SpamNet – Spam Detection Using PCA and Neural Networks

Abhimanyu Lad

B.Tech. (I.T.) 4th year student
Indian Institute of Information Technology, Allahabad
Deoghat, Jhalwa, Allahabad, India
abhimanyulad@iiita.ac.in

Abstract. This paper describes SpamNet – a spam detection program, which uses a combination of heuristic rules and mail content analysis to detect and filter out even the most cleverly written spam mails from the user's mail box, using a feed-forward neural network. SpamNet is able to adapt itself to changing mail patterns of the user. We demonstrate the power of Principal Component Analysis to improve the performance and efficiency of the spam detection process, and compare it with directly using words as features for classification. Emphasis is laid on the effect of domain specific preprocessing on the error rates of the classifier.

1 Introduction

'Spam' refers to unsolicited e-mail, generally sent in bulk for advertisement. A typical user gets around 10-50 spam mails a day [6], and even careful users can get signed up on unwanted mailing lists. Spam is undesirable because it eats up resources like disk space and user time. Thus we find it worth the effort to design a domain specific classifier for accurately weeding out spam from a user's mailbox.

The nature of spam emails received differs among users, and spam content can also vary with time[6]. The ability to adapt the classification mechanism to the evolving mind of the spammer is one of the prime concerns of any anti-spam software.

This paper introduces SpamNET – a program which uses a combination of heuristic rules, Principal Component Analysis and Neural Networks to produce an accurate spam detection system which is able to adapt to the changing trends of mails received by the user. It re-trains its network every 7 days to discover new patterns in the user's mailbox. SpamNET is a user level program, i.e. it runs as part of the user's mail client rather than sitting on the mail server providing general services to it's users. This allows SpamNET to make more generous use of the user's processing power, rather than being constrained by scalability issues that are inherent to applications running on a server.

The rest of the paper is organized as follows. In section II, we describe the existing state-of-the-art in spam detection. In section III, we describe the architecture of SpamNET in detail. Section IV shows the results obtained by our classifier, and Section V describes the conclusion drawn by us.

G. Das and V.P. Gulati (Eds.): CIT 2004, LNCS 3356, pp. 205–213, 2004.

2 Previous Work

A General Techniques

The stone age of spam detection was marked by the use of simple keyword filters – if a given term was present in the mail's headers or in the body, the message was marked as spam. This method has been found to be non scalable as each potential spam term has to be manually added to the system. Thus no appreciable time was saved on the part of the user [2]. There are several spam filters available today [1]. The most common of them can be categorized as follows:

User defined filters: They allow creation of rules of the form – "If ____ field contains _____, move it to _____ folder." Adding filter rules to combat each new kind of spam mail is often impossible.

Header filters: These analyze the headers to detect whether the headers are forged. But many spam mails have valid headers and therefore such filters are bound to fail for a large number of spam mails.

Content filters: These scan the body of the email message and use the presence, absence and frequency of words to determine whether the message is spam or not. Content based filtering has been found to be most effective for spam detection.

Two well established techniques – Bayesian classification and Neural Networks have been widely applied to the problem of classifying mails as spam or non-spam.

B Bayesian Classification

Bayesian classification has been found be an effective technique which makes use of prior knowledge about the 'training' samples to make intelligent decisions about 'unseen' or 'test' samples. Moreover, it is able to mathematically assign to each mail a probability of being spam.

One of the earliest reports on the use of Bayesian classification for spam filtering is [3] where Pantel and Lin describe the SpamCop program. They were able to achieve 92% detection rate and 1.16% rate of misclassifying non-spam mails. A message is classified as spam if

$$P(\text{Spam}|M) > P(\text{Non-Spam}|M).$$

Also, they assume the conditional independence of attributes in a message.

Bayesian classification and its application to spam classification has been described in [8]. Better detection rates have been reported by incorporating domain specific knowledge into the Bayesian classifier [4].

C Neural Networks

Neural Networks are able to discover rules which otherwise are difficult for humans to describe or even comprehend. They not only provide an easy way to model complex relationships between input and output, but also provide adaptability and learning ability implicitly.

Various types of neural networks have been tried to classify mails as spam or non-spam. [6] describes a mass of fully connected neurons used for classifying emails. It

uses no layers and no formal structure. Every neuron in the system has have access to all the inputs to the network and the outputs from every other neuron. The network is able to control the window of input it wishes to read next. Experiments show that preprocessing and choice of features is more important towards the performance of classification than the architecture of the neural network itself.

3 SpamNet

A Architecture

SpamNET categorizes emails using a Neural Network. The feed forward neural network is trained using the backpropagation algorithm. It uses a single hidden layer with 10 neurons. The 'tansig' transfer function is used both for the hidden and output layer.

The neural network takes input from the Extractor module which extracts words (e.g. free, hi, meeting, presentation, assignment etc.) as well as concepts (e.g. CONTAINS_INVISIBLE_TEXT) from the mails. While concepts are directly fed to the neural network in a quantified form, words appearing in the mail are first converted to a feature vector using a volatile vocabulary, then passed on to the PCA module before supplying it to the Neural Network. The neural network gives an output between -1 and 1. While an output of 1 represents an almost certain spam mail, -1 represents an almost certain non-spam mail. The 'concepts' provide an additional level of detection ability apart from the vocabulary based approach.

After every 7 days, SpamNET re-trains itself to capture the varying trends of the user's incoming mail. In training mode, the extractor module constructs a vocabulary of all words appearing in the mails, then creates a term document matrix (which depicts the frequency of occurrence of words in different mails) and the PCA module finds the optimal representation of this feature space by computing the eigenvectors for the term document matrix. Then the neural network is trained using the transformed feature vectors. In the testing mode, the transformed vector is used as input to the neural network, which then classifies the mail as spam or non-spam.

SpamNet also has a "short circuit" path to the output which directly classifies the mail as spam or non-spam without going through the usual path mentioned above. E.g. mails coming from addresses to which the user has replied or has composed a mail to in the past invoke this "short circuit" path and are thus are unconditionally marked as non-spam.

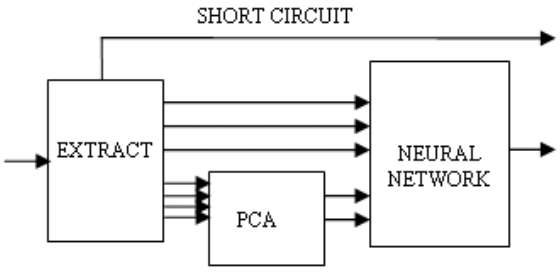

Fig. 1. Architecture of SpamNET

B Preprocessing: Extractor Module

The Extractor module is responsible for converting a mail into a form that is useful to the classifier.

The module extracts two kinds of information from the email – 1) Words 2) Concepts. The Extract module is HTML and CSS aware, and is case sensitive. Moreover, it prefers to read the HTML version of the mail when multiple MIME formats are available in the received mail. The extractor parses the message according to the following rules, and also makes note of various abnormalities, as mentioned below:

1. Invisible text is detected. This includes:
   a. Text of zero size.
   b. Text in background color or with a foreground color very close to that of the background.
   c. Text between HTML tags which will not be shown by the email client.

   Such text is meant to confuse the spam detector. E.g. a clever spammer may put in some non-spam words e.g. conference, meeting, urgent or may be even pick up the latest story on Yahoo! News and enclose it between appropriate tags such that it is invisible to the user but will be picked up by the spam detection program and used for making a decision about the mail. Presence of invisible text in the mail triggers the short-circuit path and the email is immediately marked as spam.
2. Symbols such as *, #, % etc. are stripped if they occur between alphabetic letters. Thus v*i*a#g#r#a gets converted to viagra. Even spaces occurring between single letters (except valid single letters like ‘a’ and ‘I’) are detected and stripped off. At the same time, the extractor module makes note of special punctuations like “!!!”, before stripping them off.
3. Empty tags e.g. which appear between otherwise non-delimited text are ignored and the surrounding pieces of text joined together. Thus Viagra gets converted to Viagra.
4. Currency values like. $100,000,000 and $2 million are detected & noted.

Removal of Stop Words

Words like articles, prepositions, conjunctions, common verbs (e.g. ‘know’, ‘see’, ‘do’, ‘be’), auxiliary verbs, adjectives (e.g. ‘big’, ‘late’, ‘high’), and pronouns are removed, leaving only content words likely to have some meaning. This process condenses the vocabulary and thus reduces the computational cost of further processing on it.

Stemming

The words are passed through a stemmer which reduces multiple instances of a single word to the root word. E.g. flying and flied are reduced to fly. This stemming step is found to bias the classifier towards reducing false positives (non-spam mails being marked as spam) as well as improve the learning ability of the classifier in general. This behavior can be explained as follows. Suppose “paint” happens to a non-spam word for a particular user, then “painting”, “painted” and “painter” should automatically become “non-spam” words without requiring explicit training with all such variants of a single word. This will essentially protect new non-spam mails containing such variants of a non-spam word against getting classified as spam. On the other

hand, spam words like "viagra", "rich" usually occur in the same form in all spam mails and even their variants would not usually occur in non-spam mails. Therefore non-spam mails getting classified as spam due to stemming is very rare as compared to the other way round.

The Term-Document Matrix (TDM)

The Term Document Matrix M stores information about the frequency of occurrence of words in each mail. Words are listed on the vertical axis, and mails on the horizontal axis. Mi,j represents the frequency of ith word in the jth mail. This information helps in building the set of active and dormant features, which is explained in the section on Volatile Vocabulary. The feature vectors representing individual mails are column vectors in the term document matrix. These vectors are normalized so that classification is not affected by the length of the documents.

Feature Selection

(Difference of Means Method)

We use the frequency of certain words in the mails as inputs to the PCA module, which then feeds a transformed feature vector to the neural network. These words are chosen according to their ability to distinguish between the two classes of mails we are interested in, i.e. spam and non-spam. It must be appreciated that such words may vary from user to user. e.g although presence of words like "Viagra" is almost certainly enough to classify mails as spam, the same might not be true for "free", if the user deals with free software. Therefore these words must be chosen according to the specific user.

One technique used to find such discriminatory words is the difference of means method. For each feature, the difference in means of that feature in the "spam mails" class and that in "non-spam mails" class is calculated. Features having higher differences are potentially better attributes for distinguishing samples into the two classes.

Volatile Vocabulary

This part of the Extractor module is responsible for learning the user profile and adapting to the changing patterns of mails received by the user. When the extractor is working in training mode, it creates a combined vocabulary of all the words that appear in the spam and non-spam mails. This vocabulary is not only large but also contains words which play no important role in classification. In fact, a proper selection of features can actually improve the classifying and generalizing ability of the classifier. For choosing such words which have high discriminating power for the classification problem at hand, we use the difference of means method described above. Words whose difference of means for the two classes is greater than 1/7th of the highest difference of means are marked as active i.e. they will be used as features for classifying new mails. Words which do not satisfy this criterion are either too unique to help in future classification, or they are words which appear in almost equally in both spam and non-spam mails and hence are too common to help in classification. Instead of deleting these words, they are marked as dormant – since they may become active in the future. E.g. if the user joins a new mailing list that deals with Linux, initially the word "Linux" will be added to the vocabulary but marked as dormant. Over the time, the user will receive more mails from the list which will cause the word "Linux"

to contribute heavily to the non-spam class, and hence eventually become an active word. Words that stay in the dormant list for more than 28 days (i.e. four training cycles) are removed, which prevents the vocabulary from growing infinitely large.

When in testing mode, the extractor module finds words in the message which can fall into one of the three categories – active, passive or dormant. Only the active words are used to construct the feature vector. It must be pointed out that the size of the vocabulary does not necessarily increase with time, since words like "free" may start appearing in non-spam mails when the user joins a free software mailing list, and eventually the word "free" will lose its discriminatory power and hence get removed.

C PCA Module

PCA is a useful statistical dimensionality reduction technique that has found application in fields such as face recognition and image compression, and is a common technique for finding patterns in data of high dimension. An introduction to PCA can be found in [5].

The feature vectors constructed in the previous step are used to compute the eigenvectors using Singular Value Decomposition. The top two eigenvectors are selected (having the greatest eigenvalues) which are then used to transform input vectors to the reduced vector space.

The features that make up the input vector can be strongly correlated with each other. It is generally desirable to find or reduce the feature set to one that is minimal but sufficient. This can be done by judicious elimination or by the Principle Components Analysis (PCA) reduction technique amongst others. PCA can be used to reduce the feature vector dimension while retaining most of the information by constructing a linear transformation matrix. The transformation matrix is made up of the most significant eigenvectors of the covariance matrix.

Upon experimentation, it was found that using more than two dimensions (i.e. the first two eigenvectors arranged according the their eigenvalues) reduced the distinctiveness of the clusters and hence the performance of the classifier.

The reduction in the dimensionality causes a reduction in the number of inputs to the neural network which makes it more efficient. Another implication of using PCA is that the number of inputs to the neural network is made independent of the current size of the vocabulary. The number of words in the vocabulary may increase or decrease, but it will always result in only two inputs to the neural network, since we are using only the first two eigenvectors.

The use of PCA as a preprocessing step not only improves efficiency, but has also shown a consistent decrease in the error rate of the classifier. We also tried the difference of means method to select the top 35 (highest difference in means) out of a set of 403 attributes, but applying PCA to reduce the 403 dimension space to a two dimensional space was found to be invariably better at detecting new spam mails than using the 35 dimension space. Moreover, the performance of the classifier when PCA is used is found to be fairly stable against changes in size as well as nature of the vocabulary. On the other hand, when only the difference of means method is used without PCA, the error rate of the classifier varies drastically when the number of features is changed.

D "Short Circuit" Rules

Apart from using the presence or absence of words as criteria for making a decision, we find that the use of domain specific information can greatly decrease the error rate – especially the false negatives (spam mails being marked as non-spam), caused by cleverly crafted spam mails.

The presence or absence of certain features immediately ascertains whether a mail is spam or not. In such cases, a decision can be made directly i.e. by completely bypassing the PCA and neural network modules. We call these domain dependent heuristics as short circuit rules.

E.g. If a mail appears to come from a person whom the user has replied to or composed a mail to in the past, then the mail is considered to be non-spam. However, if the user replies to a mail that was marked with a spamness value of greater than .8, the user is warned and asked whether the reply-to address should be marked as a non-spam address.

The Extractor module can detect obfuscated words e.g. v*i*a*g*r*a. If the obfuscated word happens to be on the active list in the vocabulary and having a higher mean frequency of occurrence in spam samples than non-spam samples, the mail is immediately categorized as spam. This scheme works very well for detecting spam mails since non-spam mails will almost never have obfuscated words, especially words that are known to appear in spam mails. Such a scheme also prevents mail from a friend containing "M-A-I-L-M-E" to be classified as spam due to a short circuit rule.

Presence of invisible text, as mentioned in the section of Extractor Module, also triggers the short-circuit and the mail is immediately marked as spam.

E Neural Network Inputs

The neural network used by us has 6 inputs. While two of these connect to the output of the PCA module, the rest 4 capture outputs of the following heuristic criteria:

1) Number of strong punctuators, e.g. "!!!"
2) Percentage of capital words found in the mail.
3) Number of currency values found.
4) Percentage of colored text in the mail.

4 Results

The SpamNET program was trained using a spam database available on the Internet [7]. Mails, especially non-spam, were obtained from the mailing list of the third year I.T. students at our institute. The system was trained using 100 spam and 100 non-spam mails, and tested using 200 spam and non-spam mails each.

Fig. 2 shows the output of the neural network for 200 spam (red) and 200 non-spam (blue) mails. 2 spam and 2 non-spam mails were misclassified. Moreover, 1 spam mail and 1 non-spam mail can be seen to be on the verge of misclassification.

False Negatives

Mail 1 is a mail from SCazzz@aol.com which goes as "I am looking for a friend named…". It does not appear to contain any spam like words. Mail 2 is a mail from

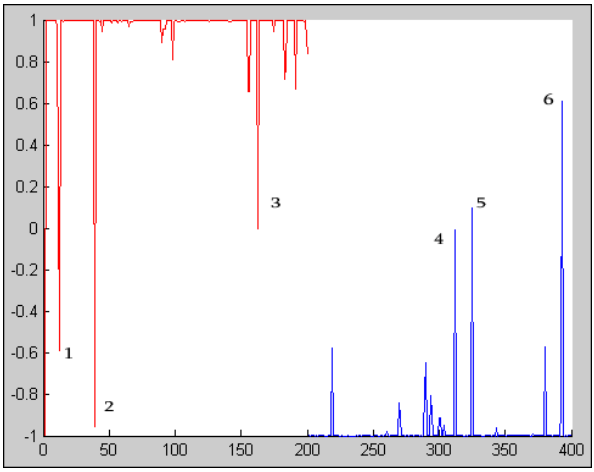

Fig. 2. Neural Network output

careerm@netexecutive.com and posed as a recruitment consultant. Although it mentioned a "large database of recruiters" which would indicate it is a spam, but terms like 'database' (and other terms found in that mail like 'resume', 'internet' etc) are usually discussed on our mailing list, and thus did not help in classifying the mail as spam. Mail 3 had many words in capitals, but almost all were innocent looking and thus it was difficult for the classifier to decide on the nature of the mail.

False Positives

Mail 4 and 5 were ACM newsletters which contained repeated references to 'business' and also contained the phrase "casualty-free". Moreover, the ACM mails made repeated use of FONT tags, while almost all non-spam mails used for training were plain text. Mail 6 described SCO's suit against IBM. It repeatedly talked of phrases like '$3 billion suit', '$5 million per quarter', 'Linux….free in its basic form' which the classifier found too tempting to ignore.

Table 1 describes detection rates obtained under various configurations.

Table 1. Average error rates obtained by multiple cross-validations over a set of 291 spam and 317 non-spam mails

Eigenvectors (PCA)	Stop wordlist used	Stemming used	Case sensitivity	Size of vocabulary	False Negatives	False Positives
2	Y	Y	Y	403	1.5%	1.3%
3	Y	Y	Y	403	2.0%	2.0%
4	Y	Y	Y	403	2.1%	2.0%
2	N	Y	Y	403	1.8%	1.2%
2	Y	N	Y	403	3.0%	4.0%
2	Y	Y	N	530	1.5%	1.5%

5 Conclusion

The traditional methods of filtering which looked for particular words and phrases have given way to more intelligent filtering techniques like Bayesian classifiers and

Neural Networks. Although Bayesian classification has been reported to provide over 99% accuracy for spam detection, neural networks have shown good potential for adaptive spam detectors. Irrespective of the classifier used, the preprocessing step has a profound effect on the performance of the classifier. Domain specific preprocessing which takes into the account the structure of the email together with the ability to handle mail headers and HTML content is found to have a positive effect on the classifier performance. Moreover, the use of dimensionality reduction has been experimentally found to hold a promising alternative to manually selecting good features for the classifier.

We are motivated by the misclassifications obtained by us to try the potential of semantic analysis to capture the context in which a word is used. The word 'database', when used with 'email', e.g. 'Email Database', or 'Database of Emails' should have a higher contribution towards classifying the mail as spam, than the individual occurrence of 'database' as well as 'email'.

References

1. 'White Papers – Spam Tutorial' – VicomSoft. http://www.spambolt.com/anti_spam_faq/ email_spam_filter.html
2. Sivanadyan, Detecting spam using Neural Networks. http://www.cae.wisc.edu/~ece539/project/f03/sivanadyan.pdf
3. Patrick Pantel and Dekang Lin. ``SpamCop—A Spam Classification & Organization Program.'' Proceedings of AAAI-98 Workshop on Learning for Text Categorization.
4. Paul Graham, at the 2003 Spam Conference http://www.paulgraham.com/better.html
5. Lindsay I Smith. A tutorial on Principal Components Analysis http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
6. Martin, Spam Filtering Using Neural Networks http://web.umr.edu/~bmartin/378Project/report.html
7. The Great Spam Archive (Online spam database) http://www.annexia.org/spam/
8. Androutsopoulos, I. Koutsias, J., Chandrinos, K. V., Paliouras, G., and Spyropoulos, C. D. An Evaluation of Naïve Bayesian Anti-Spam Filtering. In Proceedings of Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, Barcelona, 2000.