AI201 Report 1
Naive Bayes Implementation for Spam Filtering

Mary Nathalie Dela Cruz

Date of Submission: November 24, 2023

1. INTRODUCTION

In the era of digitization, email has become a convenient, fast, and effective way of exchanging information. However, as a result of its growing usage, a vast number of issues exploiting its reach and capabilities have emerged. One major drawback of email communication is the increasing number of spam emails that reach millions of internet users daily. Spam refers to unsolicited bulk emails that may contain unwanted ads, chain letters, hoaxes, scams, and/or fraud. Spam emails not only waste network resources, memory space, and user time due to high volume traffic but they also expose innocent receivers to financial and identity risks from malicious and offensive content. Hence, spam is a serious concern that needs to be addressed and prevented.[1-5]

For the past few years, several countermeasures for identifying and filtering spam have been explored. Most methods that were developed employ knowledge engineering and/or machine learning. Filtering approaches that utilize knowledge engineering are based on a set of pre-established classification rules. However, these approaches may face issues with scaling and may need regular rule updates because of the constantly changing content and structure of spam emails. Because of these challenges, more focus has been placed on machine learning for the creation of spam filters. With machine learning, classification rules are automatically created by an algorithm that learns from patterns identified in the data of a training set of emails. Machine learning methods for spam filtering have been extensively developed to keep up with the constantly changing nature of spam emails [4,6].

One of the most commonly used methods for spam filtering is the Naive Bayes algorithm. Naive Bayes algorithm is known for its simplicity, accuracy, and linear computational complexity. Sahami et al. were among the first to achieve promising results in doing this [5]. However, the Naive Bayes algorithm also has some limitations, such as the assumption of independence among features and the problem of zero probabilities when a word does not appear in the training data. To overcome these limitations, techniques such as lambda smoothing and feature selection have been proposed. Lambda smoothing is a method that adds a small positive constant to the probabilities to avoid zero values. On the other hand, feature selection is a method that reduces the vocabulary size by selecting the most informative features [6]. In this paper, we implemented a Naive Bayes classifier for spam filtering, where we also applied lambda smoothing and feature selection via mutual information. The performance of our classifier on a spam email dataset was then evaluated using precision and recall as metrics.

2. OBJECTIVE

The main objective of this study is to implement a Naive Bayes Classifier that only focuses on the textual content of email data for spam filtering. We aim to address some of its limitations by performing lambda smoothing for handling zero probabilities and feature selection via mutual information for reducing the dimensionality of the dataset. We evaluated the performance of the classifier under different conditions–with and without feature selection and with and without lambda smoothing for smoothing parameters of 0.005, 0.1, 0.5, 1.0, and 2.0. By doing so, we can look into the impact of feature selection and lambda smoothing on the classifier's performance.

3. METHODOLOGY

3.1 Data Collection and Pre-processing

The spam and ham email data used in this project was from TREC06 Public Spam Corpus, a dataset commonly used for bench-marking spam filtering algorithms. The dataset contains 37822 emails, of which 24647 were extracted to be spam and 12910 were extracted to be ham. The email data was parsed to obtain its unique words, where a word was defined to be alphabetic characters separated by a white space in front and a white space, comma, or period at the end. The words were converted to lowercase and punctuation marks, numbers, or special characters were removed. The dataset was shuffled and split into a training set and a test set using a 70-30 partitioning. The training st contains 26283 emails, of which 17278 are spam and 9005 are ham. The test set contains 11264 emails, of which 7369 are spam and 3895 are ham. From the training set, a vocabulary consisting of 73861 unique words was created and the prior probabilities for spam and ham were calculated to be 0.654 and 0.346, respectively. The conditional probability of each word in the vocabulary given that it belongs to a specific class was computed and stored in a dictionary.

3.2 Naive Bayes Algorithm

The Naive Bayes algorithm assumes that the feature variables $(x_1, x_2, ...x_d)$ are independent of one another. Each feature variable x_i represents a word in a vocabulary V with size d whose value depends on the presence or absence of a word in a document D found in a set of documents D_ω belonging to class label ω. The probability that a document

D belongs to the class label ω given the values of its feature variables $(x_1, x_2, ...x_d)$ is represented by the Bayes' formula:

$$P(\omega|x_1, x_2, ..., x_d) = \frac{\prod_{i=1}^{d} P(x_i|\omega)P(\omega)}{\sum_{\omega}\prod_{i=1}^{d} P(x_i|\omega)P(\omega)} \tag{1}$$

where class label ω can either be spam or ham and vocabulary V consists of words in both class labels. We calculated the probability of the presence $(x_i \in D)$ or absence $(x_i \notin D)$ of each feature variable x_i given that the document D belongs to class label ω using the conditional probability formula:

$$P(x_i \in D|\omega) = \frac{\sum_{D \in D_\omega} I(x_i \in D)}{|D_\omega|} \tag{2}$$

$$P(x_i \notin D|\omega) = 1 - P(x_i \in D|\omega) \tag{3}$$

where $I(x_i \in D)$ is a function that indicates that a feature variable x_i is in the document D. Document D is part of a set of documents D_ω with set size $|D_\omega|$. The probability of each class ω occurring in a set of documents is given by the prior probability formula:

$$P(\omega) = \frac{\sum I(D \in D_\omega)}{|N|} \tag{4}$$

where $|N|$ is the total number of documents and $I(D \in D_\omega)$ is function for indicating that a document D is part of a set of documents D_ω.

Equations 2 to 4 were performed on the documents in the training set. The prior probabilities and the conditional probabilities solved for each class label were stored in dictionaries and used as references in solving equation 1 and in predicting the class label of each document in the test set. The probability solved with equation 2 was used when a word in the vocabulary was in any document in the test set while the probability solved with equation 3 was used when a word in the vocabulary was not in any document in the test set. The logarithm of the probabilities was added and then the sum was exponentiated instead of directly multiplying the probabilities in equation 1 to avoid precision loss.

3.3 Performance Evaluation

The performance of the algorithm was evaluated by comparing the predicted labels with the actual labels of the test set. The evaluation metrics used were mainly precision and recall. Precision is the proportion of documents correctly classified as spam among all documents predicted to be spam. Recall is the proportion of documents correctly classified as spam among all documents that are spam. The formulas for precision and recall are as follows:

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

where TP is the count of true positives or documents that are correctly classified as spam, FP is the count of false positives or documents that are incorrectly classified as spam, and FN is the count of false negatives or documents that are incorrectly classified as spam.

3.4 Lambda Smoothing

A limitation of the Naive Bayes algorithm is the occurrence of zero probabilities when a word does not appear in the training data given a class label. To address this problem, lambda smoothing adds a constant to equation 2. The modified formula for the probability of the presence $(x_i \in D)$ of a feature variable x_i given that the document D belongs to class label ω is shown below:

$$P(x_i \in D|\omega) = \frac{\sum_{D \in D_\omega} I(x_i \in D) + \lambda}{|D_\omega| + \lambda|V|} \tag{7}$$

where $|V|$ is the vocabulary size, λ is a positive smoothing parameter, and $I(x_i \in D)$ is a function for indicating that a feature variable x_i is in the document D. In this study, lambda smoothing was applied to the Naive Bayes algorithm, using five different smoothing parameters: 2.0, 1.0, 0.5, 0.1, and 0.005. The precision and recall of the classifier were calculated for each smoothing parameter, and the results were compared. [5]

3.5 Feature Selection

In Naive Bayes algorithm, feature variables were assumed to be independent. This assumption is difficult to meet in reality because features can still be correlated with each other and some features tell more about the class label than other features. To address this problem, we can select the most relevant features to predict the class label. Doing so can reduce redundancy in the vocabulary and improve the performance of the classifier. One method for feature selection is mutual information. Mutual information measures how informative a feature can be about the class label. It can be calculated by the formula below:

$$MI(w, c) = \sum_{x \in x_i} \sum_{\omega} P(x_i, \omega) \log \frac{P(x_i, \omega)}{P(x)P(\omega)} \tag{8}$$

where x_i is a feature variable representing a word, ω is a class label, and $P(x_i, \omega)$, $P(\omega)$, and $P(x_i)$ are the joint and marginal probabilities of the feature variables and the class label which can be calculated using the following formulas:

$$P(x_i, \omega) = \frac{\sum I(x_i \in D)}{|N|} \tag{9}$$

$$P(x_i) = \frac{\sum I(x_i \in N)}{|N|} \tag{10}$$

$$P(\omega) = \frac{\sum I(D \in D_\omega)}{|N|} \tag{11}$$

where $|N|$ is the total number of documents. The mutual information of a word and a class label can be calculated for each word in the vocabulary and each class label. The words can then be ranked by their mutual information scores, of which the top 200 words can be selected as the most informative features. Feature selection was applied to the Naive Bayes classifier in this study for varying smoothing parameters. The top 200 informative words were used, and the resulting precision and recall of the classifier were calculated and compared with the results when the full vocabulary was used. [5]

4. EXPERIMENTAL RESULTS

The performance metrics of the Naive Bayes classifier under different conditions are shown in Table 1, Table 2, and Table 3. Table 1 shows the performance metrics for different parameters of lambda smoothing using the vocabulary of 73861 words obtained from the training set. Table 2 shows the performance metrics for different parameters of lambda smoothing using the reduced vocabulary of the top 200 informative words. Table 3 shows performance metrics for increasing the number of selected top features to be used in the vocabulary. The results indicate that a specific parameter for lambda smoothing can optimize the performance results while the reduced vocabulary of the top 200 most informative words did not improve the performance of the Naive Bayes classifier unless we increase the number of top features selected. However, even if we increase the number of top features, the best performance was achieved by the Naive Bayes classifier when lambda smoothing was done with a 0.1 parameter without the feature selection. This has an accuracy of 0.932, a precision of 0.984, and a recall of 0.911.

Table 1: Performance metrics of the Naive Bayes classifier for varying smoothing parameters using the vocabulary from the training set

Smoothing Parameter	Precision	Recall	Accuracy
0	0.994	0.876	0.915
0.005	0.961	0.927	0.928
0.1	0.984	0.911	0.932
0.5	0.993	0.881	0.918
1.0	0.992	0.856	0.901
2.0	0.990	0.831	0.915

Table 2: Performance metrics of the Naive Bayes classifier for varying smoothing parameters using the reduced vocabulary of 200 words

Smoothing Parameter	Precision	Recall	Accuracy
0	0.999	0.149	0.442
0.005	0.960	0.417	0.607
0.1	0.964	0.469	0.640
0.5	0.965	0.477	0.646
1.0	0.966	0.479	0.648
2.0	0.966	0.476	0.649

Table 3: Performance metrics of the Naive Bayes classifier for varying number of top features for a 0.1 smoothing parameter

No. of Top Features	Precision	Recall	Accuracy
1000	0.964	0.705	0.788
3000	0.968	0.750	0.818
5000	0.977	0.794	0.851
7000	0.982	0.813	0.867
9000	0.987	0.816	0.871

5. ANALYSIS AND DISCUSSION OF RESULTS

5.1 Impact of Lambda Smoothing to Model Performance

As shown in Figure 1, the parameter chosen for lambda smoothing has a significant impact on the performance of the classifier. It is expected that increasing the smoothing parameter adds more weight to each word in the vocabulary including rare words. This lowers the variability of the probabilities, reducing the variance but increasing the bias. Accordingly, the probability distribution of a spam filter with a lower-valued smoothing parameter has higher variance but lower bias compared to that of a spam filter with a high-valued smoothing parameter. Tracing these assumptions back to Figure 1, it is generally evident that lower smoothing parameters lead to higher recall, and higher accuracy but lower precision due to higher variance and lower bias, making the classifier more likely to label a document as spam (more TP and more FP). Meanwhile, higher smoothing parameters lead to lower recall and lower accuracy but higher precision due to lower variance and higher bias, making the classifier less likely to label a document as spam (more FN and less FP). It is thus important to select the smoothing parameter that balances the trade-off between variance and bias. In this case, the most optimal performance that considers all metrics was achieved by the classifier with a smoothing parameter of 0.1.

5.2 Impact of Feature Selection of Top 200 Informative Words

It was initially expected that selecting the top 200 features based on mutual information can improve the accuracy of the algorithm because it can reduce the number of irrelevant features. However, as shown in Figures 2 to 5, the accuracy, recall, and precision did not improve for all smoothing parameter values. It is thus recommended to check how increasing the number of features can impact the performance metrics. Figure 6 shows that increasing the number of features does improve the performance of the classifier. However, selecting too many features can possibly lead to over-

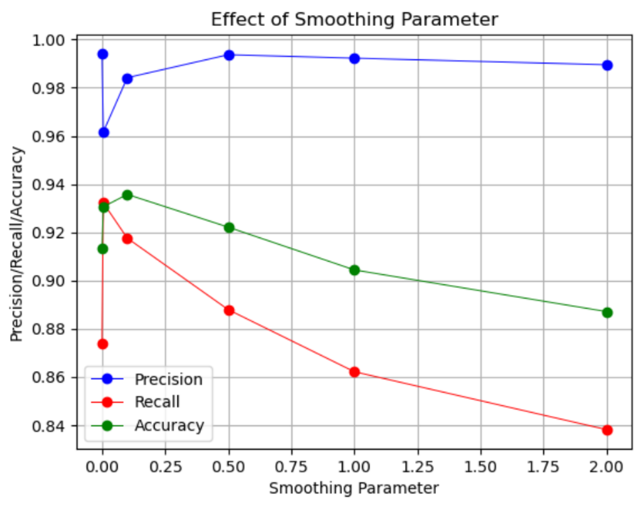

Figure 1: Effect of Smoothing Parameter

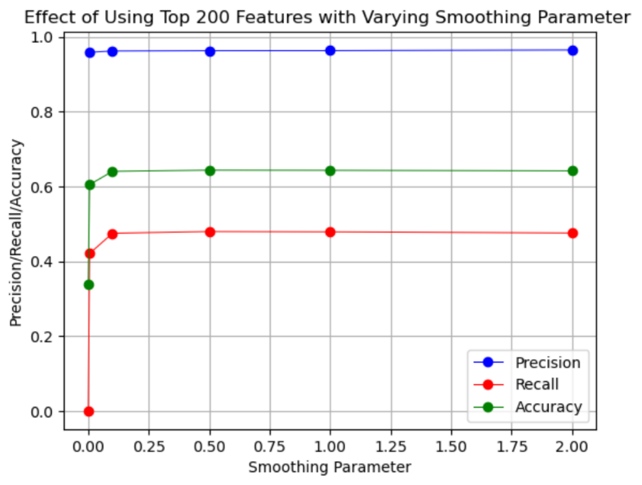

Figure 2: Effect of Using Top 200 Features with Varying Smoothing Paramters

fitting which can reduce the accuracy of the classifier. In terms of computational efficiency, feature selection reduces the time and cost to iterate over a vocabulary as the technique reduces the vocabulary size. Consequently, increasing the number of features also increases the processing time and cost. In this case, iterating over the full training vocabulary takes approximately 22 minutes while iterating over the top 200 vocabulary takes approximately a minute. It is thus important to select the optimal number of features depending on the trade-off between the performance metrics and computational cost.

6. CONCLUSION

In conclusion, the smoothing parameter must be carefully considered in spam filtering. The optimal value of the smoothing parameter will depend on the trade-off between bias and variance where increasing the value of the parameter will lead to higher precision but lower recall, while decreasing the value of the parameter will lead to higher recall but lower precision. Although we were not able to get promising

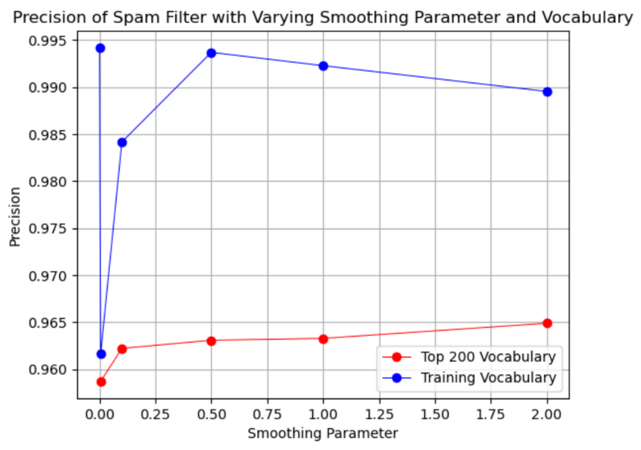

Figure 3: Precision of Spam Filter with Varying Smoothing Parameter and Vocabulary

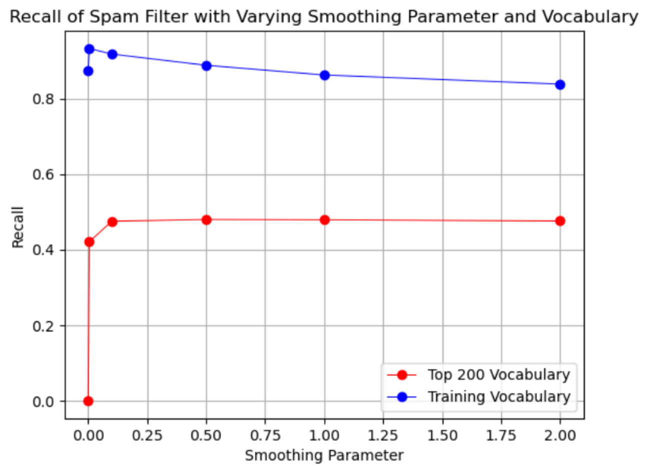

Figure 4: Recall of Spam Filter with Varying Smoothing Parameter and Vocabulary

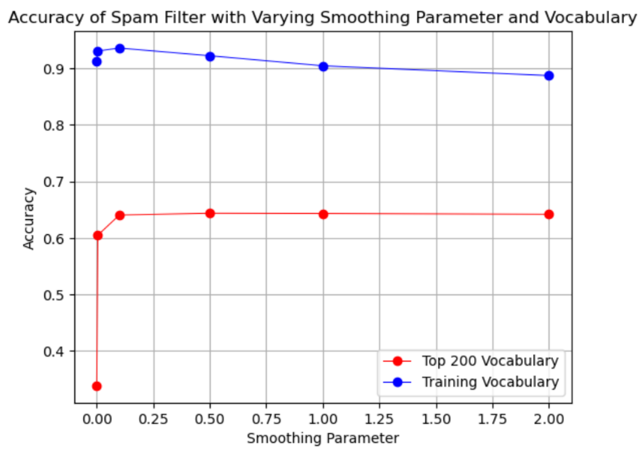

Figure 5: Accuracy of Spam Filter with Varying Smoothing Parameter and Vocabulary

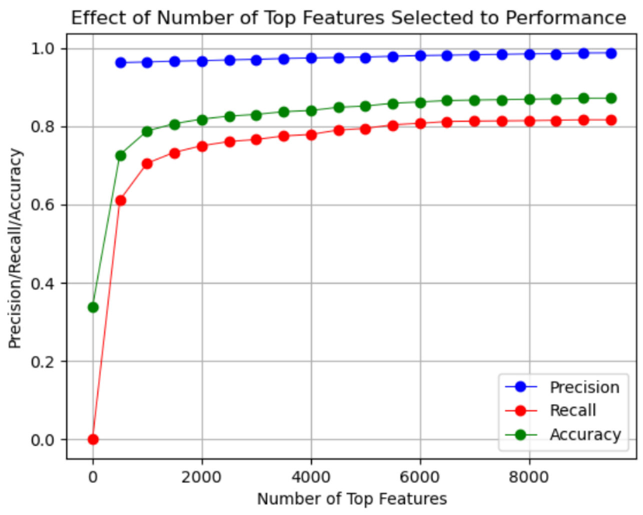

Figure 6: Effect of Number of Top Features Selected to Performance

results with the feature selection for this study when only top 200 features were taken, we had shown that the number of features can improve the performance of the spam filter. The downside is that it would use up more computational cost since the unrelated features it considers increases.

7. REFERENCES

[1] Hoanca, Bogdan. "How good are our weapons in the spam wars?." IEEE Technology and Society Magazine 25.1 (2006): 22-30.

[2] Guzella, Thiago S., and Walmir M. Caminhas. "A review of machine learning approaches to spam filtering." Expert Systems with Applications 36.7 (2009): 10206-10222.

[3] Sheneamer, Abdullah. "Comparison of Deep and Traditional Learning Methods for Email Spam Filtering." International Journal of Advanced Computer Science and Applications 12.1 (2021).

[4] Gangavarapu, Tushaar, C. D. Jaidhar, and Bhabesh Chanduka. "Applicability of machine learning in spam and phishing email filtering: review and approaches." Artificial Intelligence Review 53 (2020): 5019-5081.

[5] Hovold, Johan. "Naive Bayes Spam Filtering Using Word-Position-Based Attributes." CEAS. 2005.

[6] Sahami, Mehran, et al. "A Bayesian approach to filtering junk e-mail." Learning for Text Categorization: Papers from the 1998 workshop. Vol. 62. 1998.