Brian Kilburn

CSC 546 Artificial Intelligence

Dr. Wookie

Assignment 1

Report

After much deliberation working through this assignment, the features that were introduced into this SMS Spam Filtering system was completed with great scores. If overfitting is not a factor in this model, then the accuracy of this system produces a 92% detection rate against spam messages in an email. In this system, four features were introduced: Website/URL detection, Word frequency, Sentence length, and Spam word dictionary. The justification for implementing these features can be determined in the resulting data that was collected from the system. The system incorporates different programming languages to export, test, and validate the collected information through several statistical functions. The python program implements the detection features that will be used to calculate spam or ham messages. Upon the completion of feature implementation, the python system exports two files that will be used to measure the success of the featured system. The first file to be exported will be a feature set .csv file, this file will be used in the R system. This system will utilize logistical regression to train the model for spam or ham detection. Then it will perform some calculations to determine the feature set accuracy by measuring the ROC curve, AUC, and displaying the information in a confusion matrix. The confusion matrix will be used to calculate the True Positive and False Positive rating for the model. After completion of the logistic regression calculation, then a K-Fold Cross Validation will be implemented to ensure the model does not introduce any overfitting and therefore prove the stability of the system. In this experiment, five- and ten-fold cross validation will be implemented. Once the R system has completed, then a .arff file will be available, this file will be produced by the python system simultaneously with the .csv file. The .arff file can be opened in a none computer scientist program known as Weka. This system will utilize the same functionality as the R system and test the validation of the implemented Spam detection feature set from the python program.

In this experiment the raw dataset contained 5574 SMS messages, which is composed of 747 spam and 4827 legitimate messages. This dataset contained two columns: the first column labeled the message if it was ham or spam, and the second column is the actual message data. Before implementing the features in the python program, the message was first processed using various tested methods that improves the accuracy of a message. The preprocessing steps included: converting the message to lowercase letters, removing any punctuation, removing stop words, Snowball Stemmer implementation, and string tokenization. After processing SMS messages, the feature .csv and .arff file were exported. The .csv file was picked up and interpreted in the R system program, first splitting the data into a testing set and a training set. The training set was first used in the glm() function to train the model using the label (ham or spam) as the dependent variable and ~. Which means all other columns of data as the explanatory or response variables. Then that model was stored and used to create a prediction against the testing set and stored into a fitted results variable. The predicted variables were summarized and measured for performance. First the ROC Graph which displays the results of TPR and FPR, it measures the usefulness of the featured system. The R system produced this ROC curve for the actual testing model, this is where it counts. Sometimes systems can perform very good with training data but do very poorly during testing or under actual performance. According to this graph we are looking at a true positive rate of around 90% - 93%.

A screenshot of a cell phone

Description generated with very high confidence

The true positive rate is the measure of correctly classified or identified data. The other measurement is the false positive rate which is a measurement of the incorrectly classified or identified data. A confusion matrix is a tool that can be used to calculate tpr and fpr, it is used to visualize the performance of an algorithm, in this case the performance of the AI model.

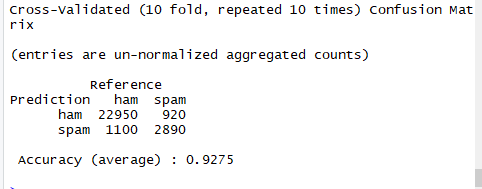
A screenshot of a cell phone

Description generated with very high confidence

As you can see in the image the performance of the model was successful, the tpr calculated to .955 and the fpr calculated to .297, next the R system will begin K-fold cross validation (5 fold and 10 fold).

A screenshot of a cell phone

Description generated with very high confidence



Both models display a 92% accuracy, which has some credibility that overfitting is not the situation that we are looking at in this model. As you can see the images share the repeated value for the cross-validation execution, for instance the 10-fold cross validation was executed 10 times which produced the correct classification of ham messages 22950, and spam 2890 correctly classified which still averages out to total accuracy of 92% - 93%.

The ROC for K-fold Cross validation demonstrates the same statistical information as the set testing curve: the 5-fold here displays the resulting matrix of tpr and fpr measurements.

A screenshot of a social media post

Description generated with very high confidence

10-fold cross validation produces the same ROC curve measurement as well: The measurement breaks just below the .8 and steadily increases falling short of 1.0 sensitivity.

A screenshot of a social media post

Description generated with very high confidence

Now for feature justification, using the weka program it is easy to determine which feature proved to be more effective when detecting spam in an SMS message. This first image is the original classification for ham(blue) and spam(red). The image on the right is the strongest feature that made it possible for the model to predict more accurately. The blue on the left represents the correct classification of ham and the little bit of red shows the misclassification of spam. The bar on the right show misclassification of ham, and the correct classification of spam. Overall it looks like it did misclassify the ham more than what should be necessary.

A screenshot of a cell phone

Description generated with high confidenceA screenshot of a cell phone

Description generated with very high confidence

Next I want to demonstrate the website/URL detection feature and show the results: It appears least effective in the detection, but what is interesting is that a URL in my opinion would seem to determine more precisely whether or not a message contained spam, but in this case that is not the situation. However, it did contribute to the correct classification of ham. The Frequency word count also was a high contributor to the correct classification of ham, more so then spam just like the website/URL detection. But, the frequency word count classifies some spam messages.

A screenshot of a social media post

Description generated with very high confidenceA screenshot of a social media post

Description generated with very high confidence

The next strongest contributor and final feature was the spam word dictionary detection feature, I also expected it to perform better but it contributed to more ham detection which could be a good sign in this situation because of the ratio of ham as opposed to spam in the original dataset. It did correctly classify more spam cases looking at the variability of the bar on the right.

A screenshot of a social media post

Description generated with very high confidence

In conclusion the Word count and Spam word dictionary features where the most important tools that were used in detecting spam messages in the dataset. Hence, those features contributed more to the accuracy of the program than any of the others. I think that this information could be used to help improve the spam filter by either taking some of the weaker features and improving them or introduce new features that can be more effective such as proper n-grams or utilize tfidVectorization. Other classifiers could be implemented to help separate the data also. The Bayesian filter was one of interest that I wanted to implement into the program.

Works Cited:

<https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>

<https://towardsdatascience.com/spam-classifier-in-python-from-scratch-27a98ddd8e73>