Jayam Sutariya

Dr. Nguyen

CS 482

7th November 2021

Project 2 Report

Introduction

For this first project, we were required to implement one of the Artificial Intelligence concepts we learned in class. The concept is that of the Naïve Bayes Classifier. The algorithm is very useful in classification problems. One such application would be spam message recognition. With our knowledge on the Naïve Bayes Classifier combined with some basic programming principles, the objective is to create a program that trains a naïve bayes classifier based on data and use that model to classify test data as either spam or ham and achieve a high classification accuracy.

Explanation of approach

The program first uses argparse to parse command line arguments to select training or classification mode. For the training mode, a training dataset file is inputted. The program uses the pandas library to categorize the data into DataFrame for ease of use. After that, some data cleaning is done on the data including removing of punctuation and making everything lowercase.

The clean data is traversed and a list of unique words (vocabulary) is created. Word counts for each unique word in each message is calculated. The prior spam and ham probabilities are calculated based on the labels. The total number of spam and ham words are calculated as well. The data is then split into two parts, one is ham and one is spam. Two output files are created that store each word and their word count depending on whether that word was found in a spam or ham message. The header of these files are the total number of spam/ham words and the prior spam/ham probabilities. The whole purpose of these files is so that they can be inputted in the classification mode.

Speaking of the classification mode, here, the test data is imported, and data cleaning is done similar to training data. The spam and ham files that were outputted in the training section are now imported for use.

The main goal of the classification mode is to use the Bayes Probability formula to classify a certain message as spam or ham. To do that, we look at which equation below is greater:

$$P(\text{Spam}|w_1, w_2, ..., w_n) \propto P(\text{Spam}) \cdot \prod_{i=1}^{n} P(w_i|\text{Spam})$$

$$P(\text{Ham}|w_1, w_2, ..., w_n) \propto P(\text{Ham}) \cdot \prod_{i=1}^{n} P(w_i|\text{Ham})$$

We already have P(Spam) and P(Ham) as those values were imported from the files. The main values that we need to calculate are product of all the conditional probabilities. Conditional probability for single certain word is given below:

$$P(w_i|\text{Spam}) = \frac{N_{w_i|\text{Spam}} + \alpha}{N_{\text{Spam}} + \alpha \cdot N_{\text{Vocabulary}}}$$

$$P(w_i|\text{Ham}) = \frac{N_{w_i|\text{Ham}} + \alpha}{N_{\text{Ham}} + \alpha \cdot N_{\text{Vocabulary}}}$$

To calculate the conditional probability, we need the word count of a certain word given its spam/ham. These values exist in the files that were imported. We also need the total number of spam/ham words and again they were imported from the files. We also need the total number of unique words, which is easy to calculate. Furthermore, we will use Laplace smoothing and set alpha to 1.

Now that we have all the information, we will create a dictionary of unique words and their respective conditional probabilities. The conditional probabilities will be calculated using the equation above.

A helper function was for the actual classification part. The helper function takes a SMS message as a parameter. The function then iterates through all the words in the message and uses the first equation to calculate $P(\text{Spam}|w_i)$ and $P(\text{Ham}|w_i)$ based on P(Spam), P(Ham) and all the condition probabilities for the words. This is done for all the test dataset messages and prediction is made depending on which value ($P(\text{Spam}|w_i)$ or $P(\text{Ham}|w_i)$) is higher. All the predictions are written to file and the accuracy is calculated by comparing the test dataset labels and the predictions.

Results and Discussion

The original spam dataset file was split into training and testing dataset randomly. Training was conducted on the training dataset and testing was conducted on the testing dataset. After multiple runs, the classification accuracy on the testing dataset came out to be **87%** on average. This classification accuracy is much higher than the 70% required.