Journal of Information Assurance and Security 4 (2009) 151-160

Online Self-Organised Map Classifiers as Text Filters for Spam Email Detection

Bogdan Vrusias[1] and Ian Golledge[2]

[1]University of Surrey, Department of Computing,
Guildford, Surrey, GU2 7XH, UK
b.vrusias@surrey.ac.uk

[2] University of Surrey, Department of Computing,
Guildford, Surrey, GU2 7XH, UK
ian_golledge@hotmail.com

***Abstract*:** Email communication today is a way of working and communicating for most businesses and public in general. Being able to efficiently receive and send emails therefore becomes a must. Spam email detection and removal then becomes a vital process for the successful email communications, security and convenience. This paper describes a novel way of analysing and filtering incoming emails based on the text (keyword) salient features identified within. The method presented has promising results and at the same time significantly better performance than other statistical and probabilistic methods and at the same time offers a mechanism that can automatically adapt to new (unseen) email trends. The salient features of emails are selected automatically based on functions combining word frequency and other discriminating matrices, and then encoded into appropriate numerical vector models. The method is compared against the state-of-the-art Multinomial Naïve Bayes, Support Vector Machines and Boosted Decision Tress classifiers for identifying spam. The proposed automatic adaptable feature extractor method and online Self-Organising Map seems to give significantly better results, with the minimal cost.

***Keywords*:** Spam Detection, Self-Organising Maps, Multinomial Naïve Bayesian, Support Vector Machines, Boosted Decision Tress, Adaptive Text Filters.

1 Introduction

Typically a user receives on average between 20 to 100 spam emails per day. This number increases year by year and it brings with it a whole series of problems to a network provider and to an end user. Networks as a whole are flooded every day with millions of spam emails wasting network bandwidth while end users suffer with spam engulfing their mailboxes. Users have to spend time and effort sorting through to find legitimate emails, and within a work environment this can considerably reduce productivity. On average 5%-10% of spam emails manage to escape the commercial filters that are installed on the email server or even on the individual computers. Realistically this can take an individual even up to 90 minutes per day "cleaning" the mail box, costing massive amounts of money and time to companies or everyday users. Spam emails increase at more than 1% per day, and this indicates that not only better filters need to be developed, but also there is a need for an automatically adaptable filter to detect new spam emails.

Over the last decade many anti-spam filter techniques have been developed to achieve significantly good results [3]-[8], [12], [14], [18], [19], but the issue was always that these models are built usually manually and need to be rebuilt to accommodate new spam emails. The overall premise of spam filtering is text categorisation where an email can belong in either of two classes: *Spam* or *Ham* (legitimate email). Text categorisation can be applied here as the content of a spam message tends to have few mentions in that of a legitimate email. Therefore the content of spam belongs to a specific genre which can be separated from normal legitimate email.

Spam is not only related to emails, but other forms of text as well. Recently spam websites have also intruded into our personal lives and now is one of the major headaches [21]. The concept though remains the same: spam text can be detected based on its context. Original ideas for filtering focused on matching keyword patterns in the body of an email that could identify it as spam [9]. A manually constructed list of keyword patterns such as "cheap Viagra" or "get rich now" would be used. For the most effective use of this approach, the list would have to be constantly updated and manually tuned. Overtime the content and topic of spam would vary providing a constant challenge to keep the list updated. This method is infeasible, as it would be impossible to manually keep up with the spammers.

Sahami et al. [5] is the first to apply a machine learning technique to the field of anti-spam filtering. They trained a Naïve Bayesian (NB) classifier on a dataset of pre-categorised ham and spam. A vector model is then built up of Boolean values representing the existence of pre-selected attributes of a given message. As well as word attributes, the vector model could also contain attributes that represent non-textual elements of a message. For example, this could include the existence of a non-matching URL embedded in the email. Other non-textual elements could include whether an email has an attachment, the use of bright fonts to draw attention to certain areas of an email body and the use of embedded images, all could be possible spam features.

Metsis et al. [3] evaluated five different versions of Naïve Bayes on a particular dataset. Some of these Naïve Bayesian versions are more common in spam filtering than others.

Received June 10, 2009

The conclusion was that the two Naïve Bayes versions used least in spam filtering provided the best success. These are a Flexible Bayes method and a Multinomial Naïve Bayes (MNB) with Boolean attributes. The lower computational complexity of the MNB provided it the edge. The purpose of their paper is not only to contrast the success of five different Naïve Bayes techniques but to implement the techniques in a situation of a new user training a personalized learning anti-spam filter. This involved incremental retraining and evaluating of each technique. Naïve Bayes have also been equally successful in identifying spam websites [21].

Furthermore, methods like Support Vector Machines (SVM) have also been used to identify spam [12]-[14]. Joachims [13] was the first to present the idea of using the technique in spam filtering and explained why they would be suitable. Specifically, term frequency with boosting trees and binary features with SVM's had acceptable test performance, but both methods used high dimensional (1000-7000) feature vectors.

The use of decision trees for text categorisation appears popular in the literature [17] and has also been used as a comparative technique for the purposes of spam filtering [12]. Boosting is also a popular technique in text categorization [15] and has shown improved results over non-boosted techniques. Boosting trees have been applied to spam filtering against base-line techniques and have also shown a strong application in this field [19], [18].

Another approach is to look into semantics. Youn & McLeod introduced a method to allow for machine-understandable semantics of data [7]. The basic idea here is to model the concept of spam in order to semantically identify it. The results reported are very encouraging, but the model constructed is static and therefore not adaptable. The future of semantics lies not only on detection of spam, but also on the prevention. Kassoff et al [23] proposed a new way of annotating and describing emails, called Semantic Email Addressing (SEA), where emails are defined by a semantic layer in order to automatically communicate with the server and mail clients and negotiate its delivery.

Finally, the latest machine learning approaches, such as the ant colony optimisation algorithm [22], have shown comparable results to NB techniques, but the issue with such supervised techniques is that most parameters are hard tuned and costly to reset for new datasets. Having to manually adjust the training parameters is not an option for constantly changing emails.

Most of the above proposed techniques struggled with changes in the email styles and words used on spam emails. Therefore, it made sense to consider an automatic learning approach to spam filtering, in order to adapt to changes. In this approach spam features are updated based on new coming spam messages. This, together with a novel method for training online Self-Organising Maps (SOM) [2] and retrieving the classification of a new email, indicated good performance. Most importantly, the method proposed only misclassified very few ham messages as spam, and had correctly identified most spam messages. This exceeds the performance of other probabilistic approaches, as later proven in the paper.

2 Spam Detection Methods

As indicated by conducted research one of the best ways so far to classify spam is to use probabilistic models, i.e. Bayesian [3], [5], [6], [8], [9]. For that reason, this paper is going to compare the approach of using SOMs to what appears to be best classifiers for spam, the Multinomial Naïve Bayes (MNB) Boolean, Support Vector Machines (SVM) and Boosted Decision Tress (BDT) classifier. All approaches need to transform the text email message into a numerical vector, therefore several vector models have been proposed and are described later on.

2.1 Classifying with Multinomial NB Boolean

The MNB treats each message d as a set of tokens. Therefore d is represented by a numerical feature vector model x. Each element of the vector model represents a Boolean value of whether that token exists in the message or not. The probability of $P(\mathrm{x}|\mathrm{c})$ can be calculated by trialling the probability of each token t occurring in a category c. The product of these trials, $P(t_i|c)$, for each category will result in the $P(\mathrm{x}|\mathrm{c})$ for the respective category. The equation is then [6]:

$$\frac{P(c_s)\cdot\prod_{i=1}^{m}P(t_i\mid c_s)^{xi}}{\sum_{c\in\{c_s c_h\}}P(c)\cdot\prod_{i=1}^{m}P(t_i\mid c_s)^{xi}}>T \quad (1)$$

Each trial $P(t|c)$ is estimated using a Laplacean prior:

$$P(t\mid c)=\frac{1+M_{t,c}}{2+M_c} \quad (2)$$

Where $M_{t,c}$ is the number of training messages of category c that contain the token t. M_c is the total number of training messages of category c. The outcomes of all these trials are considered independent given the category which is a naïve assumption. This simplistic assumption overlooks the fact that co-occurrences of words in a category should not be independent, however this technique still results in a very good performance of classification tasks.

2.2 Classifying with Support Vector Machines

From early research in spam filtering Support Vector Machines (SVM) were commonly used techniques and demonstrated successful results. Joachims [13] outlined some reasons why SVM work well for text categorization, in particular spam filtering. More specifically he argued that SVM have the ability to handle large feature spaces. It could be assumed that some features in an input space are irrelevant; however it is known in text categorisation that even lower ranked features can be useful. Consequently a good performing classifier should take into account as many features as possible and this lends itself to SVM. SVM also

have the ability to cope with problems that features sparse data. High dimensional vector representations of emails will result in few entries being non zero. However evidence in research suggests that algorithms like SVM are able to handle these problems. It can be assumed that most text classification problems are linearly separable. Considering ham and spam, the keyword based features of the two documents should be distinct, therefore represented as a vector they can be linearly separable. SVM are based on finding these linear divisions consequently the algorithm should be well suited to the task of spam filtering.

The SVM algorithm maps a non-linear instance, for example an email vector onto a new space which can be separated by a straight line. This straight line will not look straight in the original vector space. The new space which the SVM uses is called a maximum margin hyperplane. Considering the classes ham and spam that are linearly separable, the maximum margin hyperplane is a line that can find the greatest separation between ham and spam. As mentioned previously, examples in [12]-[14] all show good results in spam filtering from SVM. Therefore this model is important to consider in this project as a comparative classifier.

2.3 Classifying with Boosted Decision Tress

Decision Tress algorithms attempt a divide and conquer approach to classification. Working top down from the feature set of a vector, the algorithm selects the feature that best divides the classes and selects this as the root node. Branches from the root node reflect possible values of this attribute. The problem is then split on the second best attribute, and this recursive algorithm develops a tree like structure of all the features. Each leaf node will have a respective class value and the tree essentially forms a set of rules. Test inputs follow the tree down selecting between branches based on its own features to reach a decision leaf node to associate a class. As described previously this form of classification has been used often in text classification. However in spam filtering the use of boosting alongside decision trees appears to be a more common method.

Boosting was first presented by Schapire [16] and is intended to improve the performance of any learning algorithm. The algorithm maintains training examples in a dataset with an importance weighting. Over training, input vectors that appear easy to classify are given a lower weighting, vectors that are harder to classifier are given a higher weighting. The boosting algorithm then uses these associative weights to force the classifier to concentrate on these harder examples. Example implementations of the boosting algorithm can be found in [15], [16], [18].

Some papers studied applied this boosting algorithm to decision tree classifiers upon a spam filtering classification problem. Ali and Xiang [18] for example applied a boosting algorithm to a particular model of decision tree, J48. By comparing results using the decision tree classifier with and without the boosting algorithm there was a notable increase in performance. These results along with others mentioned in literature show the successful application of this method to spam filtering and this method is important to be considered in this paper.

2.4 Classifying with Self-Organising Maps

Self-organising map (SOM) systems have been used consistently for classification and data visualisation in general [2]. The main function of a SOM is to identify salient features in the *n*-dimensional input space and squash that space into two dimensions according to similarity. Despite the popularity, SOMs are difficult to use after the training is over. Although visually some clusters emerge in the output map, computationally it is difficult to classify a new input into a formed cluster and be able to semantically label it. For the classification, an input *weighted majority voting* (WMV) method is used for identifying the label for the new unknown input [12]. Using this WMV technique, the SOM is now highly suitable for spam filtering. The classifier can be presented with email vectors for training to form clusters of similar vectors. Labelling of the new inputs is based on the closest distance of each input vector from the node vector that was given a label during the training process. After labelling, new incoming test emails can be classified into ham or spam.

For the proposed process for classifying an email and adapting it to new coming emails, the feature vector model is constructed based on the first batch of emails and then the SOM is trained online on the first batch of emails with random initial weights. Then, a new batch of emails is appended, the feature vector model is recalculated, and the SOM is retrained from the previous weights, but on the new training batch only. Finally, more batches of emails are continuously inserted and the process is repeated until all batches are finished.

For the purpose of the experiments, as described later, a 10x10 nodes SOM is trained for 1000 cycles, where each cycle is a complete run of all inputs. The learning rate and neighbourhood value is started at high values, but then decreased exponentially towards the end of the training [11]. Each training step is repeated several times and results are averaged to remove any initial random bias.

The classifiers already mentioned and researched in this project, MNB, SVM and Boosted Decision Trees, provided good benchmark models for comparison. There is evidence of consistent performance of these models, and it would be interesting to contrast a SOM classifier against them.

3 Identifying Salient Features

The process of extracting salient features is probably the most important part of the methodology. The purpose here is to identify the keywords (tokens) that differentiate spam from ham. Typical approaches so far focused on pure frequency measures for that purpose, or the usage of the *term frequency inverse document frequency* (*tf*idf*) metric [10]. Furthermore, the *weirdness* metric that calculates the frequency ration of tokens used in special domains like spam, against the ratio in the British National Corpus (BNC), reported accuracy to some degree [1], [11].

3.1 Identifying Salient Keywords

This paper uses a combination function of the *weirdness* and a modified version of the *tf*idf* metrics; where both metrics are used in their normalised form. The *weirdness* metric compares the frequency of the token in the spam domain against the frequency of the same token in BNC:

$$weirdness_{ts} = \frac{f_{t_S} / N_{t_S}}{f_{t_{BNC}} / N_{t_{BNC}}} \quad (3)$$

The weirdness of spam token t_s is calculated by dividing the frequency f of the token by the total number of token frequencies in the spam set N_{ts}. This is then divided by the frequency of the token in the BNC divided by total number of token frequencies in the BNC N_{tBNC}. Tokens with high weirdness values represent tokens that occur less frequently in the BNC, resulting in unusual tokens, or otherwise tokens that "everyday" British language is not frequently uses. In terms of email words, this will find spam and ham words which are considered less common in natural language, and are distinguishable. This value can therefore aid the extraction of important words from an email dataset and at the same time removing the most common words, such as conjunctions, pronouns, interrogatives, prepositions or other common part of speech words.

For *tf*idf* the "document" is considered as a category where all emails belonging to that same category, spam or ham, are merged together, and document frequency is the total number of categories (i.e. 2 in this instance: spam and ham). In order to have a fair comparison between the different batches and new emails presented, the normalised *tf*idf* is considered:

$$tf * idf_{t_S} = \frac{f_{t_S} \times \log_2 \frac{N}{n_t}}{\sqrt{\sum_1^{t_{S,H}} \left(f_{t_{S,H}} \times \log_2 \frac{N}{n_t} \right)^2}} \quad (4)$$

The *tf*idf* value for a particular spam word token t_s would be calculated by finding the product of the frequency f of that term and the inverse document frequency where N is the number of documents (i.e. two, spam and ham) and n_t is the number of documents that token occurs in (either one or two). This value is then divided by the sum of all tokens in the both ham and spam documents. A high *tf*idf* value demonstrates that a particular token is frequent in a particular document but infrequent considering the whole document dataset. Therefore, this term weighting technique can highlight spam words which occur less frequently in ham emails to help identify features to distinguish between the two classes.

Both weirdness and *tf*idf* statistical measurements can provide an information gain for selecting keyword features for email vectors. The product of the two normalised statistical values has been found to be a good metric for ranking the keywords. The ranking R_t of each token is therefore calculated based on:

$$R_t = weirdness_t \times tf * idf_t \quad (5)$$

The rating metric R is used to build a list of most salient features in order to encode emails into binary numerical input vectors. Salient spam features are the words that most frequently appear in spam emails and are not as common in the general language. Some of these keywords can be seen in Figure 1.

SPAM EMAIL	HAM EMAIL
Subject: dobmeos with hgh my **energy** level has gone up ! stukm Introducing doctor – formulated hgh human **growth** **hormone** - also called hgh is referred to in medical science as the master **hormone**. it is very plentiful when we are young , but near the age of twenty - one our bodies begin to produce less of it . by the time we are forty nearly everyone is deficient in hgh , and at eighty our production has normally diminished at least 90 - 95 % . advantages of hgh : - increased muscle strength - loss in body fat - increased **bone** density - lower blood pressure - quickens wound **healing** - reduces cellulite - increased **sexual** potency ...	Subject: re : entex transistion thanks so much for the memo . i would like to reiterate my support on two key issues : 1) . thu - best of luck on this new assignment . howard has worked hard and done a great job ! please don ' t be shy on asking questions . entex is critical to the texas business , and it is critical to our team that we are timely and accurate . 2) . rita : thanks for setting up the **account** team . communication is critical to our success , and i encourage you all to keep each other informed at all times . the p & l impact to our business can be significant . additionally , this is high profile , so we want to assure top **quality** . thanks to all of you for all of your efforts . let me know if there is anything i can do to help provide any additional support . rita wynne ...

Figure 1. Sample spam and ham emails. Large bold words indicate top ranked spam words and smaller words with low ranking, whereas normal black text indicate non spam words.

In most cases of generating feature vectors, scientists usually concentrate on static models that require complete refactoring when information changes or when the user provides feedback. In order to cope with the demand of changes, the proposed model can automatically recalculate the salient features and appropriately adapt the vector model to accommodate this (see Figure 2).

The method can safely modify/update the vector model every 100 emails in order to achieve best performance. The choice of 100 emails for an update was chosen based on several trials of the given dataset. The process though can be automated to update when the performance of the system declines or when new keywords with high frequency are detected in the incoming emails. The rank list is modified depending on the contents of the new coming emails. This is visualised in Figure 3 where it is observable that as more email batches (of 100 emails) are presented, the tokens in the list get updated. New "important" tokens are quickly placed at the top of the rank, but the ranking changes based on other new entries.

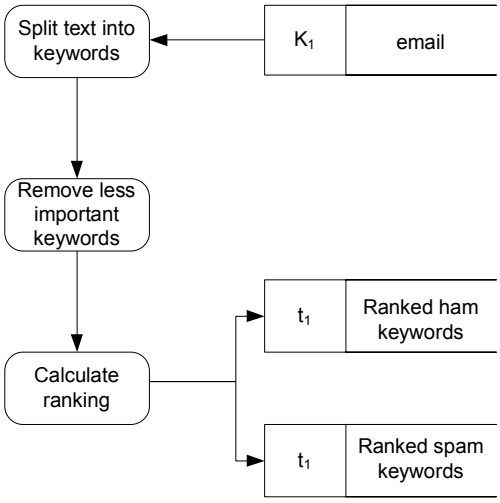

Figure 2. Keyword selection data flow diagram for both spam and ham emails. The result of the process is two ordered lists of ham and spam emails that form the basis for the next stage of encoding feature vectors.

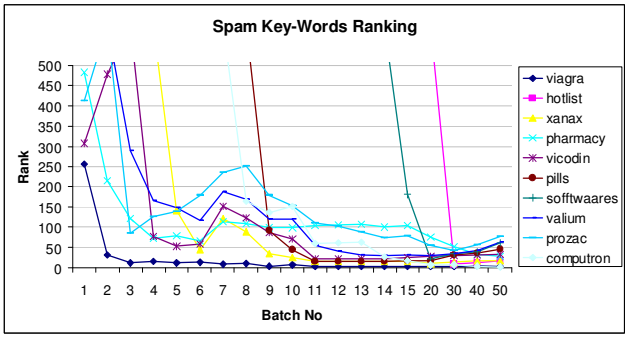

Figure 3. Random spam keyword ranking as it evolved through the training process for Enron1 dataset. Each batch contains 100 emails. The graph shows that each new keyword entry has an impact on the ranking list, and it then fluctuates to accommodate new-coming keywords. Salient keywords eventually end up at the bottom of the graph (i.e. top rated).

3.2 Encoding Feature Vectors

With the selection of keywords complete, they can now be used to create the vectors representing each email. This scenario considers a single email being converted into a vector. The email will be split into individual word tokens (String Tokenizer) as before. Previous experiments suggested that a proportion of the ranked ham and spam keywords will then be considered, e.g. the top 500 of spam words, in order to build good representative vectors [20]. Each keyword will be compared to the tokens extracted from the input email. If the token exists in the email a '1' is added to the vector, if it doesn't exist in the email a '0' is added. However this method suffered from slow training due to the high-dimensional vectors and also suffered from lack of representative words as sometimes 500 was not enough to cover all past and new coming spam keywords, therefore there were issues with sparse data (i.e. almost empty vectors).

A potential solution was devised that instead of having each element representing just one keyword, but to have each element represent a number of keywords, as recommended in [12]. So instead of being a binary value, that element would be a numeric value between 0 and 1 specifying the proportion of keywords in that range that occur in the email. To further this design it was decided to put more importance on the higher ranking spam words. Occurrences of higher ranked spam words are strong examples of a spam email. Whereas spam keywords ranked much lower indicate less strongly of a spam email, and potentially very low ranked spam words maybe even occur in ham emails. To achieve this, the number of keywords representing each element would increase the lower down the spam rankings. This is shown in Figure 4:

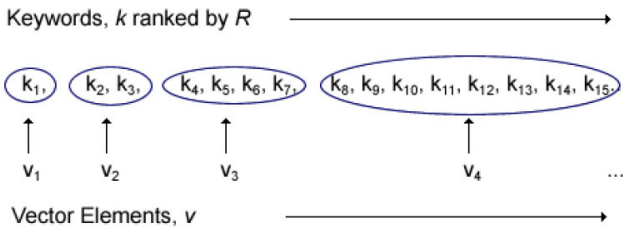

Figure 4. Keywords representing each element in a vector

With this new capacity, more than just the first 500 spam elements could be considered, in order to fit a vector of only 25 dimensions. In this design it was decided to include almost all the spam keywords into the vector. To achieve this, the number of keywords in each vector would have to adapt to the size of the spam keyword attribute set. The number of keywords per vector element would be calculated by taking the element number and raising it to a power. After several experiments with different choices for the size of the vector, it was found that 25 dimensions are more than enough to represent spam emails.

The spam vector now shows fractions of keyword occurrences across the whole vector. This particular example is a strong indication of a spam email. The ham example vector shows data towards the end of the vector demonstrating a small fraction of low spam keyword occurrences. The majority of high spam features will look empty and this will strongly reflect a ham email. This design also provides more information to the classifier than the previous design.

Furthermore, it had been discovered that many of the ham vectors were empty sets. These sets demonstrated that the ham emails had no spam keyword features and therefore were very strongly ham emails. To avoid a completely empty dataset a simple binary rule based feature was added to the vector. This resulted in all vectors not being completely empty and having some information to demonstrate features of a category. This feature was designed so that if the SOM found a 1in the 26th vector feature then it would cluster these vectors together as being ham emails. However, unlike the

501st element reported in [20], this 26th vector element does not have such a strong effect on classification. Although the results of 25 vector elements were strong and influential on training, the addition of the 26th vector element purely prevents providing the classifier with an empty set.

Along with these keyword occurrences, numerical statistical values will also be calculated and concatenated onto the end of the vector. The output is therefore a single *n*-dimensional vector representing one email. This process is then conducted over a dataset of emails to create a full set of vectors. This design of vector creation shows the basic system flow for creating a vector (see Figure 5). However it is intended that this design will be flexible as the number and type of features selected will vary.

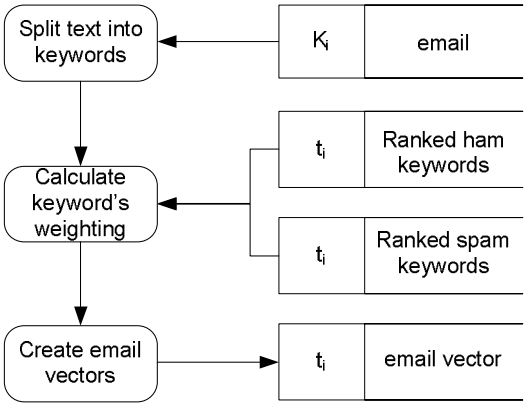

Figure 5. Vector creating flow diagram

4 Experimentation: Spam Detection

In order to evaluate spam filters a dataset with a large volume of spam and ham messages is required. Gathering public benchmark datasets of a large size has proven difficult [8]. This is mainly due to privacy issues of the senders and receivers of ham emails with a particular dataset. Some datasets have tried to bypass the privacy issue by considering ham messages collected from freely accessible sources such as mailing lists. The *Ling-Spam* dataset consists of spam received at the time and a collection of ham messages from an archived list of linguist mails. The *SpamAssassin* corpus uses ham messages publicly donated by the public or collected from public mailing lists. Other datasets like *SpamBase* and *PU* only provide the feature vectors rather than the content itself and therefore are considered inappropriate for the proposed method.

4.1 Setup

One of the most widely used datasets in spam filtering research is the *Enron* dataset From a set of 150 mailboxes with messages various benchmark datasets have been constructed. A subset as constructed by Androutsopoulos et al. [6] is used, containing mailboxes of 6 users within the dataset. To reflect the different scenarios of a personalised filter, each dataset is interlaced with varying amounts of spam (from a variety of sources), so that some had a ham-spam ratio of 1:3 and others 3:1.

To implement the process of incremental retraining the approach suggested by Androutsopoulos et al. [6] is adapted, where the messages of each dataset are split into batches $b_1,\ldots,b_l$ of k adjacent messages. Then for batch i=1 to l-1 the filter is trained on batch b_i and tested on batch b_i+1. The number of emails per batch k=100.

The SOM is retrained every 100 emails. For testing purposes the SOM is tested on 100 emails at a time. In practice the SOM will be presented with one incoming email at a time to classify. The SOM is trained over 100 cycles with an initial neighbourhood effect of 6 nodes, reducing to 0.1 through training.

The performance of a spam filter is measured on its ability to correctly identify spam and ham while minimising misclassification. $N_{h\to h}$ and $n_{s\to s}$ represent the number of correctly classified ham and spam messages. $N_{h\to s}$ represents the number of ham misclassified as spam (false positive) and $n_{s\to h}$ represents the number of spam misclassified as ham (false negative). Spam precision and recall is then calculated.

These measurements are useful for showing the basic performance of a spam filter. However they do not take into account the fact that misclassifying a Ham message as Spam is an order of magnitude worse than misclassifying a Spam message to Ham. A user can cope with a number of false negatives, however a false positive could result in the loss of a potential important legitimate email which is unacceptable to the user. Therefore, when considering the statistical success of a spam filter, the consequence weight associated with false positive emails (i.e. non spam emails that were incorrectly classified as spam emails) should be taken into account. Androutsopoulos et al. [6] introduced the idea of a weighted accuracy measurement (*WAcc*) in order to address this issue:

$$WAcc_{\lambda} = \frac{\lambda \cdot n_{h\to h} + n_{s\to s}}{\lambda \cdot N_h + N_s} \quad (6)$$

N_h and Ns represent the total number of ham and spam messages respectively. In this measurement each legitimate ham message n_h is treated as λ messages. For every false positive occurring, this is seen as λ errors instead of just 1. The higher the value of λ the more cost there is of each misclassification. When λ =99, misclassifying a ham message is as bad as letting 99 spam messages through the filter. The value of λ can be adjusted depending on the scenario and consequences involved.

As well as a comparative analysis, this paper will also give a visual analysis of the model. The visual capabilities of the SOM were one of the reasons for its inclusion here.

4.2 Results

The design behind these initial experiments is to replicate the situation faced by an email user to progressively train a classifier to filter spam based on their individual email set. This basically involves small amounts of training data initially growing in size as the incremental retraining process continues and more incoming mail is presented.

The experiment will run over 30 batches of each of the six datasets. This results in 18,000 emails being trained and

tested upon in this experiment. Each of the six datasets has been pre-processed to include the mail belonging to a single user interlaced with spam messages. Therefore the attribute sets will be cleaned to reflect a new learning experiment for each user. There is a variance in the ratio of ham to spam messages through the six datasets, and it was interesting to see how this fluctuation changes the results.

4.2.1 SOM vs BDT and SVM

The ham results are more or less perfect for all three methods used. The difference comes when measuring the recall for spam (Table 1). SOM seams to deal with a wider range of emails, whereas BDT and SVM do well on Enron 2, 5 and 6, but not as well for the rest.

As a further comparison in line with the other phase evaluations conducted in this paper, the weight accuracy results of the classifiers will be compared. The dataset Enron 4 typifies well the pattern of weighted accuracy results across all datasets. The graph in Figure 6 shows the *WAcc* results for all three classifiers over the Enron 4 dataset.

	SOM	BDT	SVM
Enron 1	87.31	87.07	87.20
Enron 2	91.74	95.20	97.21
Enron 3	94.82	94.44	94.50
Enron 4	85.87	85.55	85.73
Enron 5	97.87	97.87	97.87
Enron 6	94.43	94.39	94.43

Table 1. Contrasting the spam recall (%) results for SOM, BDT and SVM. The SOM classifier seems to be consistently better than the other two.

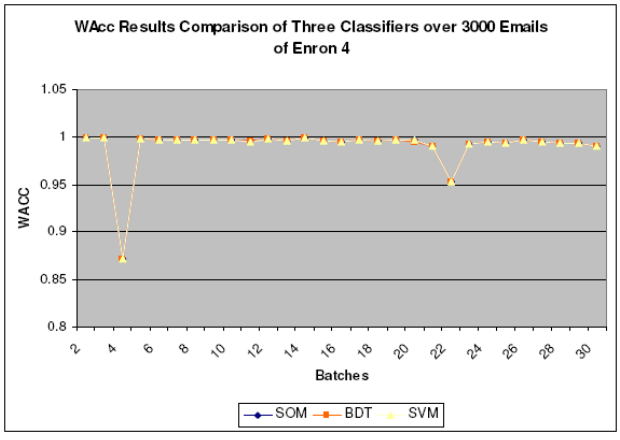

Figure 6. Weight accuracy results for the SOM, BDT and SVM classifiers on Enron 4 dataset. All results are more or less identical for all three methods.

The performance of all three classifiers is so consistent the lines are barely distinguishable. Across datasets 1 and 3-6 only 6 ham emails have been misclassified, this is the over the classification of 15,000 emails. This performance is highly impressive and highly desirable for a spam filtering system. The strength of all three models again outlines the strength of the vector creation design.

4.2.2 SOM vs MNB

Firstly considering the recall of Ham, apart from Enron 2, the SOM model outperforms the results of the MNB consistently. The paper considered datasets like Enron 1 and Enron 5 as tough and this can be seen by the MNB drop in ham recall. However the SOM maintains strong performance over the MNB. This set of results is an excellent demonstration of the capabilities of the SOM in matching and improving on other spam filtering techniques shown in research.

The spam recall for the MNB however is consistently better than the SOM. This is related to the trade off in performance seen previously in this report between ham recall and spam recall. When considering the *WAcc* results of both, the performance of the SOM almost consistently outperforms the MNB. The graph in Figure 7 shows the average *WAcc* result for each of the six Enron datasets. Apart from Enron 2 the SOM is consistently above the MNB results.

	HAM		SPAM	
	SOM	MNB	SOM	MNB
Enron 1	99.95	95.25	87.31	96
Enron 2	96.46	97.83	91.74	96.68
Enron 3	100	98.88	94.82	96.64
Enron 4	99.45	99.05	85.87	97.79
Enron 5	100	95.64	97.87	99.69
Enron 6	99.86	96.88	94.43	98.1

Table 2. Contrasting the spam recall and precision (%) results for SOM and MNB. SOM seems to be better at identifying ham emails, whereas MNB does better on spam emails.

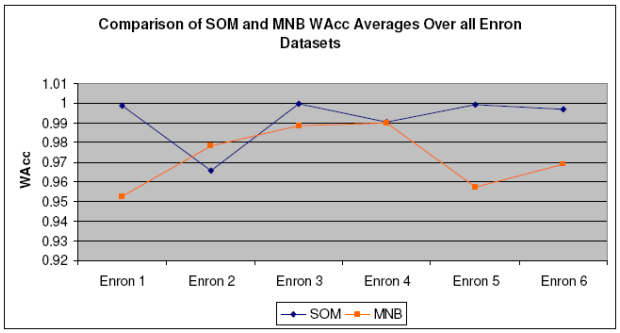

Figure 7. WAcc results over all six datasets for SOM and MNB. The MNB seems to be costing more than the SOM, which means that it misclassifies more ham emails as spam, i.e. some "good" emails may get lost!

Both classifiers use a different feature representation, however the feature vector have been modified to suit that of the SOM, and the results are across the same dataset of emails. Therefore these results show the overall prototype model presented in this report can exceed the results of models conducted over the same data.

4.2.3 Visual Evaluation of SOM

After evaluating the SOM against related work in the research area, the results demonstrated the SOMs suitability for spam filtering meeting the objectives of this project. One of the other objectives included for this paper was to include a visual representation of the classification of ham and spam. This is part of the reason for the selection of the SOM because its abilities of visual representation were identified at the start of the project. This visual representation will give a further evaluation on the performance of the SOM.

Although the results have been very positive there remains a current problem with the SOM classification around the 6th batch of the Enron 2 dataset. The performance on the dataset is accurate to begin with but there is a drop in accuracy around batch 6. Using a visual analysis it may be easier to see if the SOM struggles to recognise patterns in this batch. The SOM is setup as a 10x10 2-dimensional grid, containing 100 nodes. Each node is made of up *n*-dimensions of weights representing the number of features in the training vectors. To represent the SOM visually after a batch of training, the weights of each node are summed to form 100 summed values. These values topological position will be maintained in the 10x10 grid formation with each node represented by a circle. The size of the circle represents the value of the summation of weights. Therefore a node represented by a large circle, will have a large summation of the weights. In terms of the features vectors in this described model, bigger weighted summations, the larger circles, will represent patterns that look like spam emails. In contrast the smaller circles will represent the emptier features of ham emails.

This can be seen in Figure 8 where identifiable regions of similar size circles (weight nodes) can be identified. The figures show 10x10 grids of 100 nodes each. Large circles demonstrate how spam like features cluster together and the smaller circles show clusters of ham patterns. In the middle of these regions the medium circles show the border between the clustered regions.

The six node maps in Figure 8 show the results of training on batch 2 through to batch 7 on the Enron 2 dataset. These maps show some interesting results. The first map in the top left shows the map after the second batch of training, 200 emails. Already, even with very little training data, there are clear defined patterns visible. Weights which are closest to spam emails locate in the top right corner of the map while weights close to ham emails are in the bottom left corner. The ham region of this map is a lot larger than the spam region, and this is explained by the larger number of ham training inputs in spam for this Enron 2 dataset. The results of testing on this batch show strong results.

Batches 3 and 4 in Figure 8 show less definition in the regions of spam and ham with the spam regions pushed to either edge of the map during training. On the other hand there are still clusters of similar inputs and the results for both batches are good.

However batch 5 and 6 show a different pattern. The regions of ham and spam are less clear in these batches. Multiple smaller clusters of similar weights can be seen, but there are no obvious defined regions. It is over these batches (5 & 6) where the accuracy results fall dramatically. Over batch 5 the ham accuracy drops badly and over batch 6 spam accuracy drops badly. The smaller clusters in these two batches show how the SOM has struggled to recognise consistent patterns in these training sets and this reflects in the poor results. Notably the SOM recovers well from these bad batches and by batch 7 the clusters of ham and spam are much more defined. The results for batch 7 are also very strong. This demonstrates the SOMs ability to recover from poor training data to minimise prolonged poor performance. This is a desirable quality for a classifier, especially in this spam filtering research domain.

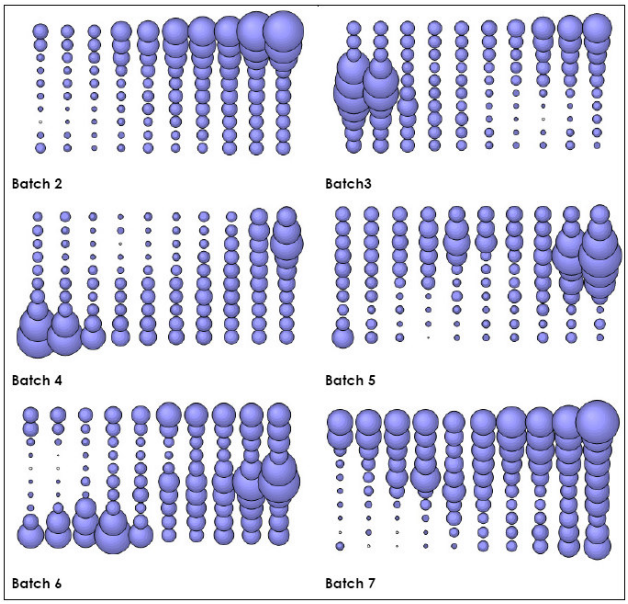

Figure 8. Visual representation of SOM training on Enron 2. Large circles represent highly ranked spam emails and small circles represent emails with no or few spam keywords.

The main aspects to conclude from this visual evaluation are that the SOM can create defined clusters with only a few input examples, and also recover well after poor vectors are presented to it. This evaluation again shows the strength of having good feature selection. The SOM can find clear distinctive patterns in a small number of examples and the maps shown can clearly demonstrate successful clustering. This visual evaluation also shows that even though the SOMs learning neighbourhood is relatively small after the first training batch, each subsequent batch of training has a relatively large effect on the map. For example in Figure 8 the regions of small circles (ham nodes) seem to move about the map between batches. This shows that the strength of the input vectors can cause dramatic changes to the SOM map and this explains the SOMs ability to recover well between batches.

5 Conclusions

This paper has discussed and evaluated four classifiers for

the purposes of categorising emails into classes of spam and ham. All MNB Boolean, SVM and BDT and SOM methods are incrementally trained and tested on 6 subsets of the Enron dataset. The methods are evaluated using a weighted accuracy measurement. A design model for comparing spam classifiers was detailed, showing the use of *weirdness* and *tf*idf* measurements to rank a list of most important spam keywords.

MNB classifiers were identified as the most popular and common models in spam filtering, whereas BDT and SVM were shown to be applied in this field with good success. The results of the SOM proved consistent over each dataset maintaining an impressive spam recall, and only a small percentage of ham emails are misclassified by the SOM. Each ham missed is treated as the equivalent of missing 99 spam emails.

The six phases of design testing resulted in a prototype model that included 26 vector features representing this email. This model was then evaluated against the Enron dataset showing consistent accuracy results. The SOM did well to match and even exceed the accuracy of other better known classifiers. The SOM was also evaluated visually with the node maps showing the learning process of the SOM. The SOM was seen to adapt quickly to changes in the vector with no prolonged areas of poor performance. Poor performance was noticed in one of the datasets and this was explained by showing that the results stabled out after progressive incremental training.

Acknowledgment

The authors would like to thank the anonymous reviewers for their insightful comments and for selecting our paper to be published in this special issue. A shorter version of this paper appeared in the Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems (CISIS 2008) [20]. Special thanks to the Britain's Royal Academy of Engineering and the Department of Computing, University of Surrey, for financially supporting this work.

References

[1] P. Manomaisupat, B. Vrusias and K. Ahmad. "Categorization of Large Text Collections: Feature Selection for Training Neural Networks". In: *Intelligent Data Engineering and Automated Learning - IDEAL 2006*, LNCS, vol. 4224, pp. 1003-1013, 2006.

[2] T. Kohonen. *Self-organizing maps*, 2nd ed., Springer-Verlag, New York, 1997.

[3] V. Metsis, I. Androutsopoulos and G. Paliouras. "Spam Filtering with Naïve Bayes – Which Naïve Bayes?", In: *CEAS, 3rd Conf. on Email and AntiSpam*, California, USA, 2006.

[4] L. Zhang, J. Zhu and T. Yao. "An Evaluation of Statistical Spam Filtering Techniques". In: *ACM Trans. on Asian Language Information Processing*, III (4), pp. 243-269, 2004.

[5] M. Sahami, S. Dumais, D. Heckerman and E. Horvitz. "A Bayesian approach to filtering junk e-mail", In: *Learning for Text Categorization – Papers from the AAAI Workshop*, pp. 55-62, 1998.

[6] I. Androutsopoulos, G. Paliouras, V. Karkaletsi, G. Sakkis, C.D. Spyropoulos and P. Stamatopoulos. "Learning to Filter Spam E-Mail: A Comparison of a Naïve Bayesian and a Memory-Based Approach", In: *Proceedings. of the Workshop Machine Learning and Textual Information Access*, pp. 1-13, 2000.

[7] S. Youn and D. McLeod. "Efficient Spam Email Filtering using Adaptive Ontology", In: *4th International Conf. on Information Technology*, pp. 249-254, 2007.

[8] R. Hunt and J. Carpinter. "Current and New Developments in Spam Filtering", In: *14th IEEE Int. Conference on Networks*, vol. 2, pp. 1-6, 2006.

[9] F. Peng, D. Schuurmans and S. Wang. "Augmenting Naive Bayes Classifiers with Statistical Language Models", *Information Retrieval*, VII (3-4), pp. 317-345, 2004.

[10] G. Salton and C. Buckley. "Term-weighting approaches in automatic text retrieval", *Information Processing & Management*, XXIV (5), pp. 513-523, 1988.

[11] B. Vrusias. "Combining Unsupervised Classifiers: A Multimodal Case Study", *PhD thesis*, University of Surrey, 2004.

[12] H. Drucker, D. Wu and V.N. Vapnik. "Support Vector Machines for Spam Categorization", In: *IEEE Transactions on Neural Networks*, X (5), pp. 1048-1054, 1999.

[13] T. Joachims. "Text Categorization with Support Vector Machines: Learning with Many Relevant Features", In: *Proceedings of 10th European Conference on Machine Learning*, pp. 137-142, 1998.

[14] A. Kolcz and J. Alspector. "SVM-Based Filtering of e-Mail Spam with Content Specific Misclassification Costs," In: *Proceedings of the Workshop on Text Mining*, San Jose, 2001.

[15] R.E. Schapire and Y. Singer. "A Boosting-Based System for Text Categorization," *Machine Learning*, pp. 135-168, 2000.

[16] R.E. Schapire. "The Strength of Weak Learnability," *Machine Learning*, V (2), pp. 197-227, 1990.

[17] S.M. Weiss, C. Apte, J. Damerau, D.E. Johnson, F.J. Oles, T. Goetz and T. Hampp. "Maximizing Text-Mining Performance," *IEEE Intelligent Systems*, pp. 63-69, 1999.

[18] S. Ali and Y. Xiang. "Spam Classification Using Adaptive Boosting Algorithm," In: *6th IEEE/ACIS International Conference on Computer and Information Sciences*, pp.972-976, 2007.

[19] X. Carreras and L. Marquez. "Boosting Trees for Anti-Spam Email Filtering," In: *Proceedings of 4th Int. Conference on Recent Advances in Natural Language Processing*, Bulgaria, pp.58-64, 2001.

[20] B. Vrusias and I. Golledge "Adaptable Text Filters and Unsupervised Neural Classifiers for Spam Detection," In: *Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems*, pp. 195-202, 2008.

[21] H. Najadat and I. Hmeidi. "Web Spam Detection Using Machine Learning in Specific Domain Features",

Journal of Information Assurance and Security, III (3), pp. 220-229, 2008.

[22] E.-S. M. El-Alfy. "Discovering Classification Rules for Email Spam Filtering with an Ant Colony Optimization Algorithm", In: *IEEE Congress on Evolutionary Computation*, pp. 1778 - 1783, 2009.

[23] M. Kassoff, C. Petrie, L.-M. Zen and M. Genesereth. "Semantic Email Addressing: The Semantic Web Killer App?", *IEEE Internet Computing*, III (1), pp. 48 - 55, 2009.

Author Biographies

Bogdan Vrusias Born in Constanta, Romania in 1975, and then returned with his family in Veroia, Greece in 1983, before moving to England in 1994 where he currently lives. He graduated in 1998 with a BSc (Honours) in Computing and IT from the University of Surrey, UK, where he also accomplished his PhD in 2004 in the area of multimedia and neural computing. He worked for the University of Surrey as a technology transfer associate in the area of data mining and neural networks from 1998 to 2001, then as a research officer for the EPSRC SoCIS project till 2004, followed by his current position as a lecturer in the area of intelligent and distributed systems. He belongs to the Biologically Inspired Modelling and Applications research group and his main areas of research include neural computing, multimedia information retrieval, distributed systems, business intelligence, image and video analysis, data mining, knowledge representation and management.

Ian Golledge Born in Ipswich, UK in 1985, is currently a consultant in the field of information security with a business and technology consultancy specialist based in London at the start of his professional career outside of academia. Prior to becoming a consultant he received his Bachelor of Science degree in computing and information technology from the University of Surrey in 2007 and his Master of Science degree in security technologies and applications from the University of Surrey in 2008. His area of research interests include artificial intelligence, in particular unsupervised classifiers, cryptography and digital watermarking with focus now on the practice of information security and security architecture.