Deggendorf Institute of Technology

Faculty of Computer Science

Bachelor Artificial Intelligence

Natural Language Processing

Mandatory Problem

Coursework Problem 7.3

Name: Zi Xun Tan

Student ID: 00819086

Examiner: Prof. Dr. Patrick Glauner

Deggendorf, 01. July 2022.

Introduction

Gone were the days when spam was merely a nuisance. Today, spam has become an aggressive threat, where attackers constantly try to trick users, manipulating users to click on things that they shouldn't using various methods. If users happen to click the wrong thing in the spam messages, personal data will be exposed.

Therefore, spam filtering has grown in importance and relevance. Organizations need to utilize a spam filter to reduce the risk of users clicking on something they shouldn't and to protect their personal data from a cyberattack.

In this report, I outline a basic implementation of a spam filter in Python using the Naive Bayes classifier. My goal is to create a spam filter that achieves over 80% accuracy in classifying messages.

Methodology

In this section, I explain the mathematical and conceptual foundations of the spam filter.

Naive Bayes classification is a simple probability algorithm based on the fact that all features of the model are independent. In the context of spam filtering, we assume that each word in the message is independent of all other words, and we count them in ignorance of the context.

Bayes' Rule

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

With Bayes' Rule, we want to determine the probability a message being spam, given it contains certain words. To do this, we determine the probability that each word to be found in spam, and then multiply these probabilities together to get the overall metric for spam used in the classification. In our context:

$P(SPAM) =$ the probability of a message being spam

$P(WORD|SPAM) =$ the probability of a word to be found in the spam messages

$P(HAM) =$ the probability of a message being non-spam

$P(WORD|HAM) =$ the probability of a word to be found in the non-spam messages

The probability of each word to be found in spam or non-spam messages is calculated as follows:

$$P(WORD|C) = \frac{count(WORD\ in\ C)}{\boldsymbol{N}c}$$

To avoid overfitting, we use Laplacian smoothing:

$$P(WORD|C) = \frac{count(WORD\ in\ C) + 1}{\boldsymbol{N}c + 1 * \boldsymbol{N}unique}$$

- where C is $SPAM\ /\ HAM$
- $\boldsymbol{N}c$ is the total number of words in C
- $\boldsymbol{N}unique$ is the total number of unique words in the whole dataset

$$P(C) = \frac{count(C) + 1}{total\ number\ of\ messages + 2}$$

We add 2 here to account for the number of classes we have – a spam class and a ham class.

In the end, the probability of a message being spam calculated as follows:

$$P(SPAM|WORDS) = \frac{P(WORDS|SPAM)\ P(SPAM)}{P(WORDS|SPAM)P(SPAM) + P(WORDS|HAM)P(HAM)}$$

If $P(SPAM|WORDS)$ is greater than the "spam threshold" of 0.5, the message is classified as a spam.

Experimental Results and Discussion

In this section, I discuss the details of the dataset used, the results and metrics and the strengths and limitations of my model.

Dataset

To implement the spam filter, I used the SMS Spam Collection Data Set from the UCI Machine Learning Repository. [1] The structure of this dataset is simple. It contains two

columns, one for the label “ham/spam” and the other with the text of the message. It contains 747 spam messages and 4825 non-spam messages. The dataset was cleaned, and the label “ham/spam” was changed to “0/1”. Then, the dataset was split into randomized train/test sets with a 70:30 ratio.

Results and Metrics

The classification results for the test set are as follows:

	Actual Ham (0)	Actual Spam (1)	Sum
Predicted Ham (0)	1390	35	1425
Predicted Spam (1)	39	208	247
Sum	1429	243	1672

$$Number\ of\ test\ data: 1672$$

$$True\ Positive: 208$$

$$True\ Negative: 1390$$

$$False\ Positive: 39$$

$$False\ Negative: 35$$

The metrics were as follows:

$$Accuracy = 0.9557$$

$$Precision = \ 0.8421$$

$$Recall = 0.8560$$

$$F1 = 0.8490$$

Strengths and Limitations

The spam filter had an accuracy of 95.57% on the test set I used, which is a promising result. My original goal was an accuracy of over 80%, and I achieved that. Our model also had high precision and high recall, which shows that the spam filter returns accurate results (high precision), as well as returns a majority of all positive results (high recall).

However, the dataset I used for training the model contains only SMS messages. For other types of spam, such as email spam, the performance may not be as good.

In addition, the model is only able to predict spam messages in English. To filter spam messages in other languages, a different dataset would be required to train the model.

Conclusion

In this report, I presented the motivation of this project and the mathematical and conceptual foundations of the spam filter. I also discussed the strengths and limitations of the spam filter.

To further improve the model, a larger dataset or a dataset in different languages is needed.

The assumption of naive independence of words is a drawback when compared to more advanced forms of NLP. Other approaches, such as classifiers that use word embeddings to encode meaning, such as context from sentences, could be used to obtain more accurate predictions.

References

[1] “UCI Machine Learning Repository: SMS Spam Collection Data Set.” https://archive.ics.uci.edu/ml/datasets/sms+spam+collection (accessed Jul. 01, 2022)