Improving Adaptive Filtering of Spam

Author: Paul Ahern (87227070)

1 April 2007

Supervisor: Gregory Provan

Module: CS5205

Report version: 1.0

Department: Computer Science, UCC

Declaration: I confirm that, except where indicated through the proper use of citations and references, this is my own original work and that I have not submitted it for any other course or degree.

Abstract

Bayesian classification of incoming emails is the basis of most modern Spam filters. This is a highly successful approach, in part because each filter learns the requirements of the individual user. Having to attack many different filters makes the Spammers' job harder.

This project models the behaviour of general classifiers to determine by how much their performance can be improved by the addition of email-specific rules. Specifically a Whitelist of email addresses from whom receiving Spam is unlikely.

Naive Bayes, Markovian and Hyperspace classifiers are tested and analysed.

Contents

Chapter 1

Introduction

This project uses two separate Bayesian Processes:

1. Two Bayesian, as well as one non-Bayesian, classifiers.
2. A Bayesian Inference Network which uses probabilities derived by running the classifiers against a corpus of email.

Bayesian classifiers work by counting the numbers of tokens in sets of data which have already been classified. The numbers of each token in the set determines the extent that the presence of that token implies membership of the set. New data is classified on the basis of the numbers of known tokens contained within it.

Spam filtering is a specific example of the classification problem. An adaptive Spam filter classifies emails into the categories of Spam and Non-Spam and updates its lists of tokens on the basis of user input. The filter can be updated when an incorrect classification occurs or when a classification is implicitly confirmed by the user deleting an email classified as Non-Spam and not marking it as Spam.

Naive Bayes operates on the level of individual words and take no account of the specific structure of email data. This is quite effective, but the addition of an email-specific heuristic may improve performance further.

For instance many email client applications maintain a list email addresses to which mail has been sent by the user. This can be used as a Whitelist - a list of trusted sources who are unlikely to send Spam.

Email recently (Jan-Mar 2007) received by a single email account data is been analysed to determine how much of it is Spam. Three classifiers are trained on a subset of this data and then used to classify the remainder. The email data was also checked to see how much was from known addresses.

Chapter 2

Background

The use of Bayesian classification on this problem was suggested in [Graham (2002)] and further elaborated on in [Graham (2003)]. Other researchers have explored the problem with generally positive results (see [Manco (2002)], [Meyer (2004)], [Oda (2003)] and [Pelletier (2004)]).

Spam filtering is an arms race between the designers of the filters and the designers of the Spam. Filter effectiveness declines as Spam is created to attack it. For instance, filter poisoning occurs by sending Spam with many attributes of Non-Spam. As these emails are flagged as Spam by the user the filter learns to associate these attributes with Spam.

CRM114 is a system for classifying data. It includes a language which can be used to apply specific rules (Regular Expressions). Three classifiers already implemented in this language are:

1. Naive Bayes.
   Which uses single works as tokens.

2. Markovian.
   Which uses chains of up to five words as tokens.

3. Hyperspace.
   A Non-Bayesian classifier which uses the k-Nearest Neighbours, whose positions are plotted in a hypercube, to determine the classification of new data.

More detailed descriptions of these and other classifiers can be found in [Yerazunis (2006)] which is available from the CRM114 website [CRM114].

Chapter 3

Details

This project uses data received by a single email account. Only recent email is used to train and test the classifiers. Training and testing is performed on individual emails. This process mirrors the normal experience of filter in the wild.

Statistics are available on how many distinctive features the Bayesian classifiers identified in the email corpus.

A Whitelist of email addresses to which mail has been sent from the account is available. The email data is analysed to determine how many emails, of each type, were received from addresses on the Whitelist.

This email classification system is modelled using a Bayesian Inference Network. The changes in the probability of an incoming email being Spam or Non-Spam, depending on the decision of a classifier and whether or not the sender was on the Whitelist, are recorded.

3.1 Software

This project uses the following software:

Thunderbird (version 1.5.0.10 20070306) Email Client. The source of the email data. Thunderbird stores emails in MBOX format. [Thinderbird 1.5]

IMAPSize (version 0.3.6) Tool to manage IMAP email accounts. Used in this project to convert emails from MBOX to EML format. This gives an individual text file for each email. [IMAPSize]

BASH Scripts Written and run under Kubuntu Linux v6.10. Copies of the scripts used are available from www.ahernp.com/docs/cs5205.zip

CRM114 (version BlameDalkey 20061103) Classifier system and language. [CRM114]

GeNIe (version 2.0.2561.0 20070105) Tool for building Bayesian inference networks graphically. [GeNIe 2.0]

3.2 Test Data

The test data consists of 1,370 emails from a single email account. These were manually divided into four classes:

- training Spam (72 emails)
- training Non-Spam (62 emails)
- testing Spam (1,107 emails)
- testing Non-Spam (129 emails)

There are a total of 1,179 Spam and 191 Non-Spam emails in the data. This means that the prior probability of an email being Spam is 86.1% and the probability of it being Non-Spam is 13.9%.

The emails were checked to see which had been sent from known addresses. None of the Spam emails were from known addresses. 167 of the 191 Non-spam emails were from known addresses. This means that the probability of a Non-Spam email being from a known address is 87.4%.

The training data was used to train the Naive Bayesian Classifier. The testing data to test it.

The emails used for training were received prior to 14 February 2007 and those used for testing between that date and 25 March 2007.

3.3 Process

The following steps are performed:

1. Copy available emails into new folders in Thunderbird. The new folders are called spamTrain, spamTest, nonSpamTrain and nonSpamTest.
2. Count how many Non-Spam emails were from unknown addresses and how many Spam emails were from known ones.
3. Convert the MBOX files spamTrain, spamTest, nonSpamTrain and non-SpamTest directories containing separate EML (text) files for each message using the IMAPSize (option: tools>mbox2eml) tool [IMAPSize].
4. Remove mozilla-thunderbird headers from all EML files. Using BASH script.
5. For each classifier (Naive Bayes, Markov and Hyperspace):
   (a) Train CRM114 classifier using spamTrainC and nonSpamTrainC directories. Using BASH script.
   (b) Classify the contents of the spamTestC and nonSpamTestC directories with the newly trained classifier. Using BASH script.
   (c) Check if each email in the test sets has been classified correctly or not.
   (d) Use the numbers of emails to calculate probabilities for the Bayes Inference Network in Genie 2.0 [GeNIe].

(e) Update the evidence in the Classifier and Whitelist nodes and note the changes in probabilities of the rest of the network.

The BASH scripts and CRM114 programs used are available for download from www.ahernp.com/docs/CS5205.zip.

Use them by unzipping to a directory. Create sub-directories called spamTrain, spamTest, nonSpamTrain and nonSpamTest and copy in the test data. Then run reset.sh.

This will create new directories called spamTrainC, spamTestC, nonSpamTrainC and nonSpamTestC which will contain a version of the email data with the Thunderbird headers removed.

Then it will train the filters using the contents of the first two directories and try to classify the contents of the second two. The results of the classifications will be in .log file.

Finally a summary of the Naive and Markovian classification statistics as well as the numbers of correct and incorrect classifications performed by each filter are written to a file called stats.log.

3.4 Results

Obtained from both from the training and testing of the CRM114 filters and from performing inference on the Bayesian Inference Network.

3.4.1 Classifier Training and Testing

Features identified in the Training data by the Bayesian classifiers:

Classifier	Naive	Markovian
Non-Spam features	47,137	754,192
Spam	27,306	436,906
Similarities	2,720	30,938
Differences	36,370	580,774
Similarity Ratio	1:13.4	1:18.8

The Markovian approach identifies far more features than that of Naive Bayes. More significant for the accuracy of the classifier is that it also finds proportionally more differences than similarities between the training sets.

Correct ($\surd$) and incorrect ($\times$) classifications, out of 1,107 Spam and 129 Non-Spam emails respectively:

Classifier	Naive	Markovian	Hyperspace
Non-Spam$\surd$	123	124	122
Non-Spam$\times$	6	5	7
Spam$\surd$	1073	1086	1088
Spam$\times$	34	21	19

Note that the accuracy of the Markovian classifier exceeds that of Naive Bayes at identifying both Non-Spam and Spam; While the Hyperspace (k-Nearest Neighbours) classifier is the most accurate at identifying Spam it also has the worst false positive rate of the three.

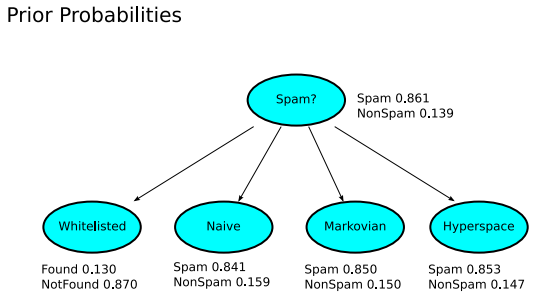

Figure 3.1: Bayesian Inference Network

3.4.2 Bayesian Inference Network

Probabilities assigned in Bayesian Inference Network depending on evidence ($\surd$ yes or $\times$ no):

Classifier	Evidence		Result	
	Spam	Whitelisted	Non-Spam	Spam
			0.139	0.861
Naive	$\surd$		0.008	0.992
Naive	$\surd$	$\surd$	0.405	0.595
Naive	$\surd$	$\times$	$<$0.001	0.999
Naive	$\times$		0.834	0.166
Naive	$\times$	$\surd$	0.998	0.002
Naive	$\times$	$\times$	0.389	0.611
Markovian	$\surd$		0.006	0.994
Markovian	$\surd$	$\surd$	0.359	0.641
Markovian	$\surd$	$\times$	$<$0.001	0.999
Markovian	$\times$		0.891	0.109
Markovian	$\times$	$\surd$	0.999	0.001
Markovian	$\times$	$\times$	0.510	0.490
Hyperspace	$\surd$		0.009	0.991
Hyperspace	$\surd$	$\surd$	0.439	0.561
Hyperspace	$\surd$	$\times$	0.001	0.999
Hyperspace	$\times$		0.101	0.899
Hyperspace	$\times$	$\surd$	0.999	0.001
Hyperspace	$\times$	$\times$	0.531	0.469

Chapter 4

Conclusion

The Naive Bayes approach is surprisingly accurate at classifying current emails into Spam and Non-Spam.

Probabilities of correct ($\surd$) and incorrect ($\times$) classifications:

Classifier	Naive	Markovian	Hyperspace
Non-Spam$\surd$	95.3%	96.1%	94.6%
Non-Spam$\times$	4.7%	3.9%	5.4%
Spam$\surd$	96.9%	98.1%	98.3%
Spam$\times$	3.1%	1.9%	1.7%

As expected the more sophisticated Markovian approach yielded even better results. The Hyperspace (k-Nearest Neighbours) approach is a little less successful at identifying Non-Spam, but was the best method in terms of accurately identifying Spam emails.

The amount of data used in the project and the small spread in the results obtained from the different classifiers means that these results cannot be considered statistically significant.

The model suggests that the addition of a Whitelist rule to the filtering process raises the success rate of all the filters to at identifying Spam and Non-Spam to the 99.9% level, so long as the Whitelist agrees with the decision of the filter.

When a filter classifies input from a Whitelisted address as Spam the certainty drops dramatically. Implementations of this type of filter should let the user decide if such an email was Spam or not.

An additional discovery in the course of this project was that even a Naive Bayes classifier trained on recent emails was more accurate than the adaptive filter built into Thunderbird client software which had been running continuously for a year. Resetting this filter immediately improved its performance. Regular retraining of filters on recent emails would seem to be indicated. Once again this lesson, as a one-off case, cannot be considered statistically significant.

Chapter 5

Appendices

5.1 Procedures

The project's goal is to model the behaviour of adaptive spam filters using a Bayesian Inference Network and to determine to what extent their performance can be improved by the application of a Whitelist of trusted email addresses. Email from a known address is considered unlikely to be spam.

The scripts and programs provided for download at www.ahernp.com/docs/CS5205.zip are used to train three adaptive spam filters on one set of email data and then test their effectiveness by using them to classify another.

5.1.1 Steps

Use the scripts and programs to train and test classifiers:

1. Divide the email data (.EML format) into spam and nonSpam instances.
2. Divide these instances into training and testing sets.
3. From these sets of emails, populate four folders: spamTrain, spamTest, nonSpamTrain and nonSpamTest.
4. Run reset.sh BASH script to build the training and testing environments and to perform the training and testing of the three filters. Press CTRL-D to start each training process.
5. The results of the training and testing are written to stats.log
6. Use the probabilities of successful classification in the Bayesian Inference Net (in Genie 2.0).

5.1.2 CS5205.zip contents

CRM114 programs:

- naiveTrain.crm - Learn input data and build statistics files using naïve Bayesian classification
- naiveClassify.crm - Classify input data using naïve Bayesian statistics files

- markovTrain.crm - Learn input data and build statistics files using Markovian classification
- markovClassify.crm - Classify input data using Markovian statistics files
- hyperspaceTrain.crm - Learn input data and build statistics files using k-Nearest Neighbours classification
- hyperspaceClassify.crm - Classify input data using KNN statistics files

Genie 2.0 model:

- CS5205.xdsl - Bayesian Inference Network

BASH Scripts:

- classify.sh - Classify every file in a directory
- cleanEML.sh - Use grep to remove thunderbird headers
- getStats.sh - Collect numbers of emails and classifications
- reset.sh - Run entire training/classification process
- retest.sh - Rerun testing part of the process
- retrain.sh - Rerun training part of the process
- train.sh - Train filter on every file in a directory

Bibliography

[CRM114] CRM114. crm114.sourceforge.net/, accessed 27 March 2007.

[GeNIe 2.0] GeNIe 2.0. genie.sis.pitt.edu/, accessed 27 March 2007.

[Graham (2002)] P Graham, A Plan for Spam (2002). www.paulgraham.com/spam.html, accessed 27 March 2007.

[Graham (2003)] P Graham, Better Bayesian Filtering (2003). www.paulgraham.com/better.html, accessed 27 March 2007.

[IMAPSize] IMAPSize. www.broobles.com/imapsize/, accessed 27 March 2007.

[Manco (2002)] G Manco, E Masciari, M Ruffolo and A Tagarelli, Towards An Adaptive Mail Classifier (2002).

[Meyer (2004)] TA Meyer and B Whateley, SpamBayes: Effective open-source, Bayesian based, email classification system (2004)

[Oda (2003)] T Oda and T White, Developing an Immunity to Spam (2003).

[Pelletier (2004)] L. Pelletier, J. Almhana and V. Choulakian, Adaptive filtering of spam (2004).

[Thinderbird 1.5] Thunderbird (version 1.5.0.10 20070306). www.mozilla.com/en-US/thunderbird/, accessed 27 March 2007.

[Yerazunis (2006)] W. S. Yerazunis, CRM114 Revealed (2006).