Spam Filter

Brandon Brodzinski

October 2022

Contents

1 Introduction

In this project, I implemented a spam filter based on the Naïve Bayes classifier using a supplied data set of emails. The goal of this assignment is to predict whether each email is spam or ham based on the probability of a given word. I did this by shuffling the data, then splitting it into a training set and a testing set with an 80 / 20 ratio, respectively. Next, I analyzed the data and applied some preprocessing techniques to make the data more efficient and increase accuracy. Lastly, I evaluated the classifier on the test set and reported the results in a confusion matrix.

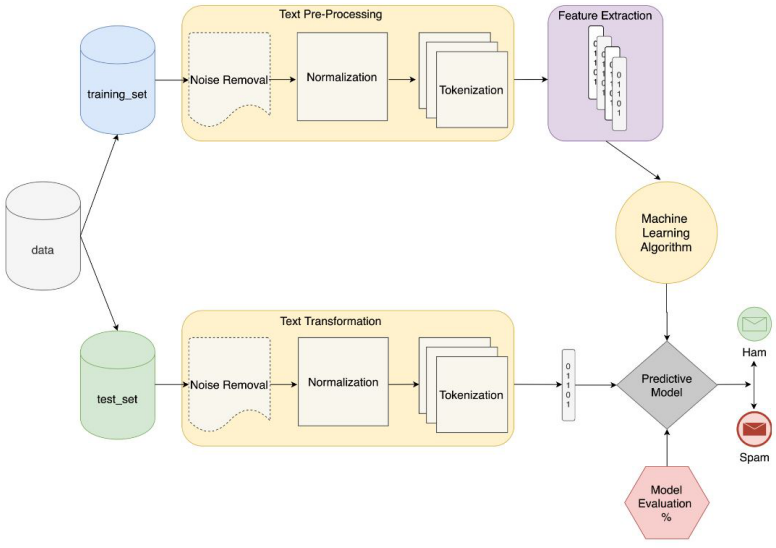

Figure 1: Spam Filter Model Implementation

2 Data Processing

In this section, I will discuss the methodology used to process the data. I did this in two steps, normalization and tokenization. In the normalization step I lower cased all the text, replaced new line characters, and removed digits, quotes, html tags, URLs, etc. After this step was done, I tokenized the string and removed punctuations, and stop words. After I processed the data, I joined all the tokens to create a new string.

2.1 Word Filtering

During the tokenization step, I decided to use a process called; stemming. Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Another way I filtered words was by removing all of the stop words which are words that are commonly used in any language like "the", "how", "to", etc. This helped increase my accuracy as I was filtering out each email's text.

3 Evaluation

In this section, I discuss the results. The highest accuracy I obtained was 96.0% and the lowest was 84.0%. The confusion matrix below is based off the highest accuracy. As you can see, out of twenty percent of the data I obtained 23 true positives, 25 true negatives, 1 false positive, and 1 false negative. Additionally, in Figure 2, I ran three different training and testing splits then graphed them accordingly based on five test runs. Given the results, I noticed as I increased the training percentage the higher accuracy I received and vice versa via the table below.

3.1 Confusion Matrix

	PP	**PN**
P	23	1
N	1	25

96% Accuracy

3.2 Accuracy Table

	Trial Runs					**Average**
70% Training	86.66	84.00	90.66	90.66	88.00	**88.00%**
80% Training	90.90	96.00	88.00	84.00	92.00	**90.18%**
90% Training	94.00	92.00	92.00	88.00	96.00	**92.40%**

Based on 5 Trials

3.3 Accuracy Graph

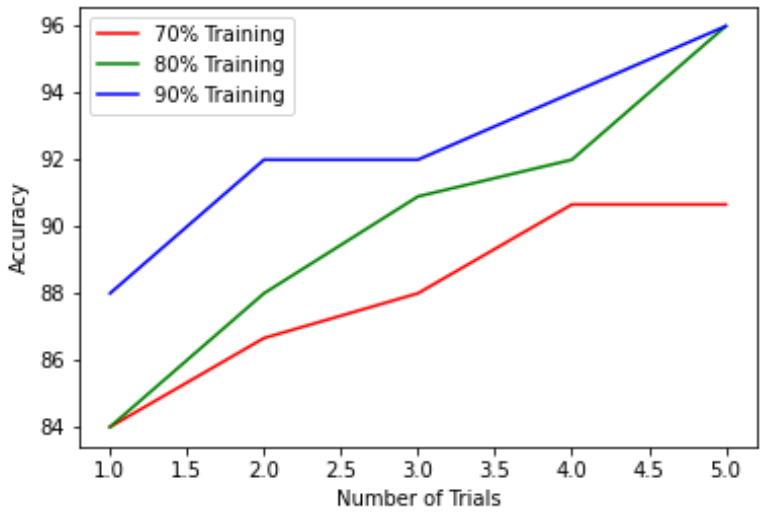

Figure 2: Data Training / Testing Graph

4 What I've learned

In this project, I learned that certain preprocessing techniques affects the accuracy of the algorithm; such as stemming and lemmatization. Another factor I learned was that depending on the percentage split chosen for the training and testing, data also affected the accuracy. Furthermore, I saw how useful a confusion matrix disperses data on the testing set. Finally, I see how beneficial the Naïve Bayes classifier is as it handles both continuous and discrete data while being highly scalable with the number of predictors and data points.