Prolog Bayesian Classifier to Detect Spam E-mails

Christopher Seniow

Abstract - With the number of e-mails being sent and received constantly increasing, there is an ever-increasing amount of spam e-mails being received. Correctly detecting and classifying these spam emails is important to ensure that users are not overrun with useless spam emails and do not potentially miss important emails because of it. One method of detecting these spam emails is though a Naïve Bayes classifier that uses the probability of certain words appearing in the email to predict whether the email is spam or not. To implement this, a Bayesian classifier was made in Prolog and Python along with the library Pyswip, was also used to format the training data and feed it into the Prolog classifier.

I. METHOD

The goal is to calculate the probability that a given email is spam (P(spam | some email)) and calculate the probability that that email is not-spam (P(not-spam | some email)), and then compare the results of the 2 calculations. If the probability that the email is spam is higher than the probability that it is not-spam, we can predict that the email is a spam email, and vice versa.

To do this, first a frequency histogram of all words that appear in emails that are spam and non-spam is created. This is calculated using the numOf(Word, Len, Spm) method in prolog where 'Len' is returned as the number of occurrences of a given variable 'Word' given that it is either spam or not spam. The total number of words that occur in spam emails and the total number of words that appear in non-spam emails are calculated as well. This is calculated using the totalLen(Spm, L) method in prolog where 'L' is returned as the total number of words that occur in either spam or non-spam emails.

Next, we are able to calculate the probability that a given word occurs in either a spam email or a non-spam email (P(Word | spam) / P(Word | not-Spam)). To do this we take the total number of times that the given word appears in spam (or non-spam) emails and divide it by the total number of words that appear in all spam (or non-spam) emails. This is calculated by the prolog function freq(Word, Spm, Freq) where 'Freq' is returned as the probability that the given word occurs in either a spam email or a non-spam email. In this method, when calculating this probability, 1 is added to the total number of times that the given word appears in spam (or non-spam) emails and the number of distinct words that appears in all emails is added to the total number of times that the given word appears in spam (or non-spam) emails. This is because a pseudocount of $\alpha = 1$ is added to the number of times each word occurs to ensure that if a word in the email that is being checked never appears in the dataset, the probability is not calculated to be 0 thus canceling out the probability obtained by the other words in the email.

Finally in order to calculate the probability that a given email is spam (P(spam | some email)), first the prior probability that a method is spam is calculated (P(spam)). This is done by dividing the total number of words that appear in spam emails by the total number of words in all emails. This is then multiplied by the probability that each word in the email appears in a spam email (P(word | spam)). The same is done to calculate the probability that the email is not spam. This can be represented by the following formulas:

$$P(\text{spam}|\text{some email}) = P(\text{spam}) * P(\text{word1} \mid \text{spam}) * P(\text{word2} \mid \text{spam}) \ldots$$

$$P(\text{Not spam}|\text{some email}) = P(\text{spam}) * P(\text{word1} \mid \text{spam}) * P(\text{word2} \mid \text{spam}) \ldots$$

Figure 1: Probability equations

This is calculated using the probabilitySpam([H|T], Prob) and the probabilityNotSpam([H|T], Prob) functions in prolog where 'Prob' is returned at the probability. Once that is calculated the results of these two calculations can be compared and the one with the higher probability is likely to tell us if the email is spam or not.

II. IMPLEMENTATION

In order to implement this method, spamClassifier.py was created. Prolog is accessed through the python program using the library "pyswip

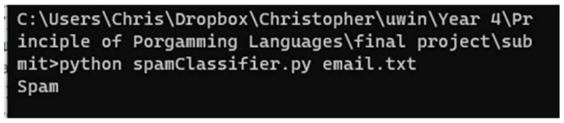
0.2.10" [1]. Using pyswip, the prolog file finalProject.pl is then consulted, which contains the implementations of the prolog methods described above. The training data is then cleaned and formatted (see Dataset section for more information) and asserted into the prolog database. The given email is then formatted correctly and the prolog method isSpam() is queried. This returns the probability that the given email is spam and the probability that it is not spam. Based on which probability is larger, a prediction if the email is spam or not is made. To use this program, simply run spamClassifier.py and input your given email in text file and the program will return whether it believes the given email is spam or not.

III. DATASET

The training data set used was found from the "Spam-or-Ham-Email-Classification" [2] project by user "balakishan77" on kaggle.com. This data set consisted of a csv file emails.csv which contained 2 columns, the first containing the emails, and the second containing a 1 or a 0 indicating if the email is spam or not respectively. This dataset was then slightly reformatted and reduced to lower run times, to the file data.csv which was used in this implementation. The mentioned python program spamClassifier.py was used to read this csv file and first remove any non character or number symbols, and then split up each individual word in the given email and separated by commas. This string was then formatted into a string of the form email([word1,word2,…],spam) or email([word1,word2,…],not) based on whether or not the second column of the file was a 1 or a 0. This string was then asserted into the prolog database to be used in calculating the probabilities.

IV. RESULTS

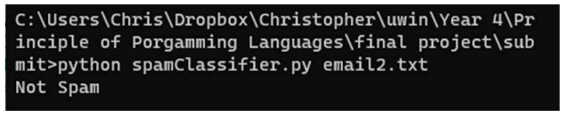
This implementation was successful in predicting the whether the given emails were spam or not. To test this some of the emails that were cut from the original emails.csv data set were used and were correctly predicted. See below figures for some demonstration of the results. Figure 2 is the result of a spam email "Subject: your trusted source for prescription medication . best prescription generic meds 4 less . anger is one of the sinners of the soul . write what you like ; there is no other rule . life ' s most urgent question is : what are you doing for others ? gold for friends , lead for foes ." which is in the text document 'email.txt'.



```
C:\Users\Chris\Dropbox\Christopher\uwin\Year 4\Pr
inciple of Porgamming Languages\final project\sub
mit>python spamClassifier.py email.txt
Spam
```

Figure 2: Spam email test

Figure 3 is the result of an email that is not spam "If against undeserved claims in the event of exigencies that force it to take an extreme step . the union power minister" which is in the text document 'email2.txt'



```
C:\Users\Chris\Dropbox\Christopher\uwin\Year 4\Pr
inciple of Porgamming Languages\final project\sub
mit>python spamClassifier.py email2.txt
Not Spam
```

Figure 3: Not Spam test email

REFERENCES

[1] "pyswip," *PyPI*. [Online]. Available: https://pypi.org/project/pyswip/.

[2] balakishan77, "Spam-or-Ham-Email-Classification," *Kaggle*, 12-Aug-2018. [Online]. Available: https://www.kaggle.com/balakishan77/spam-or-ham-email-classification/data?select=emails.csv