Naive Bayes Spam Filter

Shebna Rose Fabilloren
Department of Computer Science
University of the Philippines Diliman
Quezon City, Philippines
sdfabilloren@up.edu.ph

ABSTRACT

This paper discusses the use of Naive Bayes algorithm in solving detection of spam emails from the 2006 TREC Public Spam Corpus, specifically focusing on implementing several techniques in improving the performance of a naive bayes classifier. Natural language processing(NLP) is used in preprocessing the text from the corpus. Some tweaking performed in the preprocessing stage includes removal of unnecessary features from the text such as stop words, punctuations, and HTML tags. Exploring different variations of lambda smoothing parameters for likelihood computation is also tackled in this study. Moreover, advantages in choosing the top-most important features for the vocabulary size showed efficient use of memory resources without sacrificing classifier performance.

Removal of stopwords showed 0.15% improvement on precision and 0.20% on recall. Bigger vocabulary sizes also show improvement on both precision and recall. Performance of the classifier on different lambda parameters show an inverse relationship such that a higher lambda value results into high precision, low recall while a lower lambda value produces a high recall, low precision.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Naive Bayes, spam filtering, natural language processing

1. INTRODUCTION

Naive Bayes algorithm is a machine learning technique based on Bayes' Theorem that assumes independence among features. It assumes that the presence of a particular feature in a class is unrelated to the presence of another feature. It is one of the most commonly applied types of machine learning algorithms for classification tasks such as spam filtering.

In this project, Naive Bayes algorithm will be used to detect SPAM – an unwanted or unsolicited email. A classifier is used to determine whether an email is spam or ham (not spam). Using a bag of words model, a record of each word's frequency is used to determine the probability of it occurring as spam or ham. If the word occurs more frequently in a spam compared in a ham, then the email containing the word is most likely to be marked as spam.

The initial dataset is split into training data and testing data where both contains spam and ham emails. The training dataset were first preprocessed by removing punctuation marks, newline characters, single quotes, HTML tags, and stopwords. Afterwhich, the contents of an email D in the training dataset is first broken down into words or tokens x_1,x_2,x_3,... x_d. The probability of an email D is equal to the probability of receiving the list of words or tokens x_1,x_2,x_3,... x_d.

The prior probabilities, conditional probabilities or likelihood are first computed in order to find the posterior probability of an email as being a spam or ham. This step is further discussed in the methodology.

2. METHODOLOGY

2.1 Dataset

The dataset used for this project is the 2006 TREC Public Spam Corpus, a dataset for benchmarking spam algorithms, which is composed of 37,822 emails. It is split into a training set, 60% of dataset, and testing set, 40% of dataset. The training set is further divided into two for the spam and ham class labels. Spam training set is composed of 15,019 emails while the ham training set is composed of 7,674 emails. The labels contained in the label file was attached to its corresponding label.

2.2 Preprocessing

The dataset was first parsed by only including the body of the email and ignoring other parts of it. Since punctuation marks are not helpful features, these are removed. Other non-alphabetical characters such as newline characters, single quotes, and HTML tags were removed. However, an additional preprocessing step was added which is removing stop words or words that are too common or holds meaningless information. Performing this reduces the dimensionality

of the data.

Words or tokens generated from the training set are then used as the vocabulary but due to a memory resource bottleneck issue, initially only the top 10,000 most frequent unique words were utilized as the vocabulary. After implementing probability ratio in getting the most informative words, only the top 200 words were used in building the vocabulary.

2.3 Computing for the Prior Probabilities

One of the important parts is computing the prior probabilities for spam and ham which can be used in the later computation of the final spam or ham probability. The formula for solving these is given below:

$$P(w = H) = \frac{Count_{docs}(w = ham)}{Count_{docs}(w = ham) + Count_{docs}(w = spam)} \quad (1)$$

$$P(w = S) = \frac{Count_{docs}(w = spam)}{Count_{docs}(w = spam) + Count_{docs}(w = ham)} \quad (2)$$

2.4 Computing the Likelihood

In order to compute for the likelihood of each words as being a spam or ham, every word in the vocabulary was used to check if it exists for each documents under the spam or ham group. If the vocabulary word exists, it is recorded as its frequency. The likelihood or conditional probability can be calculated using the frequency and dividing it by the number of spam labels or ham labels in the training set. This can be expressed by the following formula:

$$P(x_i|w = H) = \frac{Count_{docs}(X = x_i, w = H)}{Count_{docs}(w = H)} \quad (3)$$

$$P(x_i|w = S) = \frac{Count_{docs}(X = x_i, w = S)}{Count_{docs}(w = S)} \quad (4)$$

However, there are cases where a word from the testing set does not exist in the vocabulary which was built using the training set. In that case the probability of a word occurring given the classification will be 0 which will make the probability of the class given the word also equal to 0. This problem can be solved by introducing a process called lambda smoothing. A lambda value is added to the numerator. For the denominator, the lambda value is multiplied by the number of words in the vocabulary and added to the denominator itself. With this mitigation technique, the formula stated above is redefined to be:

$$P(x_i|w = H) = \frac{Count_{docs}(X = x_i, w = H) + \lambda}{Count_{docs}(w = H) + \lambda|v|} \quad (5)$$

2.5 Classification Process

After computing the prior probabilities and likelihood probabilities, the posterior probability formula for classifying an email can now be used. This is given below:

$$P(w|x_1, x_2, ...x_d) = \frac{\prod_{i=1}^{d} P(x_i|w)P(w)}{\sum_w \prod_{i=1}^{d} P(x_i|w)P(w)} \quad (6)$$

This formula should be performed on both class labels, spam and ham. If the posterior probability of spam is greater than the posterior probability of ham, then the email is classified as spam. However, if the posterior probability of spam is less than or equal to the posterior probability of ham, then the email is classified as ham.

A problem arises when multiplying very small resulting values from the likelihoods. This can result into a floating-point underflow which produces loss of precision, In order to avoid loss of precision during multiplication of the likelihoods, the sum of the logarithm base ten of each multiplicand is taken. This sum is then exponentiated using the base 10 to retrieve the results.

Following the logarithm and exponent rules, we first take the logarithm of both sides:

$$log(P(w|x_d)) = log(\prod_{i=1}^{d} P(x_i|w)P(w)) \quad (7)$$

This results into the the sum of the likelihood probabilities of each word and the class label prior probability:

$$log(P(w|x_d)) = \sum_{i=1}^{d} log(x_i|w) + log(P(w)) \quad (8)$$

To get the final value, exponentiate the sum using the log base 10:

$$10^{log(P(w|x_d)))} = 10^{\sum_{i=1}^{d} log(x_i|w) + log(P(w))} \quad (9)$$

2.6 Probability Ratio in Getting the Most Informative Words

It is important to find which words the classifier identify to be most discriminative in detecting spam email. From the computed probabilities of spam and ham words, the ratio of probability of the word given it's spam over its probability can be computed by using the formula below:

$$\frac{P(x_i|w = S)}{P(x_i|w = H)} > 1 \quad (10)$$

The word is most likely to be spam if the ratio is greater than one. Computed ratios sorted in reverse will be the basis of getting the most informative words in the vocabulary.

2.7 Performance Metrics

It is important to measure the performance of the classifier created in this project. This can be done by setting metrics which will be used as the basis for determining how good the classifier is.

Definition for each performance metrics and related terms are defined as follows:

Performance Metrics	Formula
Precision	TP/(TP+FP)
Recall	TP/(TP+FN)

Precision measures the percentage of predicted spam emails that were correctly classified.
Recall measures the percentage of actual spam emails that were correctly classified.
True positive(TP) is the number of spam messages classified as spam.
True negative(TN) is the number of ham messages classified as ham.
False positive(FP) is the number of ham messages misclassified as spam.
False negative(FN) is the number of spam messages misclassified as ham

3. RESULTS AND DISCUSSION

The main goal of the evaluation is to classify the given benchmarking dataset into two categories (ham, spam) correctly. There is a total of 37,822 emails from the dataset which is then divided into two categories, namely, the training set and testing set.

The following table shows the number of spam and ham training and testing sets:

Dataset	Number
Training Set (Spam)	15,019
Training Set (Ham)	7,674
Testing Set	15,129

Table 1: Number of training and testing sets.

The vocabulary used for this project is based on the frequencies of the unique words from the training set. Only the top 10,000 most frequent unique words served as the vocabulary.

It is also important to evaluate whether the preprocessing steps taken were useful in producing good precision and recall results. Multiple preprocessing techniques were applied on the dataset which includes removing punctuations, newline characters, and single quotes, removing HTML tags, and removing stopwords. Since all of the removed characters are considered noise, not including them in the final preprocessed dataset will improve the performance of the classifier.

Dataset	Precision(%)	Recall(%)
Without stop words	95.68	88.73
With stop words	95.83	88.93

Table 2: Precision and Recall on dataset with stop words vs. without stop words.

It is observed in the Table 2 that the precision drops by 0.15%, and the recall by 0.20%. It can be inferred that some words are less informative than others. Stop words like “is”, “the”, “and”, etc are mostly common in all English sentences and are not helpful in deciding spam or ham so these words have been removed from the vocabulary. Alphanumeric characters are also removed from the emails to further simplify the results. Also, it should be taken note that the algorithm extracted only the body of the email.

Vocabulary Size (Top N by frequency)	Precision(%)	Recall(%)
10,000	90.42	88.63
1000	84.87	93.02
100	81.64	89.10

Table 3: Precision and Recall on Different Vocabulary sizes.

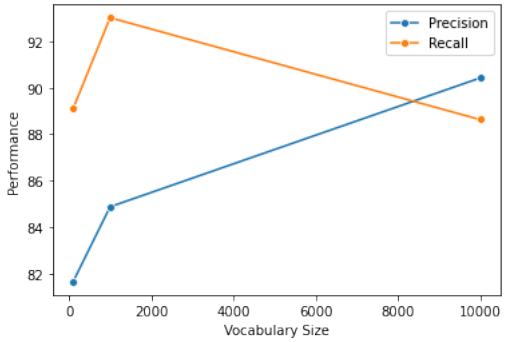

Figure 1: Visual Representation of Table 3

It can be observed that the precision results are comparatively lower in vocabulary sizes 1,000 and 100 than the vocabulary size of 10,000 unique words. The testing set used for this experiment was reduced into 1000 due to time constraints. With the above results, it can be analyzed that precision can be improved with increasing the vocabulary size.

Therefore, from the discussed findings above, it is required to adjust the preprocessed model. It can be concluded that using preprocessing and feature selection can achieve better classification results.

Earlier, it was discussed that lambda smoothing can solve the problem of cases where the probability of a certain word or token is zero. During computation of the likelihoods, two formulas were applied, one without lambda smoothing and another with lambda smoothing. In order to find out whether applying lambda smoothing helps in creating better classification results, a comparative analysis using different lambda values was performed.

Lambda	Precision(%)	Recall(%)
2.0	98.33	81.90
1.0	97.98	84.37
0.5	97.19	85.62
0.1	95.93	88.24
0.005	95.70	89.54

Table 4: Performance results of training data on different lambda parameters.

It can be observed from Figure 2 that $\lambda = 2.0$ generated the lowest recall of 81.90% but has the highest precision of 96.06.

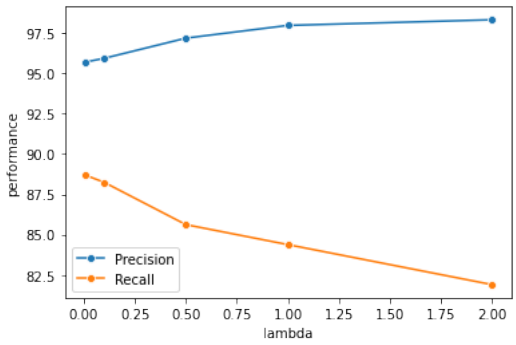

Figure 2: Performance of Precision and Recall on Different Lambda Values

There is an inverse relationship between precision an recall as lambda increases. In cases where, precision metric is more prioritized, a higher lambda value is preferred. However, if having a high recall is more prioritized, then a lower lambda value should be used.

4. CONCLUSION

In this project, it has been shown that Naïve Bayes algorithm is effective in achieving a good performance when applied to spam filtering. Although some factors need to be considered since they influence the performance results of the classifier. In the results discussed above, it can be observed that the classifier's performance depends on the number of vocabulary words. This means that the precision of a classifier can be increased by increasing the vocabulary size. The choice of preprocessing steps taken in the experiments conducted contribute a great role in improving the performance metrics.

The preprocessing stage is the most time-consuming part of the project. This is important in machine learning since a low quality data will result into a low quality model. The project has shown that Bayesian estimates can be extended to avoid zero probabilities on the classifier. Different parameters were also performed and determined which one is best for providing good performance. It can be concluded that using Naïve Bayes for spam filtering can produce good results as long as the number and quality of vocabulary words used can be ensured to be high. Additionally, the preprocessing techniques used can impact the performance results.

5. RECOMMENDATIONS

To further improve performance results of the classifier, more preprocessing technique can be applied, which produces words that may increase the probability values, some of these are ngrams so not only single words are considered, using specific parts-of-speech (POS) tags in order to find the significant parts of speech such as nouns and adverbs, and applying stemming or lemmatization to group together words by their base form. Applying feature selection algorithms that can produce better top performing words can help improve the performance of the classifier without having to sacrifice the large dimensionality of vocabulary data.

6. REFERENCES

[1] Hovold, John *Naive Bayes spam filtering using word-position-based attributes and length-sensitive classification thresholds*. 2005.

[2] Androutsopoulos et al., *Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach*. 2000