Filtering Spam E-Mail with Generalized Additive Neural Networks

Tiny du Toit
School of Computer, Statistical and Mathematical Sciences
North-West University, Potchefstroom Campus
Private Bag X6001, Potchefstroom, 2520
South Africa
E-mail: Tiny.DuToit@nwu.ac.za

Hennie Kruger
School of Computer, Statistical and Mathematical Sciences
North-West University, Potchefstroom Campus
Private Bag X6001, Potchefstroom, 2520
South Africa
E-mail: Hennie.Kruger@nwu.ac.za

***Abstract*—Some of the major security risks associated with spam e-mail are the spreading of computer viruses and the facilitation of phishing exercises. Spam is therefore regarded as one of the prominent security threats in modern organizations. Security controls, such as spam filtering techniques, have become increasingly important to protect information and information assets. In this paper the performance of a Generalized Additive Neural Network on a publicly available e-mail corpus is investigated in the context of statistical spam filtering. The neural network is compared to a Naive Bayesian classifier and a Memory-based technique. Generalized Additive Neural Networks have a number of advantages compared to neural networks in general. An automated construction algorithm performs feature and model selection simultaneously and produces results which can be interpreted by a graphical method. This algorithm is powerful, effective and performs highly accurate compared to other non-linear model selection methods. The paper also considers the impact of different feature set sizes using cost-sensitive measures. These criteria are sensitive to the cost difference between two common types of errors made by filtering systems. Experiments show better performance compared to the Naive Bayes and Memory-based classifiers where legitimate e-mails are assigned the same cost as spams. This result suggests Generalized Additive Neural Networks may be utilized to flag spam e-mails in order to prioritize the reading of messages.**

***Index Terms*—Generalized Additive Neural Network, Memory-based classifier, Naive Bayesian classifier, Neural Network, Security risk, Spam, Spam filtering.**

I. Introduction

The Internet has promoted new channels of communication which allow an e-mail to be sent to people thousands of kilometers away [1]. Sending messages by e-mail has a number of advantages including high reliability, relatively low transmission costs, generally fast delivery and the ability to be automated [2]. These strengths enable almost free mass e-mailing which can reach out to hundreds of thousands of users within seconds. Unfortunately, this freedom of communication can be exploited. Over the last number of years a situation has been reached where users' mailboxes have been flooded with unwanted messages. This deluge of spam has escalated to the point where the viability of communication via e-mail is threatened.

Although spam messages can be easily recognised, it is difficult to develop an accurate and useful definition of spam. The Concise Oxford English Dictionary (eleventh edition) defines spam as "irrelevant or inappropriate messages sent on the Internet to a large number of newsgroups or users". Spam is primarily used for advertising. Commodities proposed for commercial purposes range from computer software and medical products to investments. Spam is also employed to express religious or political opinions, mislead the target audience with promises of fortune and distribute useless chain letters.

Spam causes a number of problems to the Internet community. It takes time to sort unwanted messages which poses a risk that legitimate mail can be deleted. Delivery of normal mail can be delayed by large amounts of spam-traffic between servers. Users with dial-up Internet access have to waste bandwidth downloading junk mail. Furthermore, some spam are pornographic in nature and should not be revealed to children.

Spam is no longer just considered as an invasive annoyance or a problem of convenience but it is regarded and accepted as an issue which poses a considerable security risk to enterprises. This view is due to the fact that spam is used, amongst other things, for spreading computer viruses and as a deceptive method of obtaining sensitive information. It has already been reported in 2004 that about ninety percent of companies agreed that spam makes their companies more vulnerable to security threats [3]. More recent surveys and reports support this point of view [4]; [5]; [6]. It has therefore become imperative to ensure that proper policies and controls are in place to mitigate the security risks associated with spam. One important control is the detection and management of spam messages.

A number of techniques have been applied to filter spam messages [7]; [8]; [9]. In this paper a Generalized Additive Neural Network (GANN) is applied to a publicly available corpus to detect spam messages. The Ling-Spam collection was constructed by [10] and made available as a benchmark corpus. Results obtained by the GANN are then compared to the performances of a Naive Bayesian classifier and a Memory-based learning technique applied by [10]. This comparison will provide insight into the feasibility of using a GANN to detect unwanted messages.

The rest of the paper is organized as follows. GANNs

are discussed in Section II. Examples of typical supervised prediction models are presented and difficulties encountered with neural networks in general are considered. In Section III the publicly available Ling-Spam corpus is discussed. As preprocessing steps, vector representations of the messages are constructed and feature selection is performed. Next, Naive Bayesian classification and Memory-based classification are considered in Sections IV and V. In addition three practical scenarios where these methods can be applied are discussed. Cost-sensitive measures used to evaluate the learning techniques are presented in Section VI. These metrics takes into account the differences in cost of classifying a legitimate message as spam and vice versa. Experimental results are considered in Section VII. Finally, some conclusions on the feasibility of GANNs to detect spam are presented in Section VIII.

II. Generalized Additive Neural Networks

To arrive at a spam filter, a decision function f must be obtained that classifies a given e-mail message $\boldsymbol{m}$ as spam (S) or legitimate mail (L) [2]. If the set of all e-mail messages is denoted by $\mathbb{M}$, a search for the function $f : \mathbb{M} \rightarrow \{S, L\}$ is performed by supervised learning. With this technique learning methods are trained on a set of pre-classified messages $\{(\boldsymbol{m}_1, c_1), (\boldsymbol{m}_2, c_2), \ldots, (\boldsymbol{m}_n, c_n)\}, \boldsymbol{m}_i \in \mathbb{M}, c_i \in \{S, L\}$. Examples of such learning methods are Generalized Linear Models, Multilayer Perceptrons, Generalized Additive Models and Generalized Additive Neural Networks [11].

Generalized Linear Models [12],

$$g_0^{-1}(E(y)) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k,$$

are often used for predictive modeling. The range of predicted values are restricted by the link function, g_0^{-1}. For spam detection, the logit link

$$g_0^{-1}(E(y)) = ln(\frac{E(y)}{1 - E(y)})$$

is appropriate as the expected target (probabilities) is bounded between zero and one. The parameters are usually estimated by maximum likelihood.

Multilayer Perceptrons [13]; [14]; [15] are the most widely used type of neural network for supervised prediction. A Multilayer Perceptron (MLP) with a single hidden layer with h hidden neurons has the form

$$g_0^{-1}(E(y)) = w_0 + w_1 tanh(w_{01} + \sum_{j=1}^{k} w_{j1} x_j) + \ldots$$

$$+ w_h tanh(w_{0h} + \sum_{j=1}^{k} w_{jh} x_j),$$

where the link function is the inverse of the output activation function. Although other sigmoidal functions could be used, the activation function in this case is the hyperbolic tangent. The unknown parameters are estimated by numerically optimizing some appropriate measure of fit to the training data such as the negative log likelihood.

A Generalized Additive Model (GAM) is defined as

$$g_0^{-1}(E(y)) = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_k(x_k),$$

where the expected target on the link scale is expressed as the sum of unspecified univariate functions [16]; [17]; [18]. Each univariate function can be regarded as the effect of the corresponding input while holding the other inputs constant. When a GAM is implemented as a neural network it is called a *Generalized Additive Neural Network*.

The main architecture of a GANN is comprised of a separate MLP with a single hidden layer of h units for each input variable:

$$f_j(x_j) = w_{1j} tanh(w_{01j} + w_{11j} x_j) + \ldots$$

$$+ w_{hj} tanh(w_{0hj} + w_{1hj} x_j).$$

The individual bias terms of the outputs are incorporated into the overall bias β_0. Each individual univariate function contains $3h$ parameters, where h, the number of hidden neurons, could be different across inputs. This architecture can be extended to include an additional parameter for a direct connection (skip layer):

$$f_j(x_j) = w_{0j} x_j + w_{1j} tanh(w_{01j} + w_{11j} x_j) + \ldots$$

$$+ w_{hj} tanh(w_{0hj} + w_{1hj} x_j).$$

A backfitting algorithm is used by [16] and [17] to estimate the individual univariate functions f_j. Backfitting is not required for GANNs. Any method that is suitable for fitting more general MLPs can be utilized to simultaneously estimate the parameters of GANN models. The usual optimization and model complexity issues also apply to GANN models.

In general, to construct a neural network and interpreting results obtained are not trivial tasks. There are a number of decisions that must be made to determine the architecture. These decisions include the number of hidden nodes and the different activation functions. Trial-and-error or experiments is currently the most common way to determine the number of hidden nodes [15]. Additionally, various rules of thumb have been suggested. Examples of these rules are that the number of cases determine the number of hidden nodes and for each weight there should be at least ten records. Some researchers limit the number of hidden nodes with empirical rules. Alas, it has been found that none of these heuristic methods perform well for all problems.

Choosing the appropriate number of inputs to the neural network is also not obvious. At best, a relatively small number of essential nodes are required which can identify the unique features found in the data. The learning or prediction capability of the network can be negatively influenced by too little or too many input nodes. When the number of inputs is too little, the neural network may not achieve the desired level of accuracy. Overtraining may occur when too many input nodes are utilized.

To make matters worse, neural networks in general are regarded as black box methods. There is no explicit form

to explain and analyze the relationship between inputs and the target which causes difficulty in interpreting results from the networks. Fortunately, these concerns are addressed by the automated construction algorithm for GANNs.

At present two algorithms exist to estimate GANN models. An interactive construction algorithm was suggested by [19] which utilizes visual diagnostics to specify the complexity of each univariate function. To perform model selection, plots of the fitted univariate functions, $\hat{f}_j(x_j)$ overlaid on the partial residuals

$$pr_j = g_0^{-1}(y) - \hat{\beta}_0 - \sum_{l \neq j} \hat{f}_l(x_j)$$

$$= (g_0^{-1}(y) - g_0^{-1}(\hat{y})) + \hat{f}_j(x_j),$$

versus the corresponding jth input are examined [20]; [21]; [22]. For a large number of inputs this evaluation of the partial residual plots can become a daunting and time consuming task. It is also known that human judgement is subjective which may result in models that are suboptimal. Therefore [11] developed an automated construction algorithm based on the search for GANN models using objective model selection criteria or cross-validation. In this algorithm, partial residuals plots are used as a tool to provide insight into the models created and not for model building. With sufficient time to evaluate candidate models, this best-first search method is optimal and complete. It was shown that the algorithm is effective, powerful and is comparable to other non-linear model selection techniques found in the literature [11]. The implementation of the automated construction algorithm, called *AutoGANN*, was applied to the publicly available Ling-Spam corpus to detect spam.

Prior to discussing the Naive Bayesian classifier and the Memory-based classifier to which the results of the GANN will be compared, specific vector notation to represent e-mail messages will be given. This notation together with the Ling-Spam collection and the preprocessing steps performed on the corpus are considered next.

III. Ling-Spam Corpus Collection and Preprocessing

The Ling-Spam collection used in this paper was compiled by [10] and is a combination of messages obtained from the Linguist list and spam e-mail messages. The former is a moderated mailing list about the science and occupation of linguistics. The corpus contains 2893 messages which is partitioned into 2412 Linguist messages and 481 spam messages. Cost-sensitive evaluation metrics were introduced by [10] to get an objective picture of the performance of the Naive Bayesian algorithm. These metrics are also utilized in this paper since the cost of misclassification differs for the two classes (spam and legitimate).

As in [23] a vector representation $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$ is constructed for every message in Ling-Spam where $x_1, x_2, \ldots, x_n$ denote the values of attributes $X_1, X_2, \ldots, X_n$. These values are binary with $X_i = 1$ if some characteristic corresponding to X_i is present in the message and $X_i = 0$ otherwise. For the experiments, each attribute indicates whether a particular word (e.g. computer) can be found in the message.

Feature selection is performed by ranking the candidate attributes by their mutual information (MI) values and choosing the attributes with the m highest MI scores. The MI value of each candidate attribute was computed as follows:

$$MI(X;C) = \sum_{\substack{x \in \{0,1\}, \\ c \in \{S,L\}}} P(X = x, C = c) \cdot$$

$$log \frac{P(X = x, C = c)}{P(X = x) \cdot P(C = c)},$$

where C denotes the category which can be spam (S) or legitimate messages (L) [23].

The probabilities are estimated as frequency ratios from the training corpus. Each word in the Ling-Spam corpus was substituted by its base form with a lemmatizer to prevent handling forms of the same word as different attributes.

In the next section the Naive Bayesian filter utilized by [10] to detect spam is discussed. Moreover, three practical scenarios in which the filter are employed are considered.

IV. Naive Bayesian Classification

According to Bayes' theorem and the theorem of total probability, the probability that a document d with vector $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$ is a member of c is:

$$P(C = c|\vec{X} = \vec{x}) =$$

$$\frac{P(C = c) \cdot P(\vec{X} = \vec{x}|C = c)}{\sum_{k \in \{S,L\}} P(C = k) \cdot P(\vec{X} = \vec{x}|C = k)},$$

where S denotes a spam message and L a legitimate message.

The probabilities $P(\vec{X}|C)$ are impossible to estimate in practice without simplifying assumptions since the possible values of $\vec{X}$ are too many and data sparseness problems exist. As a result the Naive Bayesian classifier assumes that $X_1, \ldots, X_n$ are conditionally independent given the category C, which yields

$$P(C = c|\vec{X} = \vec{x}) =$$

$$\frac{P(C = c) \cdot \prod_{i=1}^{n} P(X_i = x_i|C = c)}{\sum_{k \in \{S,L\}} P(C = k) \cdot \prod_{i=1}^{n} P(X_i = x_i|C = k)}.$$

By using the frequencies of the training corpus, $P(X_i|C)$ and $P(C)$ are easy to estimate. $P(X_i|C)$ is the percentage of training corpus messages present ($X_i = 1$) or absent ($X_i = 0$) given a certain class (spam or legitimate). $P(C)$ is the percentage of spam or legitimate training corpus messages.

A message is filtered as spam if the following condition is satisfied:

$$\frac{P(C = S|\vec{X} = \vec{x})}{P(C = L|\vec{X} = \vec{x})} > \lambda$$

with

$$P(C = S|\vec{X} = \vec{x}) = 1 - P(C = L|\vec{X} = \vec{x}).$$

This classification condition is equivalent to

$$P(C = S|\vec{X} = \vec{x}) > t, \text{with } t = \frac{\lambda}{1+\lambda}, \lambda = \frac{t}{1-t}.$$

In the experiments performed by [23] t was set to 0.5 ($\lambda = 1$), 0.9 ($\lambda = 9$) and 0.999 ($\lambda = 999$). For the first case, it is presumed the spam filter flags messages considered to be spam to help the user prioritize the reading of the messages. These flagged messages are not removed from the user's mailbox. Since none of the two types of errors is significantly more severe than the other, this setting seems reasonable.

More work is needed to recover from a blocked legitimate message than deleting a spam message that passed the filter. For this scenario, λ is set to 9 to penalize a legitimate message being blocked slightly more than letting a spam message pass the filter.

In the last scenario blocking a legitimate message is regarded as undesirable as letting 999 spam messages pass the filter. Setting λ to such a high value can be justified when blocked messages are discarded without further processing as most users would consider losing a legitimate message as unacceptable.

Next, a Memory-based technique applied by [10] on the Ling-Spam corpus is discussed. Results obtained by this filter on the Ling-Spam corpus is compared to the performance of the AutoGANN system in Section VII.

V. Memory-based Classification

Memory-based (instance-based) methods [24] store all training instances in a memory structure and use them directly for classification. The multi-dimensional space defined by the attributes in the instance vectors constitutes the simplest form of memory structure. Each training instance vector is represented in this space by a point. A variant of the simple *k*-nearest-neighbour (*k*-nn) algorithm is normally employed by the classification procedure. This algorithm considers the k training instances (its *k-neighbourhood*) closest to the unseen instance and assigns the majority class among these instances to the new unseen instance.

The memory-based classification algorithm implemented in the Tilburg Memory-Based Learner (TiMBL) open source software package was utilized by [10]. This software provides a basic memory-based classification algorithm with extensions such as attribute weighting and efficient computation of the *k*-neighbourhood. TiMBL takes the *k* closest training instances from the unseen instance into account. When more than one neighbour is found at each distance, the algorithm considers many more instances than the *k* neighbours. For these cases, a small value of *k* is chosen to avoid examining instances that are very different from the unseen one. A post-processing stage was added to the basic TiMBL algorithm by [10] to take λ into account. This extension simply multiplies the number of legitimate neighbours by λ before deciding on the majority class in the neighbourhood.

In the following section metrics used to evaluate the performance of the AutoGANN system are examined.

VI. Classification Performance Measures

Classification performance is frequently measured in terms of accuracy (Acc) or error rate ($Err = 1 - Acc$) [10]. Suppose N_L and N_S denote the total number of legitimate and spam messages respectively to be classified by the filter and $n_{A\to B}$ the number of messages that belongs to category A that the filter classifies as belonging to category B with $(A, B) \in \{L, S\}$. Accuracy and error rate can then be defined as

$$Acc = \frac{n_{L\to L} + n_{S\to S}}{N_L + N_S} \qquad \text{and} \qquad Err = \frac{n_{L\to S} + n_{S\to L}}{N_L + N_S}.$$

The measurements for accuracy and error rate assign equal weights to the two types of error ($L \to S$ and $S \to L$), but $L \to S$ is λ times more costly than $S \to L$. For evaluation purposes, each legitimate message is handled as if it were λ messages to make the accuracy and error rate sensitive to the difference in cost. Therefore, when a legitimate message goes successfully through the filter it counts as λ successes and when it is blocked, it counts as λ errors. As a result, the following definitions of weighted accuracy ($WAcc$) and weighted error rate ($WErr = 1 - WAcc$) can be given:

$$WAcc = \frac{\lambda \cdot n_{L\to L} + n_{S\to S}}{\lambda \cdot N_L + N_S}$$

and

$$WErr = \frac{\lambda \cdot n_{L\to S} + n_{S\to L}}{\lambda \cdot N_L + N_S}.$$

Accuracy and error rate values (or their weighted versions) are often deceivingly high. To counter this effect, it is common to compare the accuracy or error rate to that of a simplistic baseline approach. The case used by [10] is where no filter is present, thus legitimate messages are never blocked and spam messages always pass. This leads to the weighted accuracy and weighted error rate of the baseline:

$$WAcc^b = \frac{\lambda \cdot N_L}{\lambda \cdot N_L + N_S}$$

and

$$WErr^b = \frac{N_S}{\lambda \cdot N_L + N_S}.$$

The total cost ratio (TCR) enables the performance of a filter to be easily compared to that of the baseline:

$$TCR = \frac{WErr^b}{WErr} = \frac{N_S}{\lambda \cdot n_{L\to S} + n_{S\to L}}.$$

Higher TCR values suggest better performance. When $TCR < 1$ it is better not to utilize the filter (baseline approach). An intuitive meaning of TCR can be obtained by assuming cost is proportional to wasted time. Therefore, TCR measures how much time is spend manually deleting spam messages when no filter is used (N_S) compared to the time spend manually deleting spam messages that passed the filter ($n_{S\to L}$) plus time needed to recover from legitimate messages mistakenly blocked ($\lambda \cdot n_{L\to S}$).

In the next section experimental results obtained by applying the AutoGANN system to the Ling-Spam corpus are discussed.

These results are compared to the performances of the Naive Bayesian learning method and the Memory-Based technique harnessed by [10].

VII. Experimental Results

As in [10] three experiments were performed by the AutoGANN system. These experiments correspond to the three scenarios for the λ parameter described in Section IV. The number of selected attributes ranged from 25 to 100 in steps of 25 for each scenario[1]. The AutoGANN system also performed feature selection in addition to that performed as part of the preprocessing step. In all the experiments, 10-fold cross-validation was performed and $WAcc$ was then averaged over the ten iterations. Finally, TCR was calculated as $WErr^b$ divided by the average $WErr$ value.

Figures 1 to 6 show the average performance of each learning method in each experiment which includes TCR scores obtained by [10] with the keyword-based Outlook 2000[2] filter. The performance of Outlook was included by [10] as an example of a widely used e-mail reader and does not form part of the main comparison with the AutoGANN system.

Results attained on the three scenarios are discussed next.

A. Scenario 1: Labelling spam messages ($\lambda = 1$)

In the first scenario the misclassification cost is the same for both error types. Figures 1 and 2 show the corresponding results.[3] All three learning techniques improve significantly on the baseline ($TCR = 1.0$) and produce very accurate results. AutoGANN outperforms the other two techniques by a large margin over the interval [50, 100] (Figure 2).

On an individual basis, the techniques performed as follows. The Naive Bayesian classifier achieves the best results for 100 attributes, while TiMBL performs best with a smaller attribute set size of 50 (Figure 1). AutoGANN does best for 100 attributes. Three different values of *k* (1, 2, and 10) were chosen to evaluate TiMBL's performance. From Figure 1 it seems the method performs best for small values of *k*. For $k = 10$ the method improves only slightly on the base case. This can be ascribed to the large number of ties for each of the $k = 10$ distances which leads to a very large neighbourhood taken into consideration (> 500 neighbours). For these cases the filter approximates the default rule which classifies all messages according to the majority class (legitimate in the case of the Ling-Spam corpus). This also explains the insensitivity of the method to the number of attributes for $k = 10$. Compared to the other three methods, Outlook's keyword patterns perform very poorly.

[1]It was found that the AutoGANN system is most effective with 100 or less attributes [11].

[2]Outlook 2000 is a trademark of Microsoft Corporation.

[3]Note than the horizontal and vertical axes of the two figures have different scales.

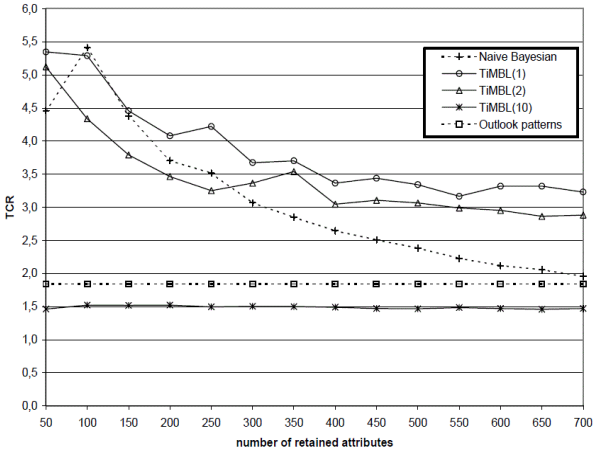

Figure 1: TCR scores for the comparative techniques when $\lambda = 1$ [10].

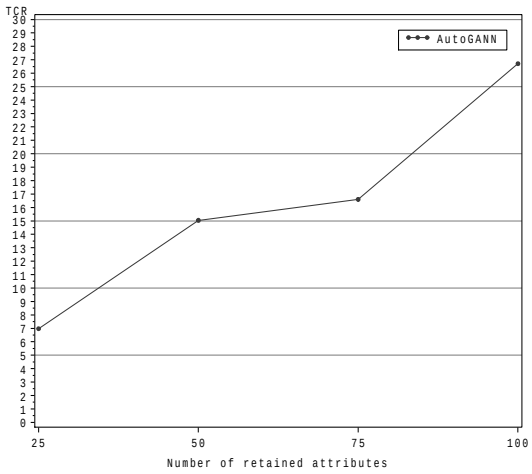

Figure 2: TCR scores for the AutoGANN technique when $\lambda = 1$.

B. Scenario 2: Notifying senders about blocked messages ($\lambda = 9$)

For the second scenario the cost of misclassifying legitimate messages is increased by setting $\lambda = 9$. Figures 3 and 4 show the corresponding results. The most important difference compared to the first scenario is less improvement of the learning techniques over the baseline. As λ increases the performance of the learning techniques relative to the baseline decreases. This is due to the fact that without a filter all legitimate messages are retained which becomes more beneficial as λ increases. Consequently, it becomes harder to "beat" the baseline. As in the first scenario, AutoGANN is clearly superior compared to the other two classifiers (Figure 4). From Figure 3 it can be observed that Outlook's patterns perform below the base case ($TCR < 1.0$). This suggests one is better off not utilizing the Outlook filter.

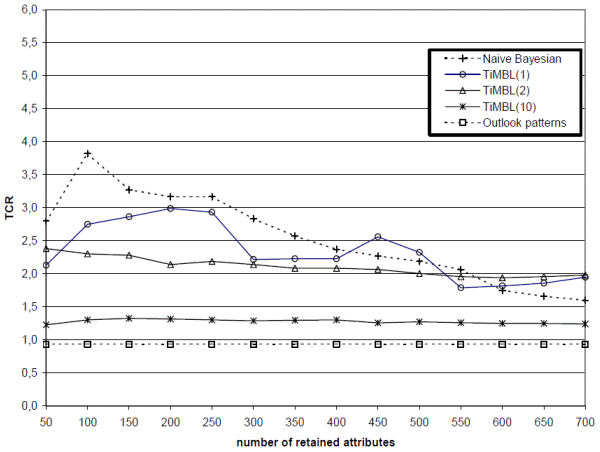

Figure 3: TCR scores for the comparative techniques when $\lambda = 9$ [10].

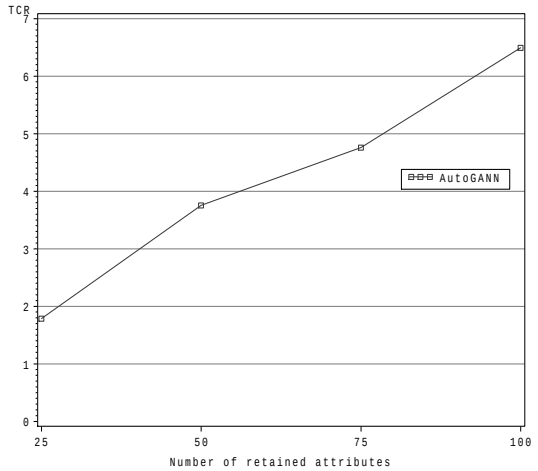

Figure 4: TCR scores for the AutoGANN technique when $\lambda = 9$.

C. Scenario 3: Removing blocked messages ($\lambda = 999$)

For the last scenario a large λ value is used. From Figures 5 and 6 it can be seen that the performance of the learning techniques decreases to such a level that any improvement on the baseline is very hard to achieve. Accordingly, the choice to use any filter at all becomes doubtful. Note that this high λ value was proposed by [23]. TiMBL with $k = 10$ is one exception by performing consistently higher than the base case by a small margin. This is again influenced by the very large neighbourhood, which classifies most messages as legitimate, due to the large value of λ. The Naive Bayesian classifier delivers better performance with 300 attributes, but this is the only point where it outperforms the baseline. Locating the optimal attribute size exactly is infeasible in practical applications and therefore TiMBL for $k = 2$ is the preferred choice. Abrupt fluctuations in the performance of the learning methods can be ascribed to the misclassification of a legitimate message which causes a very large slump in TCR. This effect can be observed, for example, with TiMBL for $k = 2$ and 550 attributes. AutoGANN performs significantly worse than the baseline (In Figure 6, $TCR < 1.0$).

Table 1 summarizes the number of attributes (# attr.) for which each learning technique performs best. In the next, final section, some conclusions are presented.

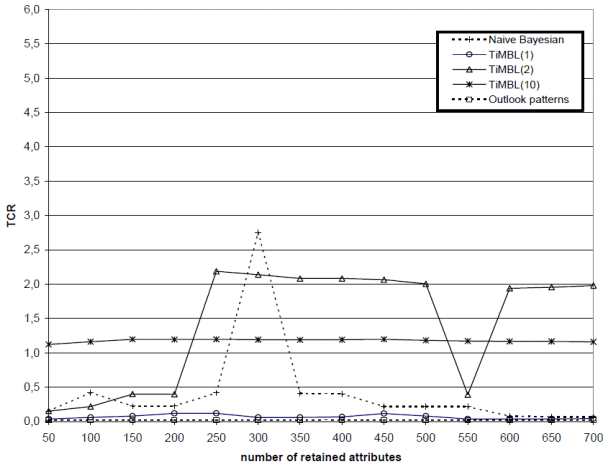

Figure 5: TCR scores for the comparative techniques when $\lambda = 999$ [10].

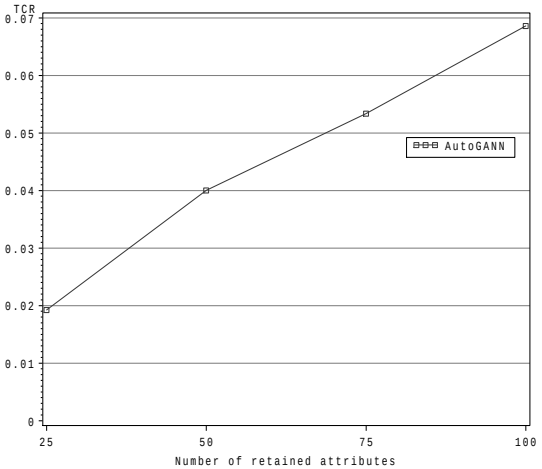

Figure 6: TCR scores for the AutoGANN technique when $\lambda = 999$.

Filter used	λ	# attr.	$WAcc$ (%)	TCR
AutoGANN		**100**	**99.378**	**26.71**
Naive Bayesian		100	96.926	5.41
TiMBL(1)		50	96.890	5.35
TiMBL(2)	1	50	96.753	5.12
Outlook patterns		-	90.978	1.84
TiMBL(10)		100	89.079	1.52
Baseline (no filter)		-	83.374	1
AutoGANN		**100**	**99.666**	**6.50**
Naive Bayesian		100	99.432	3.82
TiMBL(1)		200	99.274	2.99
TiMBL(2)	9	50	99.090	2.38
TiMBL(10)		150	98.364	1.33
Baseline (no filter)		-	97.832	1
Outlook patterns		-	97.670	0.93

Table 1: Results on the Ling-Spam corpus.

Filter used	λ	# attr.	$WAcc$ (%)	TCR
Naive Bayesian		300	99.993	2.86
TiMBL(2)		250	99.991	2.22
TiMBL(10)		250	99.983	1.18
Baseline (no filter)	999	-	99.980	1
TiMBL(1)		200	99.829	0.12
AutoGANN		**100**	**99.709**	**0.07**
Outlook patterns		-	98.952	0.02

Table 1: Results on the Ling-Spam corpus (continued).

VIII. Conclusions

Spam has become one of the major risk security threats in modern organisations and in order to protect enterprises against risks such as computer viruses, phishing, overloading, unnecessary cost etc., efficient security controls, such as spam filters, are necessary.

In this paper a comparison between the Naive Bayesian classifier, a Memory-based classifier and the automated construction algorithm for Generalized Additive Neural Networks was performed to determine the feasibility of the latter technique to filter spam e-mail messages. These techniques were applied to a publicly available corpus using cost-sensitive evaluation measures.

When spam messages are simply to be flagged or when additional means are available to inform senders of blocked messages, the AutoGANN system provides a considerable improvement over the performances of the Naive Bayesian classifier and the Memory-based technique. These findings suggest the AutoGANN system can be used successfully as an anti-spam filter for these two scenarios. Moreover, the AutoGANN system clearly exceeded the performance of the anti-spam keyword patterns of an extensively used e-mail reader.

When no additional means are available and messages are discarded without further processing, a memory-based approach seems to be more viable, but the filter must be configured appropriately with great care.

Acknowledgment

The authors wish to thank SAS® Institute for providing them with Base SAS® and SAS® Enterprise Miner™ software used in computing all the results presented in this paper.

This work forms part of the research done at the North-West University within the TELKOM CoE research program, funded by TELKOM, and THRIP.

References

[1] J. Kagstrom, "Improving naive bayesian spam filtering," Master's thesis, Department for Information Technology and Media, Mid Sweden University, 2005.

[2] K. Tretyakov, "Machine learning techniques in spam filtering," May 2004, Data Mining Problem-oriented Seminar, MTAT.03.177, pp. 60-79.

[3] CNET News, 2004, Spam seen as security risk, http://news.cnet.com/spam-seen-as-security-risk/2100-7355_3-5157275.html, Date of access: 14 March 2012.

[4] K. Jansson, "A model for cultivating resistance to social engineering attacks," Master's thesis, Nelson Mandela Metropolitan University, Port Elizabeth, South Africa, 2011.

[5] Deloitte Touche, Raising the bar, TMT Global Security Study - Key Findings. Report published by Deloitte, 24p, 2011.

[6] Ernst & Young, Global information security survey, http://www.ey.com/GL/en/Services/Advisory/2011-Global-Information-Security-Survey—Into-the-cloud–out-of-the-fog, Date of access: 16 March 2012, 2011.

[7] G. Caruana and M. Li, "A survey of emerging approaches to spam filtering," *ACM Computing Surveys*, vol. 44, no. 2, pp. 1–27, 2012, article 9.

[8] E. Blanzieri and A. Bryl, "A survey of learning-based techniques of email spam filtering," University of Trento, Information Engineering and Computer Science Department, Italy, Tech. Rep. DIT-06-056, January 2008.

[9] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to spam filtering," *Expert Systems with Applications*, vol. 36, pp. 10 206–10 222, 2009.

[10] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. D. Spyropoulos, and P. Stamatopoulos, "Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach," in *Proceedings of the workshop 'Machine Learning and Textual Information Access'*, H. Zaragoza, P. Gallinari, and M. Rajman, Eds. Lyon, France: 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (KDD-2000), September 2000, pp. 1–13.

[11] J. V. Du Toit, "Automated construction of generalized additive neural networks for predictive data mining," Ph.D. dissertation, School for Computer, Statistical and Mathematical Sciences, North-West University, South Africa, May 2006.

[12] P. McCullagh and J. A. Nelder, *Generalized Linear Models*, 2nd ed., ser. Monographs on Statistics and Applied Probability. London: Chapman and Hall, 1989, vol. 37.

[13] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press, 1995.

[14] B. D. Ripley, *Pattern Recoqnition and Neural Networks*. Cambridge, United Kingdom: Cambridge University Press, 1996.

[15] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 14, pp. 35–62, 1998.

[16] T. J. Hastie and R. J. Tibshirani, "Generalized additive models," *Statistical Science*, vol. 1, no. 3, pp. 297–318, 1986.

[17] ——, *Generalized Additive Models*, ser. Monographs on Statistics and Applied Probability. London: Chapman and Hall, 1990, vol. 43.

[18] S. N. Wood, *Generalized Additive Models: An introduction with R*, ser. Texts in Statistical Science. London: Chapman & Hall/CRC, 2006.

[19] W. J. E. Potts, "Generalized additive neural networks," in *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 1999, pp. 194–200.

[20] K. N. Berk and D. E. Booth, "Seeing a curve in multiple regression," *Technometrics*, vol. 37, no. 4, pp. 385–398, 1995.

[21] M. Ezekiel, “A method for handling curvilinear correlation for any number of variables,” *Journal of the American Statistical Association*, vol. 19, no. 148, pp. 431–453, December 1924.

[22] W. A. Larsen and S. J. McCleary, “The use of partial residual plots in regression analysis,” *Technometrics*, vol. 14, no. 3, pp. 781–790, August 1972.

[23] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, “A bayesian approach to filtering junk e-mail,” Learning for Text Categorization Papers from the AAAI Workshop, Madison Wisconsin, Tech. Rep. WS-98-05, 1998.

[24] T. M. Mitchell, *Machine Learning*, ser. McGraw-Hill Series in Computer Science. Boston, Massachusetts: WCB/McGraw-Hill, 1997.