**Email Ham or Spam Classifier using spam pattern regular expressions and finite autmaton recognizer**

Bautista, Cempron, Eugenio

**Abstract**

String pattern recognition can be used to classify an email as spam or ham. Spam email are considered as noise. When detected spam mails are excluded from the user’s email inbox then transferred to the spam folder. While ham emails are the actual emails that the user might be interested in. To classify a spam using pattern recognition, regular expressions are used to model the pattern while finite automatons are used as acceptors that recognizes the patterns. Detmination of patterns which are useful for detection and the accuracy of combined pattern recognitions are presented in this paper

**Introduction**

String pattern recognition can be used to classify an email as spam or ham. Spam email are considered as noise. When detected spam mails are excluded from the user’s email inbox then transferred to the spam folder. While ham emails are the actual emails that the user might be interested in. To classify a spam using pattern recognition, regular expressions are used to model the pattern while finite automatons are used as acceptors that recognizes the patterns.

Cetrain string patterns that can be used to classify spam or ham emails are presented by the kernlab package by the UCA spam data [1]. These includes frequent use of: the word “you”, uppercased words, and dollar signs. This data set contains pre counted pattern recognized and each email is pre labelled as ham or spam. This data set dates back to 1999 and is currently only being used for demonstration purposes of teaching machine learning

Another more recent dataset containg ham and spam data are provided by kaggle and are used for competition. With the winner having the most accurate predictor of ham or spam [2]. This dataset contains 5572 emails. Arranged in tabular form each row represents an entry and with two columns: one for classification of the email as ham or spam, and the other column contains the email message.

Our goal is to present a method of predicting the classification of an email as spam or ham using regular expressions to model the patterns and finite automatons as recognizers. Programming laguages used are: Python for developing the spam detector software, and R for exploratory analysis on determining the best predictors to use.

**Methodology**

|  |
| --- |
|  |
| Figure 1: System flow |

The flow of the system are as follows. First is to determine the search tags which will be used for the ham spam repdictor. Second is to generate the regular expressions of the search tags. Third and Fourth is to generate an non deterministic then deterministic finite automaton that will be used to recognize the pattern. Fifth an pattern recognizer that takes the email input and check againts the DFAs. Then if any pattern was recognize classify the email as spam or otherwise ham. This is illustrated in Figure1.

Determine Search Tags

|  |
| --- |
|  |
| Table 1: Search tags and regular expression |

Our intuition while viewing the hamspam kaggle data suggest we include the following search tags displayed in Table1. The corresponding regular expression are also shown. However each regular expression are tested as a single predictor using generalized linear model [3], and not all are necessary for prediction. Search tags that qualified as predictors are shown in bold and italicized.

Regular Expression to Finite Automaton

|  |
| --- |
|  |
| Figure 2: NFA transition table for search tag **free** |

Selected Regular expressions are first transformed into Non Deterministic Finite Automaton using the Thompson algorithm[4]. Showin in Figure 2. Then the Non Determinism is transformed into Deterministic Finite Automaton Shown in Figure 3. The generated DFA are used as the recognizer for the spam detecting pattern

|  |  |
| --- | --- |
|  |  |
| Figure 3: DFA for spam pattern recognizer of search tag **free** | |

Email Input Parsing

|  |
| --- |
|  |
| Figure 4: pattern matching abc in substrings of input |

On finding accept states on the input, the input is parsed as follows. Each substring of the input is checked. First substring is the whole input string. The next substring excludes the last character. And so

on. Afterwards, the first character is excluded from the input string and then the next. And so on. For each exclusion the previous parse are repeated. When an accepted substring is found. The accepted substring will be excluded from the parse already. Shown in Figure 4 for a finite automation that only accepts abc.

**Performance Evaluation**

|  |
| --- |
|  |
| Figure 5: Performance analysis of the spam detector software |

Two datasets was used to determine the performance of our developed spam detector software. The first data set was used in linux servers as the databse of common spam emails from 2002. Testing the effectiveness of our software in this dataset yields 97% accuracy.

The other dataset was the kaggle hamspam data where our software yields 97% accuracy on detecting spam emails but onlu 80% accuracy on detecting ham emails. The total combined accuracy of our spam detector is 83%.

**Summary and Conclusion**

As can be seen from the performance analysis of our detector that it is prone to false positives. As is common to our personal experience with using email where several mails are classified as spam when is not.

Presented is a method for detecting emails as spam or ham using regular expression and finite automatons to recognize a spam and a ham email. The accuracy is relatively low with only 83% combined. And the low accuracy is the effecto of false positives favoring emails to be classified as spam.

Accuracy can be achieved by using more complex approach as, say, using Natural Language Processing. However our simplistic approach, despite its inaccuracy can stil be used in a practical sense because of the scalability inherent to its simplicity. Because scalability is important to such applications as detecting spam and ham on email servers which usually processes several emails at a time. Thus simpler faster methods are favored over accurate but slower ones.

**Resources**

Executables, scripts, source codes, and exploratory analyses used in the development of our spam detector software is available for download in <https://github.com/jonathan-cempron-1/hamspam?fbclid=IwAR1rBGo5yFRCIy3I3dnLDiJugsCa-PXVRgBvzBgi5AXBjTLTNVI69LCy6O0>

**References**

[1] UCA spam <https://archive.ics.uci.edu/ml/datasets/Spambase>

[2] kaggle hamspam <https://www.kaggle.com/uciml/sms-spam-collection-dataset>

[3] generalized linear model <https://onlinecourses.science.psu.edu/stat504/node/216/>

[4] thompson algorithm <http://www.cs.may.ie/staff/jpower/Courses/Previous/parsing/node5.html>

[5] linux focus spam emails 2002 <http://www.linuxfocus.org/common/src/article279/spam_samples.html>