Email Ham or Spam Classifier Using Spam Patter Regular Expressions and Finite Automaton Recognizer

CSC615M Milestone 2

Carlo Migel Bautista
De La Salle University
Manila, Philippines
carlo_migel_bautista@dlsu.edu.ph

Jonathan Paul Cempron
De La Salle University
Manila, Philippines
jonathan_paul_cempron@dlsu.edu.ph

Patricia Anne Eugenio
De La Salle University
Manila, Philippines
patricia_anne_eugenio@dlsu.edu.ph

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. Determination of patterns which are useful for detection and the accuracy of combined pattern recognitions are presented in this paper.

KEYWORDS

Finite Automata, Regular Expressions

ACM Reference Format:
Carlo Migel Bautista, Jonathan Paul Cempron, and Patricia Anne Eugenio. . Email Ham or Spam Classifier Using Spam Patter Regular Expressions and Finite Automaton Recognizer: CSC615M Milestone 2. In *Proceedings of CSC615M*. ACM, New York, NY, USA, 4 pages.

1 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 userâĂŹs 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 UCI spam data [3]. These includes frequent use of: the word âĂIJyouâĂİ, 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.

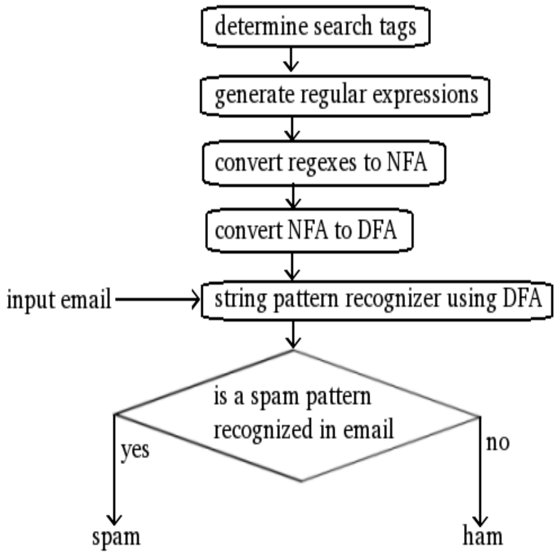

determine search tags
generate regular expressions
convert regexes to NFA
convert NFA to DFA
input email
string pattern recognizer using DFA
is a spam pattern recognized in email
yes
no
spam
ham

Figure 1: System Flow

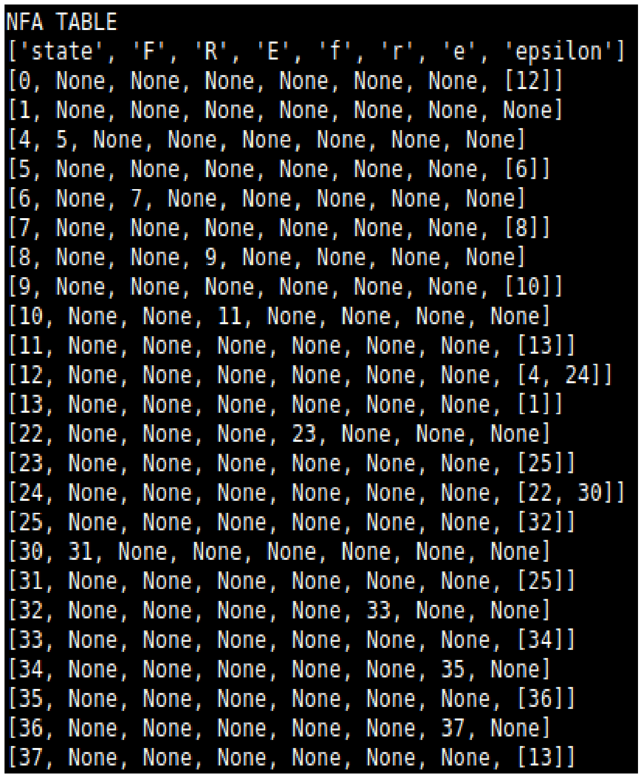
Our goal is to present a method of predicting the classification of an email as spam or ham using regular expressions to model the patterns and finite automatons as recognizers. Programming laguages used are: Python for developing the spam detector software, and R for exploratory analysis on determining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

2 METHODOLOGY

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 Figure 1.

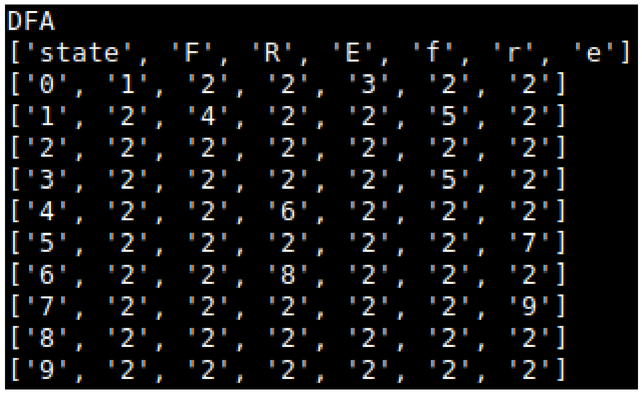
2.1 Determine Search Tags

Our intuition while viewing the hamspam kaggle data suggest we include the following search tags displayed in Table 1. The corresponding regular expression are also shown. However each regular expression are tested as a single predictor using generalized



```
NFA TABLE
['state', 'F', 'R', 'E', 'f', 'r', 'e', 'epsilon']
[0, None, None, None, None, None, None, [12]]
[1, None, None, None, None, None, None, None]
[4, 5, None, None, None, None, None, None]
[5, None, None, None, None, None, None, [6]]
[6, None, 7, None, None, None, None, None]
[7, None, None, None, None, None, None, [8]]
[8, None, None, 9, None, None, None, None]
[9, None, None, None, None, None, None, [10]]
[10, None, None, 11, None, None, None, None]
[11, None, None, None, None, None, None, [13]]
[12, None, None, None, None, None, None, [4, 24]]
[13, None, None, None, None, None, None, [1]]
[22, None, None, None, 23, None, None, None]
[23, None, None, None, None, None, None, [25]]
[24, None, None, None, None, None, None, [22, 30]]
[25, None, None, None, None, None, None, [32]]
[30, 31, None, None, None, None, None, None]
[31, None, None, None, None, None, None, [25]]
[32, None, None, None, None, 33, None, None]
[33, None, None, None, None, None, None, [34]]
[34, None, None, None, None, None, 35, None]
[35, None, None, None, None, None, None, [36]]
[36, None, None, None, None, None, 37, None]
[37, None, None, None, None, None, None, [13]]
```

Figure 2: NFA Transition Table for Search Tag free



```
DFA
['state', 'F', 'R', 'E', 'f', 'r', 'e']
['0', '1', '2', '2', '3', '2', '2']
['1', '2', '4', '2', '2', '5', '2']
['2', '2', '2', '2', '2', '2', '2']
['3', '2', '2', '2', '2', '5', '2']
['4', '2', '2', '6', '2', '2', '2']
['5', '2', '2', '2', '2', '2', '7']
['6', '2', '2', '8', '2', '2', '2']
['7', '2', '2', '2', '2', '2', '9']
['8', '2', '2', '2', '2', '2', '2']
['9', '2', '2', '2', '2', '2', '2']
```

Figure 3: DFA Transition Table for Search Tag free

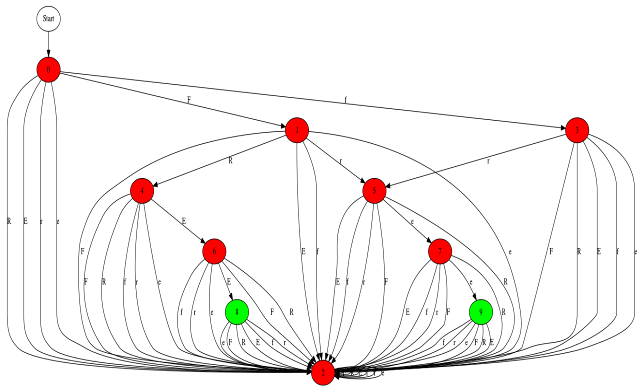
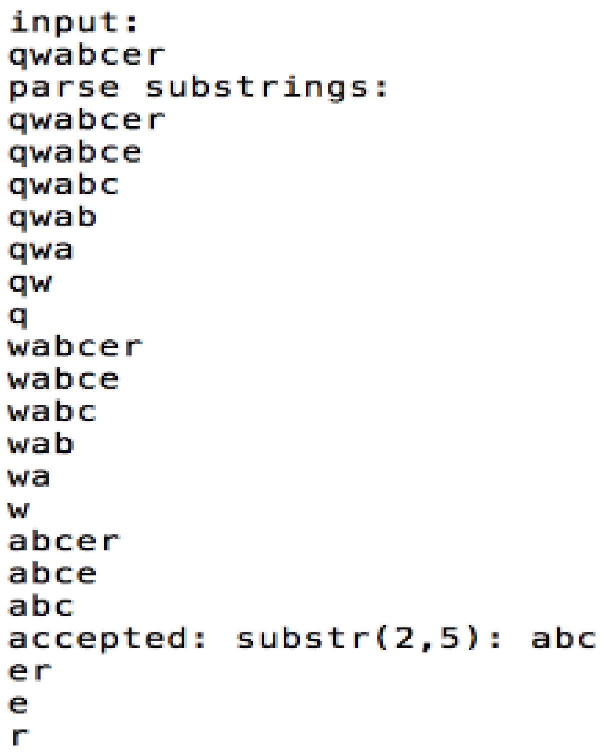


Figure 4: DFA Transition Table for Search Tag free



```
input:
qwabcer
parse substrings:
qwabcer
qwabce
qwabc
qwab
qwa
qw
q
wabcer
wabce
wabc
wab
wa
w
abcer
abce
abc
accepted: substr(2,5): abc
er
e
r
```

Figure 5: Pattern Matching abc in substrings of input

linear model [1], and not all are necessary for prediction. Search tags that qualified as predictors are shown in bold and italicized.

2.2 Regular Expression to Finite Automaton

Selected Regular expressions are first transformed into Non Deterministic Finite Automaton using the Thompson algorithm [4]. Shown in Figure 2.

Then the Non Determinism is transformed into Deterministic Finite Automaton Shown in Figures 3 and 4. The generated DFA are used as the recognizer for the spam detecting pattern.

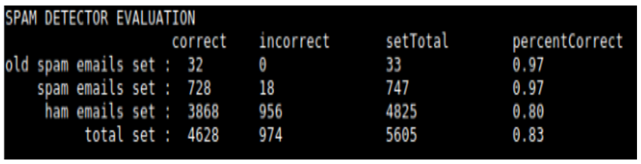
2.3 Email Input Parsing

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 5 for a finite automation that only accepts abc.

Table 1: Search Tags and Regular Expression

Tag	Regular Expression
upperCaseWords	[A-Z][A-Z]+ |[A-Z][A-Z]+,|[A-Z][A-Z]+.
dollarSigns	$+
you	you|YOU
million	(M|m)illion
weightLoss	(L|l)ose *(W|w)eight|(W|w)eight *(L|l)oss
free	FREE|free
win	(W|w)in|WIN|(W|w)on|WON
porn	(P|p)orn|PORN
sex	(S|s)ex|SEX
nude	(N|n)ude|NUDE
jackpot	(J|j)ackpot|JACKPOT
prize	(P|p)rize|PRIZE
discount	(D|d)iscount|DISCOUNT
sale	(S|s)ale|SALE
virus	(V|v)irus|VIRUS
congrats	(C|c)ongratulations|(C|c)ongrats|CONGRATULATIONS|CONGRATS
award	(re|a|RE|A)ward|(RE|A)WARD
urgent	(U|u)rgent|URGENT
client	(C|c)lient|CLIENT
customer	(C|c)ustomer|CUSTOMER
cash	(C|c)ash|CASH
number	[0-9]+
claim	(C|c)laim|CLAIM
private	(P|p)rivate|PRIVATE
exclusive	(E|e)xclusive|EXCLUSIVE
guarantee	(G|g)uarantee|GUARANTEE
apply	(A|a)pply|APPLY
asap	(A|a)pply|APPLY
money	(M|m)oney|MONEY
subscribe	(S|s)ubsribe|SUBSCRIBE
total	*



```
SPAM DETECTOR EVALUATION
                     correct    incorrect      setTotal      percentCorrect
old spam emails set :  32        0              33            0.97
    spam emails set :  728       18             747           0.97
     ham emails set :  3868      956            4825          0.80
          total set :  4628      974            5605          0.83
```

Figure 6: Performance analysis pf the spam detector software.

3 PERFORMANCE EVALUATION

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 only 80% accuracy on detecting ham emails. The total combined accuracy of our spam detector is 83%.

4 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 it is not.

Presented is a method for detecting emails as spam or ham using regular expression and finite automatons in order to recognize a spam and a ham email. The accuracy is relatively low with only 83% combined. And the low accuracy is the effects 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.

5 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] [n. d.]. 6.1 - Introduction to Generalized Linear Models. https://onlinecourses.science.psu.edu/stat504/node/216/
[2] [n. d.]. SMS Spam Collection Dataset. https://www.kaggle.com/uciml/sms-spam-collection-dataset
[3] [n. d.]. UCI Machine Learning Repository: Spambase Data Set. https://archive.ics.uci.edu/ml/datasets/Spambase
[4] James Power. 2002. Converting a regular expression to a NFA - Thompson's Algorithm. http://www.cs.may.ie/staff/jpower/Courses/Previous/parsing/node5.html