Simple Naive Bayes Classifier for Email Classification

Jessie James Suarez

October 14, 2018

1 Introduction

In this assignment, the Naive Bayes classifier was used to differentiate between legitimate messages (also known as ham messages) and spam messages. In addition, a feature selection method called Mutual Information has also been used to further improve the recall and accuracy of the said classifier [1].

As an overview, this report will be comprised of four more sections, the second section will discuss the Naive Bayes Algorithm that was used during the implementation. The third section will tackle in depth how the implementation was done. The fourth section will contain the analysis and discussion of results and lastly, the fifth section will contain the conclusions and the recommendations for future work.

2 Naive Bayes Algorithm

According to Bayes' theorem, the posterior probability of an instance x belonging to class c is

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{1}$$

where x is a vector of words containing $[x_1, x_2, ..., x_v]$, c is the class of the email being spam or a legitimate email, and v is the vocabulary size $|V|$. Further expanding this equation, the resulting algorithm will be:

$$P(c|x_1, x_2, ..., x_v) = \frac{\prod_{i=1}^{v} P(x_i|c)P(c)}{\sum_{c \in C} \prod_{i=1}^{v} P(x_i|c)P(c)} \tag{2}$$

For simplicity, the computation of the class conditional likelihood for a word x_i is done by simply checking the presence or absence of a word which can be formally written as follows:

$$P(x_i|c) = \frac{\sum_{D \in D_c} I(x_i \in D)}{|D_c|} \tag{3}$$

However, during training there may be times where a word x_i may appear only in one class and will not appear in another class. In this case, Equation (2) will entirely collapse since anything multiplied and/or divided by zero, will be zero. To circumvent this problem, Lambda smoothing is used to allocate the probabilities to those who have zero probabilities. The modified class conditional likelihood will now be:

$$P(x_i|c) = \frac{\sum_{D \in D_c} I(x_i \in D) + \lambda}{|D_c| + \lambda|V|} \tag{4}$$

Finally, in order to classify whether or not an input document is either ham or spam, the algorithm will need to generate two probabilities for a new document, one is the probability that it might be spam and the other is that the probability that it might be ham. Whichever has a higher probability, the document will be assigned to that class.

3 Implementation

This section discusses in detail how the Naive Bayes Classifer was implemented for both the simple and improved variants.

3.1 Simple Naive Bayes Algorithm

The implementation of the Simple Naive Bayes algorithm is summarized by the flowchart shown below.

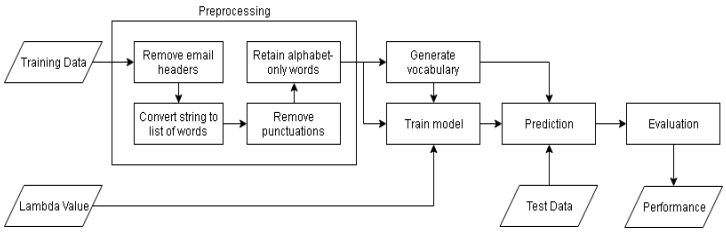

Figure 1: Simple Naive Bayes Flow

3.1.1 Preprocessing

Initally, the training will be loaded to the preprocessing module of the program. For each email in the training data, the email headers are removed and only the body is kept. The removal of email headers was done using the *email* library of Python.

This was done to reduce unnecessary inputs to the vocabulary generation and model training later on.

After the email headers are removed, the body of the email which is a string will be converted to a list of words by treating each token delimited by a space as a word. For simplicity, a valid word is defined as a sequence of symbols containing only alphabets. However, there are some exceptions: tokens ending with a comma or a period are counted as valid words. Therefore, the next step is to remove a trailing comma or period from each token in the list of words.

At this point, the list of words for each email are now preprocessed and it can be easily distinguishable as to whether or not a token is a valid word; The last step in the preprocessing module is to retain these valid words. Furthermore, the retained words will be converted to lowercase since it is assumed that different word cases would likely mean the same thing, thus, making almost no difference even if they were to be treated as different words.

3.1.2 Vocabulary Generation

From the preprocessed output (which is a list of lists of words), the vocabulary is generated by simply retrieving the unique words across all documents and counting how many times they appear in the training data. After obtaining the counts for each unique word, the vocabulary will be sorted in descending order and only a subset of the vocabulary will be used for training and prediction. To be precise, only the top 10,000 words by count will be used.

3.1.3 Model Training

When training the model, the preprocessed output, vocabulary, and lambda value is taken as parameters for training. To make computation easier, each document is represented as follows:

$$D = \begin{bmatrix} y_1 \\ y_2 \\ ... \\ y_v \end{bmatrix} \tag{5}$$

where

$$y_i = I(x_i \in D) \tag{6}$$

This would then give a vector representation of the document D of length v. Each y_i will have a value of 1 if the vocabulary word at index i, (the word x_i), is in document D and will have a value of 0 if otherwise.

Once all documents are converted to their respective word vector representations, all of them are joined together such that the columns are the document vectors and the rows represent whether or not the vocabulary word is in a document. Note that an additional column containing only the parameter lambda will be appended for computation. Thus, the resulting matrix would have a shape of $i \times |D| + 1$.

This resulting matrix would then be used to compute for the conditional probabilities of each word for each class. This is done by simply creating two matrix representations for emails that are in ham and spam, respectively. Then, add values row-wise and then divide it by $|D_c| + \lambda|V|$.

3.1.4 Prediction

After successfully training the model, it will be ready to predict new emails and determine if they are spam or not.

First, the new email will be preprocessed similar to how it was discussed in Section 3.1.1 but this time, it will only be done for a single email. After preprocessing, the email is then converted into its word vector representation. Once it has been successfully converted, Equation 2 will be applied to the vector for both spam and ham classes. Whichever has the higher probability, that will be the resulting class of the prediction.

3.1.5 Evaluation

Model evaluation is done by simply employing the formula for precision and recall where the positive value is spam and the negative value is ham.

3.2 Improved Naive Bayes

The Naive Bayes Algorithm is improved further by adding a feature selection to its steps. Instead of just getting the top words in the vocabulary, Mutual Information is used to measure how much a word would be able to discriminate between different classes [1]. Mostly the implementation remains the same except for the addition of the new section with generates the Mutual Information Scores and using words from the vocabulary with only high Mutual Information scores.

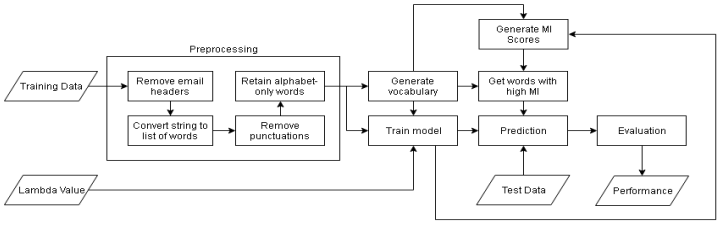

Figure 2: Improved Naive Bayes Flow

3.2.1 Mutual Information

Similar to what is written in Hovald's paper [1], Mutual Information is defined as:

$$\sum_{x\in\{0,1\}}\sum_{c\in C} P(x,c)\log\frac{P(x,c)}{P(x)P(c)} \tag{7}$$

Since the joint probability $P(x,c)$ can be represented as $P(x|c)P(c)$, the equation can be transformed to the following:

$$\sum_{x\in\{0,1\}}\sum_{c\in C} P(x|c)P(c)\log\frac{P(x|c)}{P(x)} \tag{8}$$

Then, the class conditional likelihood of Equation 8 will still use Equation 3 for its computation.

In this implementation, Equation 8 will be used to compute for the Mutual Information scores. The Mutual Information function will take both the vocabulary and the trained model as its inputs. However, for the vocabulary the 100 most frequent words were removed since the most frequent words would likely have a bigger impact on the probabilities [1]. After retrieving all the scores, only those with high Mutual Information scores will be selected when predicting a new email.

4 Analysis and Discussion

4.1 Top Words by Count

During preprocessing and vocabulary generation, it is evident that the top 50 words in the vocabulary with respect to counts are usually stopwords or words that do not convey much meeting. An illustration is shown in the graph below.

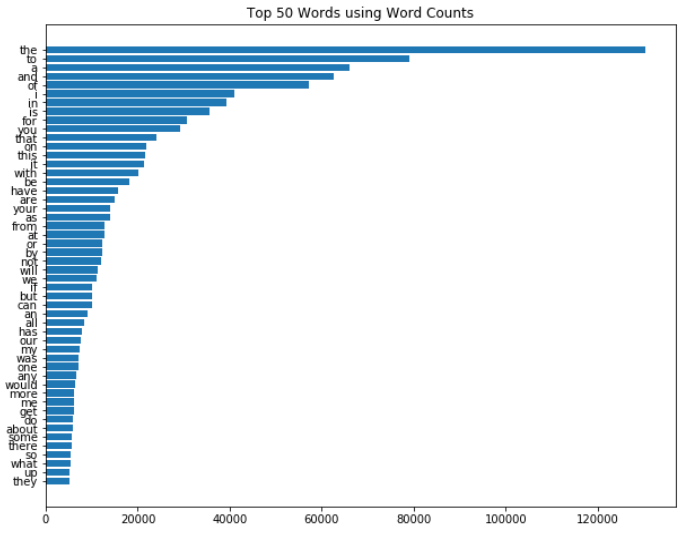

Figure 3: Top 50 Words by Count

4.2 Experimental Results and Analysis

The Naive Bayes classifier has different parameters for its model training; one of them is the lambda value. One problem is the need to decide as to which lambda value should be used for production. In order to look for the best lambda value among a set of possible lambda values $[2.0, 1.0, 0.5, 0.1, 0.005]$, a model was created for each lambda value in order to measure how well it would perform on the test set in terms of precision and recall. The values are shown in the table below.

Lambda	Precision	Recall
2.0	0.862773	0.974489
1.0	0.880770	0.970166
0.5	0.902097	0.961354
0.1	0.919013	0.947578
0.005	0.921893	0.944501

Table 1: Precision and Recall for Simple NBC

Based from the values, there is this trend where the recall is initially high but as the lambda value becomes smaller, it decreases. Conversely, precision started at a lower value and is becoming better as the lambda value decreases. To illustrate this, a line plot is shown below:

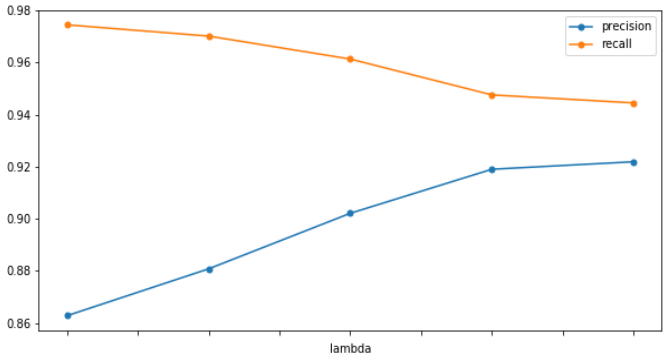

Figure 4: Precision and Recall Trend for Lambda Values

Clearly, while the lambda becomes smaller, both precision and recall draw closer and closer to each other. The trend that the lambda value is introducing may be attributed to the fact that the higher the lambda value, the more importance it gives to unknown words which was why it was able to get a high recall but a price of a lower precision. The same principle applies if situation is otherwise, the smaller the lambda values, the less importance it gives to unknown words thus focusing more on the probabilities that the vocabulary already has, making it less susceptible in classifying emails as false positives.

The point where both of them are close enough

to each other would be the best lambda value since it is at that point where they become stable (refer to the value for lambda at 0.1 and 0.005).

Additionally, using the best lambda value, the model was evaluated on using a decreasing number of words in the vocabulary for prediction. At the start, the top 10,000 vocabulary words by count were selected, and then the top 9,000, gradually decreasing by 1,000 steps up until the top 1,000 words. The graph below shows the result of this experiment:

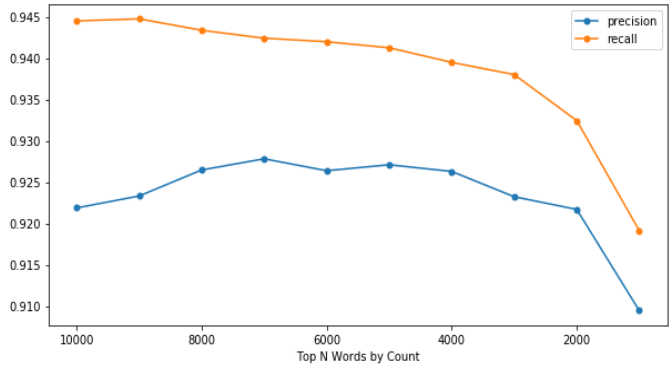

Figure 5: Precision and Recall Trend for Top N Words by Count

Based from the graph alone, it can be deduced that just getting the top words in the vocabulary isn't enough to yield good results. This was why Mutual Information scores are used: to select words that are better representatives of their respective classes [1] as opposed to using simply the top words.

After computing for the Mutual Information scores for the top 10,000 words in the vocabulary, the top 50 MI scores are shown below

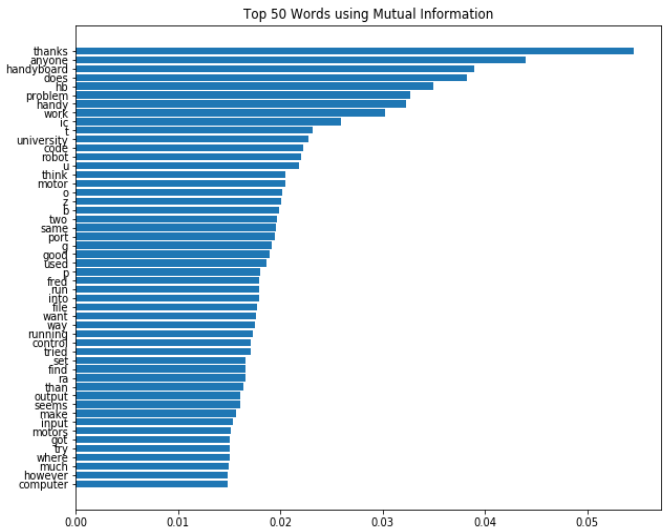

Figure 6: Top 50 Words by Mutual Information Scores

Comparing the top 50 words by count and top 50 words by Mutual Information scores, it's notable that the top 50 MI words had more meaning to it as opposed to the top 50 count words which were mostly composed of stopwords, conjunctions, etc.

Using the top MI scorers, a smaller vocabulary was formed using these words. The smaller vocabulary contains only 200 words. However, despite having a smaller vocabulary, the precision has gone up to 0.990552 while the recall has drastically went down to 0.718022. For a particular scenario where it is more favorable to sacrifice spam emails getting in while there will be significantly less legitimate emails being misclassified as spam, this is an excellent result.

To look at the behavior further, the model was made to predict the test set again given a vocabulary with top 1,000 MI scoring words and decreasing it by 100 steps until the top 100 MI scoring words, the behavior is shown in the line plot below:

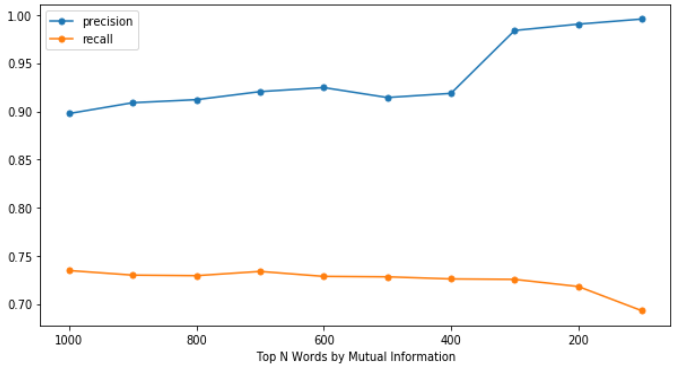

Figure 7: Precision and Recall Trend for Top N Words by Mutual Information Scores

As the number of vocabulary word decreases, the more precision the classifier gains with a significant trade-off with recall. Inspecting this further, most words that had high MI scores were from ham. For both a 100-word and 200-word vocabulary, there were 84 and 169 words, respectively, that had higher ham probabilities as opposed to spam. This might have been the cause why the classifier was good at avoiding classifying emails as false positives.

5 Conclusions and Recommendations

5.1 Conclusions

Based from the foregoing results, (1) it is suffice to say that using even the simple Naive Bayes Classifier, has yielded above average results in terms of actually being able to classify an email as spam or ham. (2) In addition, it also became clear that the preprocessing step is critical when dealing with Natural Language Processing tasks, the results

would vary depending on how much of the data was preprocessed, which words were selected, removed, cleaned, etc. (3) Moreover, selecting which words to include in the vocabulary is yet another important task since it has greatly affected the performance of the experiments as shown extensively in Section 4.

5.2 Recommendations

Despite having yielded above average results, there are a lot of improvements that can be done to this such as: (1) perhaps changing the feature selection method aside from using Mutual Information. (2) Another improvement that is possible is to determine the context of how words are being used. Sometimes words have more than one meaning. For more frequently appearing words, it might be best to be able to distinguish between using a word in a context of a spam email as opposed to using a word in the context of a legitimate email. The information from such a frequently appearing word and its context might be valuable. (3) For methods like these where the best words are to be selected to be part of the vocabulary, it might be best to scientifically determine an optimal threshold such that it maximizes the probabilities of each word within the vocabulary while clearly not allowing them to be overpowered by the uncertainty of unknown words. (4) Lastly, a better way of handling unknown words should also be implemented since in lengthy texts, a lot of information is being lost if the vocabulary will be the sole basis to determine a document's class and the text in the data would not be made for good use.

References

[1] Johan Hovald, Naive Bayes Spam Filtering using Word Position Attributes, *Conference on Email and Anti-Spam*, Stanford University, 2005.