraries. We proceeded to the next step with leaving the less important words applying Stopwords. We grabbed the rest of the words to stem until the root phase and extracted with PorterStemmer, joined them back. We utilized countvectorizer in order to look for similar root words for the sentimental analysis purpose. We splitted our dataset into 80% : 20% and set them as training and testing dataset. Later we chose MultinomialNB (multinomial naive bayes) algorithm as one of the most suitable ones for supervised learning with predefined classes (e.g. text classifying or spam filtering etc). Specifically, we implemented the 'bag of words' method as it is efficient enough that allows flexibility to personalize the dataset. The model enabled achieving a significant reliability on prediction problems like language modeling, documentation, or spam classification.

The procedure might be described with the flowchart in figure 5 in a nutshell.

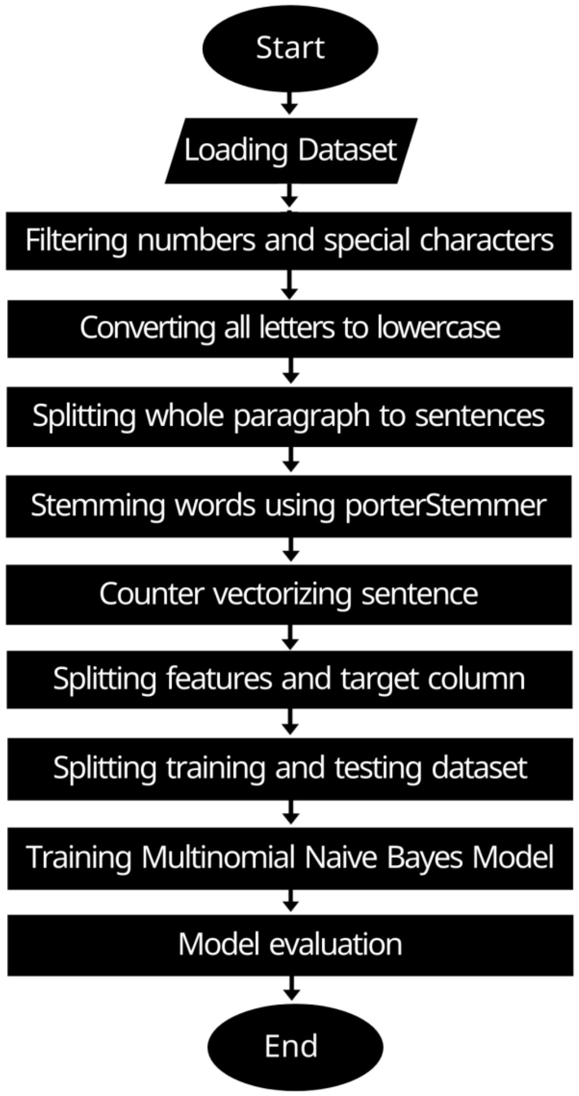
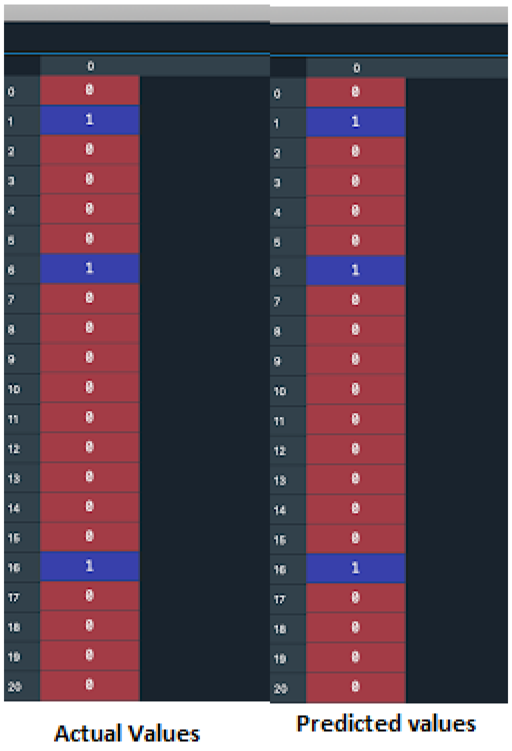

Start
Loading Dataset
Filtering numbers and special characters
Converting all letters to lowercase
Splitting whole paragraph to sentences
Stemming words using porterStemmer
Counter vectorizing sentence
Splitting features and target column
Splitting training and testing dataset
Training Multinomial Naive Bayes Model
Model evaluation
End

Figure: 5

6. RESULT ANALYSIS:

To evaluate our results, we proceed and come up with some of the evaluation criteria as follows:

- Confusion Matrix
- Accuracy
- Precision
- Recall
- F1-Score
- Other Comparison



	0		0
0	0	0	0
1	1	1	1
2	0	2	0
3	0	3	0
4	0	4	0
5	0	5	0
6	1	6	1
7	0	7	0
8	0	8	0
9	0	9	0
10	0	10	0
11	0	11	0
12	0	12	0
13	0	13	0
14	0	14	0
15	0	15	0
16	1	16	1
17	0	17	0
18	0	18	0
19	0	19	0
20	0	20	0

Actual Values

Predicted values

Figure: 6.1

After training the model with Naive Bayes application, we get two variables which are the actual values and the predicted values (y_test, y_pred). For comparing weather the predicted ham/spam values match with the actual ones and give us correct answers we may look at the figure 6.1 where the detection of predicted spams(value = 1) exactly matches with the the actual given set of spams in the actual test data for the first 20 sentences of the test dataset.

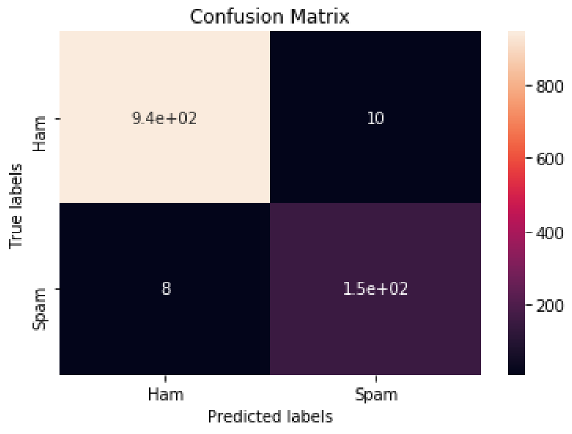


Figure: 6.2 (a)

We have taken a total of 1115 values or sentences as our test data to be marked as spam or ham. For comparing all the 1115 values predicted correctly or not we used a confusion matrix which is a specific table layout that allows visualization of the performance of our algorithm (Figure 6.2(a)). By observing the given table it can be noticed that, out of the total 1115 values 945 sentences are predicted correctly as hams (True Positive) whereas on the other hand 152 sentences are correctly predicted as spams (True Negative). On the contrary, a number of only 10 sentences are incorrectly predicted as hams (False Positive) whereas only 8 sentences are incorrectly predicted as spams (False Negative).

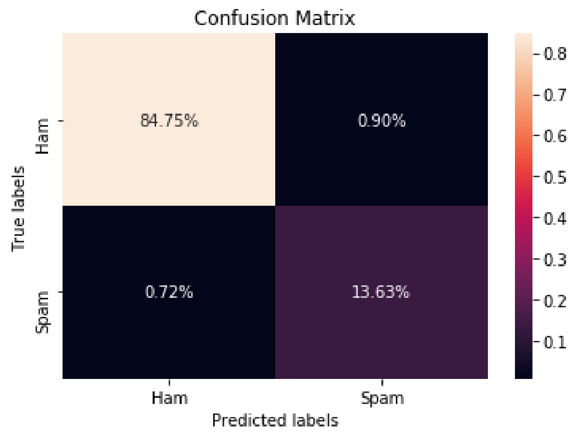


Figure: 6.2 (b)

As for the percentage is concerned, it is clearly seen from the figure above (figure 6.2(b)) that summation of both true positive and true negative percentages give us the total percentage of accuracy which is shown below:

Total Prediction with correction should be:

= TP + TN

= (84.75% + 13.63%)

= 98.38% (Accuracy)

Therefore, 98.38% of the sentences given in the test data are predicted accurately without any mistake and this gives us the accuracy of the model that could also be calculated by implementing the following python code which outputs the same accuracy:

```
From sklearn.metrices import accuracy_score
accuracy_score(y_test, y_pred)
#98.38% accuracy
```

Hence, it proves that implementing the Naive Bayes Model in order to predict multiple categorical values is a better choice to use with a high percentage of accuracy.

Precision	**Recall**	**F1-Score**
0.94 Or, 94%	0.95 Or 95%	0.94 Or 94%

Figure: 6.3

Precision speaks about how precise our values are out of the anticipated positive, how many of them are truly positive whereas recall calculates how many of the Real Positives our model catches by marking it as positive (True Positive) and for both aspects, the higher the results mean the higher possibility of the used model to be fitted for prediction purpose. In our model, we have a precision score of 94% and a recall score of 95% (Figure: 6.3) which are very impressive. Moreover, our model's F1-Score concludes at 94% which is again an excellent value where the relation between Recall and Precision is well balanced.

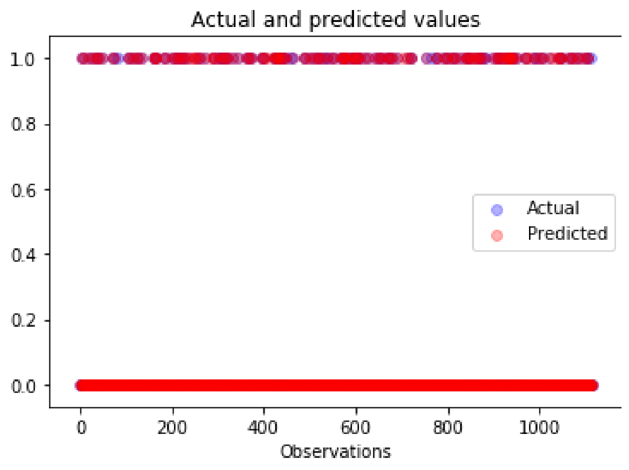


Figure: 6.4

By performing a scatter plot operation, we come up with the above result (Figure: 6.4) where it demonstrates a relation between the actual (blue plots) and predicted values (red plots) in order to observe and point out the dissimilarities/similarities between them. By keeping in mind that there are only two values to be concerned (1=spam Or 0 = ham), it is seen that the predicted values are pointed to the actual values where a few have been plotted near the actual ones which means only a few sentences were predicted incorrectly out of 1115 sentences in our test data which is true. Moreover, we can also notice that the amount of 0's (hams) are predicted more compared to the amount of 1's (spams) which is also correct, as the original ones have the same pattern of distribution of the values.

Note that at the 0 level (ham detection) both actual and predicted values are present at the same points and are overlapped with each other which may cause it to show only a single red line.

7. ERROR ANALYSIS:

In terms of analyzing the error, most of the things have been clarified by analyzing the results that we already get from our model before. Nonetheless, by observing the confusion matrix with percentages (*Figure: 6.3 (b)*), it could be calculated that only a total of 1.62% by performing the following error rate evaluation:

= FP + FN

= (0.9% + 0.72%)

= 1.62%% (Error rate)

From the above calculation it is clear that an error with a lower rate is not going to harm the model in terms of doing this type of categorical predictions. Having an error rate that is less than 2%, it can be said that the Naive Bayes model is one of the best models that can be used in dealing with spam classification problems.

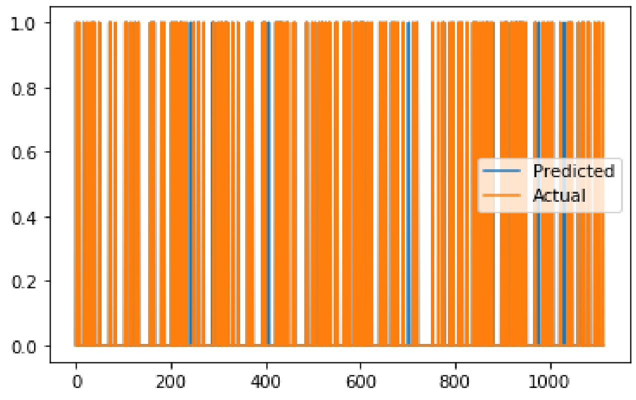


Figure: 6.5

As far as the 2- dimensional standard graph is concerned, the visualization of the actual detections and the predicted detections have been plotted together (Figure: 6.5), and they almost match with each other which can be considered as the same where the dissimilarities are hardly noticed. There is no significant gap or space between the two lines (Actual vs Predicted), moreover, a few predicted values (plotted in blue color), do not predict correctly compared to the actual ones (plotted in orange color) which is why they are not overlapped with the real values and have been exposed on the graph.

Therefore, by plotting this 2D graph we can conclude that after doing predictions, the spam detections are almost the same as the actual detections with a little error or unsimilarity which is even hardly noticed or barely seen.

8. FUTURE WORKS:

There could be some of the possible future works that can be implemented for further improvements. Although our model performance is excellent with an accuracy of 98.38%, there could be some additional works that may help improve the model:

- For future works, maybe we can take a bigger dataset with more balanced features. A 50-50 spam-ham distribution features would allow to be a more precise dataset to work with.
- We could also take more than 2500 words as max_features(most frequently occurring words) inside the CountVectorization() function at the time of creating the bag of words model. This may help to get more samples and ideas on the frequency of words and may boost the prediction accuracy while the complexity of the program may grow higher.
- Word lemmatization could also be used in the future for better stemming purposes. Because lemmatization can identify the root of a word including the POS features of it, assisting in recognizing the root term more unambiguously.

9. CONCLUSION:

We proposed an integrated approach in this paper to identify a text message as spam or valid. The combined approach to creating our spam detection model includes regular expression, stopwords corpus, porter stemmer, bag of words, Naive Bayes classifier, accuracy score, etc. The integrated solution provides greater precision than the implementation of any of these methods alone. We have also defined numerous collections of expressions, terms, and other features here that can indicate that a message is a spam or ham.

In the future, we plan to make our model's application work in real-time. We also intend to expand the dataset that is used for both teaching and research. This helps to verify if we are having more precision. We would like to use word lemmatization for better stemming purposes and might also take initiatives to incorporate crowdsourcing into our work.

REFERENCES:

1. Davison B. D., "Recognizing Nepotistic Links on the Web". In AAAI 2000 Workshop on Artificial Intelligence for Web Search, 2000, pp.23-28.

2. I. Drost and T. Scheffer., "Thwarting the nigritude ultramarine: Learning to identify link spam". In ECML'05 Proceedings of the 16th European Conference on Machine Learning, 2005, Berlin, Germany, pp.96-107.

3. Gilad Mishne, David Carmel, and Ronny Lempel, "Blocking blog spam with language model disagreement". In Proceedings of the First International Workshop on Adversarial Information Web (AIRWeb), Chiba, Japan, May 2005, pp. 1-6.

4. A. Bhattari and D. Dasgupta, "A self-supervised Approach to Comment Spam Detection based on Content Analysis". In International Journal of Information Security and Privacy (IJISP), Volume 5, Issue 1, 2011, pp. 14-32.