# COMP 472 Project 2 Winter 2019

Daanish Rehman
27027753 daanish764@gmail.com

## 1 Introduction

Given a set of ham and spam emails, a naïve bayes classifier model can be built to detect a new email as either spam or ham. Naïve bayes is a conditional probabilistic model that computes the probability that an event occurred given a set of features. The naïve bayes model can be extended to email spam classification. In the case of spam detection, the features would be the bag of words that are contained in the email. Therefore, the equation in figure 1 can be simplified to detect P(spam|a) and P(ham|a) where a are the bag of words. This simplistic model assumes conditional independence among words.

$$P(b \mid a) = \frac{P(a \mid b)P(b)}{P(a)}$$

***Figure 1***

As such, this allows us to simplify the equations to

$$P(spam|words) = \frac{P(words\ |spam)P(spam)}{P(words)}$$

$$\mathrm{P(ham|\ words)}\ = \frac{\mathrm{P(words|ham)P(ham)}}{\mathrm{P(words)}}$$

Whichever probability is higher is a good predictor of whether the email is spam or not. The equation can also be modified further. Since P(words) is same for both in the comparison, it can be removed. Furthermore, if a word has a probability of 0, it would result the output of the equation yielding to 0. In order to prevent this, 0.5 smoothing is used to add a small probability to words that appear in one category and not the other. Furthermore, since multiplication of small number can result in extremely small probabilities, this can yield a comparison that is insignificant and cause underflow. To address this issue, instead of multiplying the probability of each word in the bag of words, it is better to take the log of the probabilities and compute the sum for the comparison. The naïve bayes model that is used to classify emails is shows in Figure 2.

$$H_{NB} = argmax_c(P(c_i) + \sum \log(P(w_k|c_i)))$$

***Figure 2***

## 2 Baseline Experiment Analysis

**Table 1.** Confusion Matrix of the Baseline Experiment

| | Actual Spam | Actual Ham |
|---|---|---|
| Predicted Spam | 357 | 11 |
| Predicted Ham | 43 | 389 |

**Table 2.** Baseline Experiment Results

| | Accuracy | Precision | Recall | F1-Measure |
|---|---|---|---|---|
| Result | 0.9325 | 0.9701 | 0.8925 | 0.9297 |

From the results shown in Table 2, the accuracy of the model is 93.25%. Furthermore, the percentage of instances labelled as '*Spam'* which are correctly classified by the Naïve Bayes classifier is around 97; the percentage of instances which are correctly classified in the '*Spam*' label is around 89; and the percentage of instances which are correctly labelled in the overall test set is 93. To be more concrete, from the results shown in Table 1, the Naïve Bayes classifier has correctly classified 357 emails in the '*Spam*' label and 389 emails in the '*Ham*' label. Overall, the classifier has correctly classified 746 emails in a set of 800 emails. In this bag of words model, the Naïve Bayes classifier has misclassified 11 emails in the '*Spam*' label and 43 emails in the '*Ham*' label. From the evidence in the tables, the classifier makes less mistakes in predicting emails which belong to the '*Ham*' label than in predicting emails which belong to the '*Spam*' label.

## 3 Stop Words Filtering Experiment Analysis

**Table 3.** Confusion Matrix of the Stop Words Experiment

| | Actual Spam | Actual Ham |
|---|---|---|
| Predicted Spam | 357 | 13 |
| Predicted Ham | 43 | 387 |

**Table 4**. Stop Words Experiment Results

| | Accuracy | Precision | Recall | F1-Measure |
|---|---|---|---|---|
| Result | 0.9300 | 0.9649 | 0.8925 | 0.9273 |

From the results shown in Table 3, the accuracy of the model is 93%. The percentage of instances labelled as '*Spam'* which are correctly classified by the Naïve Bayes classifier is around 96; the percentage of instances which are correctly classified in the '*Spam*' label is around 89; and the percentage of instances which are correctly labelled in the overall test set is 93. To be more concrete, from the results shown in Table 4, the Naïve Bayes classifier has correctly classified 357 emails in the '*Spam*' label and 387 emails in the '*Ham*' label. Overall, the classifier has correctly classified 744 emails in a set of 800 emails. In this bag of words model, where a set of provided stop words are omitted, the Naïve Bayes classifier has misclassified 13 emails in the '*Spam*' label and 43 emails in the '*Ham*' label. From the evidence in the tables, the classifier makes less mistakes in predicting emails which belong to the '*Ham*' label than in predicting emails which belong to the '*Spam*' label.

## 4 Word Length Filtering Experiment Analysis

**Table 5.** Confusion Matrix of the Word Length Filtering Experiment

| | Actual Spam | Actual Ham |
|---|---|---|
| Predicted Spam | 354 | 11 |
| Predicted Ham | 46 | 389 |

**Table 6.** Word Length Filtering Experiment Results

| | Accuracy | Precision | Recall | F1-Measure |
|---|---|---|---|---|
| Result | 0.9288 | 0.9699 | 0.8850 | 0.9255 |

From the results shown in Table 5, the accuracy of the model is 92.88%. The percentage of instances labelled as '*Spam'* which are correctly classified by the Naïve Bayes classifier is around 97; the percentage of instances which are correctly classified in the '*Spam*' label is around 89; and the percentage of instances which are correctly labelled in the overall test set is 93. To be more concrete, from the results shown in Table 6, the Naïve Bayes classifier has correctly classified 354 emails in the '*Spam*' label and 389 emails in the '*Ham*' label. Overall, the classifier has correctly classified 743 emails in a set of 800 emails. In this bag of words model, where a set of words which do not satisfy the word-length condition are omitted, the Naïve Bayes classifier has misclassified 11 emails in the '*Spam*' label and 46 emails in the '*Ham*' label. From the evidence in the tables, the classifier makes less mistakes in predicting emails which belong to the '*Ham*' label than in predicting emails which belong to the '*Spam*' label.

## 5 Comparison & Discussion of The Three Experiments

As discussed in the previous sections, the Naïve Bayes classifier has correctly labelled 746 emails by using the baseline model, 744 emails by using the stop-words model, and 743 emails by using the word-length model. Naïve Bayes classifier implemented by using either the stop-words model or the word-length model would result in a decrease in accuracy (93.00 and 92.88 percent accuracy, respectively) with respect to the results obtained by applying the baseline model (93.25 percent accuracy). When precision and recall are equally important, the F1-measure has a percentage of 92.97 (in Table 2), 92.73 (in Table 4), 92.55 (in Table 6) for the baseline model classifier, stop-words model classifier, and word-length model classifier, respectively. In evidence, the best classifier for the prediction of Ham/Spam emails is the one implemented using the baseline model.

When using the stop-words model, there is a slight decrease in the accuracy, recall, and f1 measure. The difference is too small, and therefore it is not statistically significantly to conclude anything about the stop-word model. However, it is expected for a model that ignores stop word to perform better. Furthermore, it is expected that better statistical measures are reached for accuracy, and precision are obtained. The reason that the stop-word model did not perform better could be because the training has disproportionate stop words in one category over the other. A script was

written that checked the model.txt file and computed the frequency of stop words in each category and the probability of every stop word in ham and spam emails. From the result, out of 198 stop words provided, 162 stop words were used in the training set. Out of 162 stop words, 100 stop words had higher ham probabilities than spam probabilities. In other words, stop words occur more frequently in the HAM training set. Only 64 out of 162 stop words has higher SPAM probabilities. Furthermore, the total occurrence of stop words in ham and spam was 196523 and 169317. To clarify, a stop words occurs was removed in ham category 16% more than in the spam category. This clearly shows a higher amount of stop words in the ham category skewing the training set for stop word removal and not having the desired increase in performance.

When using the word-length model, the percentage of instances labelled as '*Spam*' which are correctly classified by the Naïve Bayes classifier has decreased with respect to the result obtained from the classifier implemented using the baseline model. The accuracy of model decreased from 93.25 to 92.88%. The difference is too small, and therefore it is not statistically significantly to conclude anything about the word-length model. However, this decrease is unexpected because removing very short and very long words should have resulted in better performance. A script was written to check the check which words were removed from the baseline model and what their probabilities were. It is found that 19090 unique words were removed from the model because they were of almost 2 characters or at least 9 characters in length. Out of 19090 words, 14922 words had higher SPAM probabilities. This could be problematic because it would result in spam words being ignored in model and result in spam emails being missed classified as ham. Furthermore, the total number of occurrences of ham words occurrences and spam word cooccurrences that were removed are 177429 and 300535 spam words. To clarify, a word was removed in spam 1.69 times more often than in the ham category. This causes the model to not detect a spam email when it is actually spam resulting in lower performance.

Overall, comparing the results obtained in the three experiments, the Naïve Bayes classifier has better performance in the baseline model. In a test set where the '*Ham*' and '*Spam*' instances are of equal weight (i.e. 400 emails are labelled as '*Ham*' and 400 emails are labelled as '*Spam*'), the classifier makes more correct predictions for '*Ham*' emails than it does for '*Spam*' emails. In the baseline experiment, it can be seen from Table 1 that the classifier has accurately predicted 389 '*Ham*' labelled emails. In the case of '*Spam*' labelled emails, the classifier only managed to make 357 accurate precisions on a scale of 400. The cause for decrease performance can actually be that stop words removal and word length restriction is removing a disproportional amount of probability from one category of another. Removing a disproportional amount of ham probability will lead to a miss classification of ham email as spam. Likewise, a disproportional amount of spam probability will lead to a miss classification of spam email as ham. Also another reason can be all the noise in training set. For instance, words like "aaacl", "bsvzknnttzbqgkrfuh", "sgpgegahjlzj", and many other that are not words in English. There are so many of these bogus words with very little probability, the probability adds in a naïve bayes classification. Too much noise can also cause misclassification.

## 6 Difficulties

One of the major difficulties encountered in the project was to attempt to improve the classifier. This was not possible because not enough flexibility was given to enhance the classifier. The tokenization string required that certain words to extracted and placed into the model. Even though some of these words were not even actual words, they contributed to the probability of an email not containing that word at all. Also, the use of natural language processing libraries would have made it easier to run classification on the dataset and extract more pertinent data. The quality of the stop words was also fixed and more appropriate could have been used to better the results. For instance, one of the stop words provided was "x-mailman-version" which never appeared in the testing set. This would not even be considered stop word in English.

Furthermore, the training set created difficulties in building a good model. Since the training set had more occurrence of spam words, skewed the model. The model then had lower accuracy in the stop word removal and word length restriction. This causes an unexpected decrease in accuracy which posed challenges and inspired further investigation into the reason behind the result. Even though the results of the three experiments are different enough

to conclude anything. Ideally, we expect that accuracy would have been higher in stop word removal and word length restriction. Discovering the reason behind this was the greatest challenge in the project.

## 7 Investigations/Questions

If there were enough time, it would be amazing to run the experiment run a natural language processing library. One of the issues with the current model is that there is too much noise and useless data. Words that are not even words in English are part of the model. The NLTK library can check whether a word is really a word in the English dictionary before adding it to the model and inserting excess probabilities that will only skew the result.

Furthermore, the NLTK library comes with a list of English stop words that are more appropriate in the analysis. A comparison between the two approaches could really reveal the difference of a better model. A statistical analysis between the results can show how much of an improvement or loss in the accuracy has occurred.

Furthermore, it would be great to run the email classification through a different classification approach rather than a bag of word approach. An n-gram model can possibly be more appropriate for spam detection. For instance, common phrases like “you could win a million dollars, click this like below!” would have the same probability as if these were scrambled and placed out of order. The probability would be the same. A comparison between the n-gram model could reveal improvements that can be made in spam detection. By taking order of words into consideration, the detection can be enhanced. A statistical analysis can be more interesting too see the difference in the performance in these two approaches.

## Reference

1. Machine Learning Crash Course,
https://developers.google.com/machine-learning/crash-course/, last accessed 2019/03/01.