Report for mandatory Problem - NLP

Matthias Thurnbauer
Deggendorf Institute of Technology
Deggendorf, Germany
matthias.thurnbauer@stud.th-deg.de

Ihab Ahmed
Deggendorf Institute of Technology
Deggendorf, Germany
ihab.ahmed@stud.th-deg.de

Karim Sayed
Deggendorf Institute of Technology
Deggendorf, Germany
karim.sayed@stud.th-deg.de

July 3, 2022

1 Introduction

1.1 Goals of our project

This project was our attempt at creating a version of a Naive-Bayes based spam filter for emails that would give accurate classifications of new emails. We employ Laplacian smoothing to improve the model.

1.2 Related Work

2 Background

In this section the methods and mathimatics that are used in our implementation are discussed.

2.1 Naive Bayes Classifiers

Naive Bayes classifiers are a category of classifiers that use probaility to predict classifications of data. Typically Naive Bayes classifiers are based on Bayes' theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using this formula one would be able to calculate a probability of an event A occuring givent the fact that event B has happened using conditional probability. The same concept can then be used to find the probability of data item B being a part of class A by assuming that it indeed is a part of that class this process is done for all classes and the class that produces the highest probability is considered to be the most likely that B belongs to.

2.2 Laplacian smoothing and Naive Bayes

One of the problems that can occur when using Naive Bayes happens becasue we must have a likelihood calculated for each word when training the data on the training set, it can happen that the classifier has never encountered that particular word in the training set and thus cannot accurately calculate the likelihood, for that reason laplacian smoothing is introduced. This changes the formula for the liklihood to:

$$P(new\_word|A) = \frac{n\_word\_A + alpha}{total\_n\_words\_A + alpha * total\_n\_unique\_words}$$

Now, even if the word was never encountered before we will still be able to give it a probability and having it be set to zero will not disrupt the Naive Bayes classifier.

3 Methodology

This section goes over how the data was preprocessed and how we implemented the idea of a naive bayes classifier that uses laplacian smoothing

3.1 Preprocessing

To begin, we import the dataset and start with a data exploration and cleaning phase were we start off by randomizing the data set to avoid any biasies. Next, we can begin splitting our randomized data into test and training at an 80% training to 20% test. The data we use have some extra charects that would cause problems for the program and are thus removed. All text is then set to lowercase to make the process of counting and comparision of the string easier later on. We then extract all the unique words that exist inside the training set. We then calculate the count of each unique word we have previously detected and add that data to our training data. We now have all our training email with a sort of count encoding for each of the words in the dictionary as diffferent categories.

3.2 Implementation

Once our data is clean we finally get to applying naive bayes to produce a probability that can help us decide wether or not a new email is spam. We begin by calculating the probabilities for an email being spam or not, called ham. where $P(Spam) = \frac{totalspamemails}{totalemails}$ and $P(Ham) = \frac{totalhamemails}{totalemails}$. Next we calculate the probability of each world, in the previously defined list of unique words, given that its ham and then given that its spam using Naive Bayes: $P(word|Spam) = \frac{n\_word\_spam+alpha}{total\_n\_word\_spam+alpha*total\_unique\_words}$ and $P(word|Ham) = \frac{n\_word\_ham+alpha}{total\_n\_word\_ham+alpha*total\_unique\_words}$. Now that we have calculated the Naive Bayes probabilities of each of our classes, we can implement them into a a function that can categorize a new email into spam or ham based on the comparison of the probability of being ham and spam given the input email.

4 Results

This section covers the results our model achieved and a description of our used dataset for training and testing the model.

4.1 Dataset description

We use a dataset consisting of 5572 ham and spam SMS messages which are not chronologically sorted [**dataset**]. We first randomize the whole dataset and then split it to 80% train and 20% test data, respectively, which results in 4458 entries for training data and 1114 for test data. After that we validate the splitting process by taking a look at the label distributions. We've found that we have around 86% ham and 14% spam messages in the training data whereas the test data has around 87% ham and 23% spam messages. We can conclude that this is a good data distribution for both datasets as there are a lot more ham than spam messages in the real world.

4.2 Metrics

See following figure for the result of the classification on the test data set 1. We can see that 1107 messages have been correctly classified whereas the remaining 7 data points were classified incorrectly. We use the metrics *accuracy*, *precision*, *recall* and *f1-score* for measuring the performance of our model. To calculate these metrics we have to compute the true positive, true negative, false positive and false positive values for the predictions. For our calculations the true spam prediction is denoted as true positive which makes the other notations clear.

Accuracy Based on the number of correct predictions we get an overall accuracy of 0.9937. We calculate the accuracy like this:

$$accuracy = \frac{true\_positive + true\_negative}{false\_positive + false\_negative + true\_positive + true\_negative}$$

Precision The precision describes the ability of predicting a message as spam correctly and we calculate it like this:

$$precision = \frac{true\_positive}{true\_positive + false\_positive}$$

The calculated precision is 0.9855 so 98% of all predicted values are classified as spam.

Recall Recall is the metric which takes the true positives and divides is by the total number of spam classes. It describes how many spams were correctly classified. We calculate it like following:

$$recall = \frac{true\_positive}{true\_positive + false\_negative}$$

Our recall score is 0.9645, which means 96% of spam messages could be classified as spam.

F1 Score This is the harmonic mean between the precision and recall, which we calculate like this:

$$f1\_score = 2 * \frac{precision * recall}{precision + recall}$$

We get an *f1_score* of 0.9749 which is interpreted as a good value because the higher (maximum value of 1) the better.

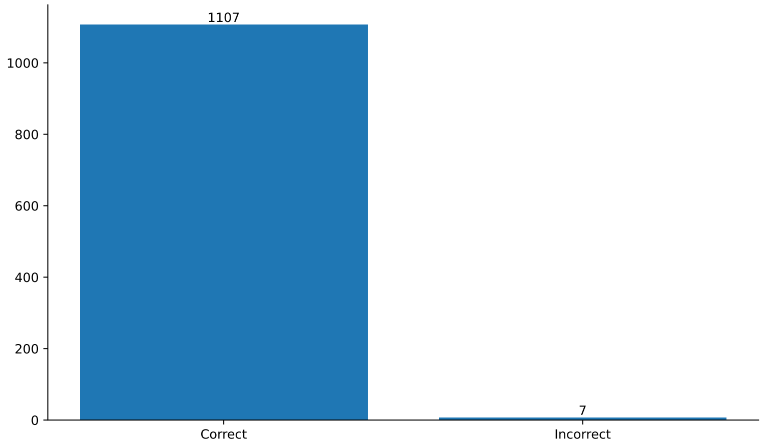

Figure 1: Results for classifying the test data

4.3 Strength and limitations

Hm let's see if we keep this section

4.4 Other interesting findings

I will try to analyze which messages are incorrectly classified.

5 Conclusion

This section concludes this report on our implementation of a Naive Bayes spam filter for our course *Natural Language Processing*.

5.1 Further improvements

5.2 Summary

We want to finish this report with a short summary of our work:

- We want to implement a spam filter based on the Naive Bayes algorithm to detect whether a message is spam or ham
- Our algorithm uses multinomial Naive Bayes
- Dataset consists of spam and ham SMS messages
-