A Content-Based Spam E-Mail Filtering Approach Using Multilayer Percepton Neural Networks

Article *in* International Journal of Engineering Trends and Technology · November 2016

DOI: 10.14445/22315381/IJETT-V41P210

A Content-Based Spam E-Mail Filtering Approach Using Multilayer Percepton Neural Networks

A.Sesha Rao[#1], P.S.Avadhani [#2], Nandita Bhanja Chaudhuri[#3]

[#1]*Dept. of Computer Science & Systems Engineering, AU College of Engineering, Visakhapatnam, India.*
[#2]*Principal, AU College of Engineering, Andhra University, Visakhapatnam, India*
[#3]*Dept. of Information Technology, Vignan's Institute of Engineering for Women, Visakhapatnam, India*

Abstract— *Nowadays increased spam e-mails are causing inconvenience to internet users and organizations and are considered as a serious wastage of resources, time, memory, space and efforts. Therefore, it is crucial to have an automatic e-mail classification system for the identification of spam e-mails. Spam mails need to be classified and separated from ham (non-spam) mails as they are the source of financial loss and annoyance for the recipients. The spam e-mail classifier performance can be greatly enhanced with the use of Artificial Neural Network classification. It has capability of learning huge amount of data with high dimensionality in a better way. In this paper, Multilayer Perceptron and Back Propagation Training algorithm is explored where 'generalized delta' rule is used for weight adjustments for hidden layers. The Perceptron uses Back Propagation Learning model for calculating its gradient. For fast convergence the learning rate η is changed for every iteration which is proportional to the negative gradient of the instantaneous error with respect to η. To avoid the local minima problem the weights are initialized to small random numbers which are uniformly distributed in the range [$-\alpha/\sqrt{Ni}$, $+\alpha/\sqrt{Ni}$], where Ni is the number of inputs, and α takes value in (1, 3). In this paper, four Multilayer Perceptron (MLP) Network models are constructed. For testing our model bench mark data drawn from UCI, Machine learning Repository is employed for training the neural network. The results of our MLP model are reasonable in terms of TP rate, FP Rate, Accuracy, Precision, Recall, F-measure, MCC, ROC Area, PRC Area.*

Keywords —back propagation, delta rule, F-measure, hidden layers, learning rate, local minima, MCC, perceptron, precision, PRC Area, recall, ROC.

I. Introduction

Along with the growth of the Internet and e-mail, there has been a dramatic growth in spam in recent years. E-mail spam, known as Unsolicited Bulk Email (UBE), junk mail, or Unsolicited Commercial Email (UCE), is the practice of sending unwanted e-mail messages, frequently with commercial content, in large quantities to an indiscriminate set of recipients [1].

These junk mails can contain various types of messages such as pornography, commercial advertising, doubtful product, viruses or quasi legal services [1]. The inverse of "spam" email is called "ham" which needs by recipients. [2] Links on social networks that take us to free downloads, easy income, pornography and unsolicited text messages offering loans, low priced products etc. Why we need spam detection? Spam causes annoyance and wastes user's time to regularly check and delete this large number of unwanted messages. Flooding of mailboxes with spam e-mails waste storage space and overload the server; thus it may lead to losing legitimate e-mails, degrading the server performance, or even make it totally unavailable. Hence, spam consumes network bandwidth and server storage space.[3] Hence it is crucial to have automatic spam filtration system for every individual user.

[4] The intent of spam content is to catch eyeballs and conduct users to take some actions. Spam email usually contains links to entice the users to click on those links and visit particular Web sites, which we call spam-advertised sites.

Spam has grown steadily since the early 1990s. Recent reports show that spam accounts for approximately 70-90% of email traffic sent on the Internet-about 70 billion messages a day [5,6,7]. Although several spamming botnets were taken down, such as Rustock, Kelihos, and Grum [7,8], spam remains a serious security threat and attackers keep establishing new spam campaigns.

The information and analysis of E-mail statistics 2015-2019 – Executive Summary report, based on primary research was conducted by The Radicati Group, Inc. It reported that over next four years, the average number of email accounts per user ratio will grow from an average of 1.7 to 1.9 accounts. In the year 2015, the number of emails sent and received per day is around 205 billion [9].

The task of spam filtering is to rule out deceptive messages automatically from a user's inboxes. There are many techniques evolved which seek, to identify whether a message is spam or ham based on the content and other characteristics of the message which improves the accuracy of spam filters. These machine learning algorithms are important in the continuous fight against spam. In spite there are large number of methods and techniques available to filter spam, but the volume of spam on the internet is still rising. [10]

Due to security of information, a spam filter must be placed in computer network. There are various techniques which have been used by the different authors among these data mining is one of the popular techniques to develop classifier to classify spam and non-spam data. In order to tackle problems faced by users due to spam e-mail, it is necessary to classify them with the help of intelligent and robust classifier. These classifiers should have the capability to classify spam e-mail against non-spam e-mail. The spam e-mail classifier performance can be greatly enhanced with the use of Artificial Neural Network (ANN) classification algorithms. Artificial Neural Network is a powerful tool used for classification of data [11], it has capability of learning huge amount of data with high dimensionality in better way. There are various parameters of ANN to be set to tune for the better performance of neural network model, these are learning rate, architecture of ANN and momentum term, these all parameters play a very important role in improving the accuracy of ANN model.

In this paper we have used Multilayer Perceptron Network model with BPL. Multilayer Perceptrons are network of linear classifiers. The Back Propagation Learning (BPL) used generalised delta rule for weight adjustments in hidden layers. BPL is trained on spam e-mail data set with different partitions and different learning rate. Different performance measures like precision, recall, F-measure and accuracy of the best model with feature selection have also been calculated.

Rest of the paper is organised as follows. The related work is presented in section II. The proposed design and implementation of the mathematical model for spam mail filtering using Multilayer Perceptron network is described in section III. In section IV, we present the experiential results and evaluation process. Conclusion and scope of the future work are given in section V.

1.1 Artificial Neural Network (ANN)

ANN is a computational model, which is based on Biological Neural Network. To build ANN, artificial neurons, also called nodes are interconnected. An artificial neuron is an abstraction of biological neuron and the basic unit in ANN. Arranging neurons in layers mimics the layered structure of certain portions of brain. The power of neural computation comes from connecting neurons into networks. There are several classes of neural networks, classified according to their learning mechanisms, but structurally they are classified into three fundamentally different classes.

i) Single layer artificial neural networks.
ii) Multilayer artificial neural networks.
iii) Recurrent Networks.
(i) and (ii) are called feed forward neural networks.

In single layer artificial neural networks, the inputs are directly supplied to the outputs through a series of weights. Multilayer artificial neural networks map the input to suitable output via hidden layers. In recurrent neural network, the connection between units forms a directed cycle.

Multilayer artificial neural network is a feed forward neural network which maps a set of inputs x onto a set of outputs y using multi weights connections. It consists of an input layer, an output layer, and one or more hidden layers.

II. RELATED WORK

Several Machine Language techniques are brought together to explore the best methods [12,13]. The most commonly used technique is ANN as it gives better classification results. Authors Ali Rodan et al., [14] developed an MLP neural network model with Biogeography Based Optimization for identifying e-mail spam. In this work, MLP is trained using Biogeography Based Optimization based on two different spam data sets and compared with other MLPs trainer with back propagation algorithm and common meta heuristic algorithms: Genetic Algorithm, Distributed System Operators, Differential Evolution and Ant Colony Optimization [15]. An anti-spam filtering was presented by his techniques are centred on artificial neural network (ANN) and Bayesian networks.

Rodan et. al., [15] proposed the application of Miltilayer Perceptron for the purpose of e-mail classification where the weights of this network model are found using a new nature-inspired meta heuristic algorithm called Biogeography Based Optimization (BBO). Experiments are conducted using two different datasets revealed that MLP model trained by BBO algorithm gets high generalization performance compared to any other optimization methods.

WangI et. al., [16] proposed Hill Climbing, Simulated Annealing, and Threshold Accepting optimization techniques as feature selection algorithms. Performance of the above three techniques are compared with the Linear Discriminate Analysis. The experimental results show that the proposed strategies not only reduce the dimensions of the e-mail, but also improve the performance of the classification filter. The classification accuracies are 90.0% for LDA, 93.6% for HC, 94.6% for TA and 95.5% or SA as compared to 88.1% for the system without feature selection.

Qiuming et. al. [17], projected a method to filter text spam e-mails using Back Propagation Neural

Network model. In the experiment, they utilized spamassassin corpus as data set, and implemented cross validation method to inspect network simulation result. Experimental result showed that by taking different model parameters, the network will gives different results. To get more accurate classification results, a large number of data sets need to be trained and shall learn how to optimize the network model to achieve better filter results.

Ismailia Idris [18] has proposed neural network model for spam e-mail classification, he has compared the result of both neural network and SVM models in terms of accuracy.

EL-Sayed M., EL-Alfy et al, [19] have introduced abduct network model for spam email classification and compared the results of abduct network model with other Group Method of Data Handling (GMDH) based network.

Harshal Deshmukh et al, [20] used ANN to detect text as well as image based spam to achieve the objective and they have used Enron corpus's data set of spam emails. Various spam detection steps such as pre-processing step, or data clearing step, representation of data and classification of spam or non spam e-mail messages are discussed.

Mohamad and Selamat [21], worked on, a spam filtering technique, which implements a combination of two types of feature selection methods before its classification task. The authors attempted to conduct an experiment to test the efficiency of the proposed Hybrid Feature Selection, which is a combination of Term Frequency Inverse Document Frequency (TFIDF) with the rough set theory in spam email classification problem. The result shows that the proposed Hybrid Feature Selection returns a good result.

Loredana Firte et al,[22] presented a new approach for spam detection filter. The solution proposed is an offline application that uses the K-Nearest Neighbour algorithm and pre- classified email data set for the learning process. K-Nearest Neighbor algorithm classified the messages which are based on features extraction from the emails properties and content .

Nosseir, Khaled Nagati et al, [23] performed a work, Intelligent Word-Based Spam Filter Detection Using Multi-Neural Networks‖. They proposed a character-based technique. A multi-neural networks classifier is used by this approach. A normalized weight values derived from the ASCII value of the word characters are used to train the neural network.

Reena Sharma et al, proposed [24] an efficient spam filtering technique based on neural network. The technique used is RBF a neural network technique in which neuron are trained. The results obtained by using this technique are compared with SVM. The parameter for comparison is precision and accuracy.

Thomas et. al., [25] their study identifies five feature selection techniques namely, namely TFDF,MI, WMI, CDM and Chi-square which are used in the general text classification for spam filtering. Also, the classification and prediction is performed using different entities of email such as header, body and subject that can be used for effective identification of spam mails. They presented a comparative study of different feature selection methods. Through extensive experiments we demonstrated that Weighted Mutual Information feature selection with header and body of the emails is efficient in email classification based on SVM**.**

Mohammad et. al., [26], utilized Fisher filtering feature selection algorithm. The best error rate is 0.0089. They implemented CAV algorithm which reduced the error rate by 39% compared with other Rnd Tree and SVM.

Moreover, several ML techniques are combined together to produce a more accurate and robust detectionvmethods. For instance, Artificial Neural Network (ANN) is a commonly used technique as it gives accuratevclassification results [27]. ANNs are inspired by the biological neural systems. The most popular and applied type of ANNs is the Feedforward Multilayer Perceptron (MLP).

III. DESIGN AND IMPLEMENTATION

The model of a Multilayer Perceptron is designed and implemented in this paper to classify spam e-mails. It is flexible structure and non-linearity transformation to accommodate latest spam mails. The experiments are done on publicly available UCI Machine Learning Repository.

Perceptron is a general feed-forward single layer net work with one output neuron. It learns to separate the input pattern. It uses supervised learning algorithm. Hence, the training set consists of a set of input vectors, along with its desired target vector. Input vector components take on a continuous range of values; and target vector components are binary valued. After training, the network accepts a set of continuous inputs and produces the desired binary valued outputs. The perceptron separates linearly separable input patterns. It cannot separate the input pattern sets which are linearly inseparable.

Multilayer neural networks may be formed by simply cascading a group of single layer networks. Such networks provide increase in computational power over a single layer networks by having non linear activation function between layers. Hence multilayer networks consist of one or more hidden layers apart from input layer and output layer. The input layer neurons receive the input signals, perform no computation, and the output layer neurons produce simply the output signals. The function of hidden neurons is to intervene between the external inputs and the network outputs. Connection between neurons is represented as synaptic weights. A two layered neural network require two synaptic weight matrices $w_{i,j}$ and $k_{i,j}$ as shown in our model Fig. 1.

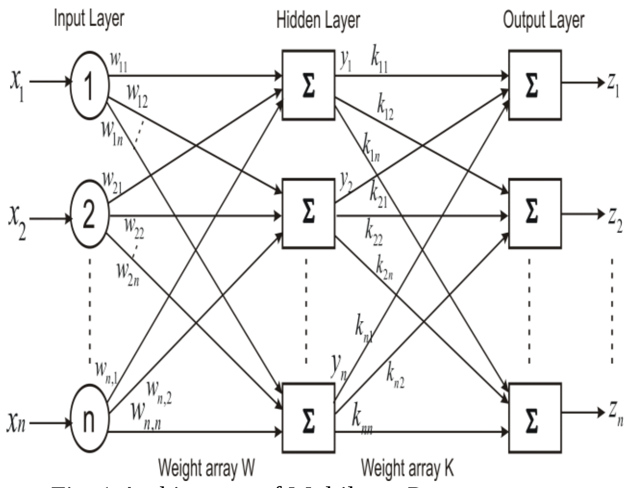


Fig. 1 Architecture of Multilayer Perceptron

Calculation of output for the first (hidden layer) layer consists of multiplying the input vector by the first weight matrix, w_{ij}. Output of hidden layer is given by

$$y_i = \sum_{i,j=1}^{n} x_{i.} .w_{i,j} \qquad \rightarrow (1)$$

The output of output layer is given by

$$z_i = \sum_{i,j=1}^{n} y_{i.} .k_{i,j} \qquad \rightarrow (2)$$

Where

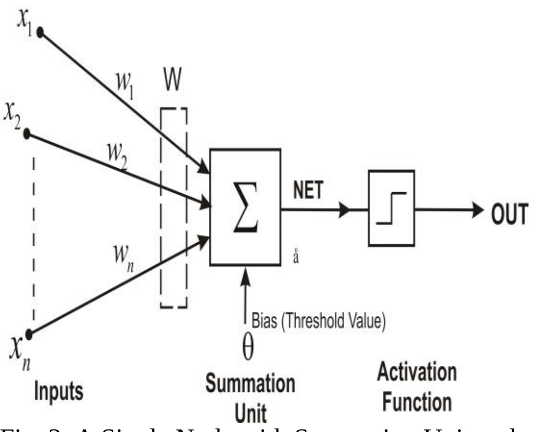
$x_i = (x_1, x_2, ...x_n)$

$y_i = (y_1, y_2, ...y_n)$

$z_i = (z_1, z_2, ...z_n)$ and

$w_{i,j}$ & $k_{i,j}$ are synaptic weight matrices of hidden layer and output layer.

Each neuron consists of a summation function and activation function as shown below. A single artificial neuron with activation function is shown in Fig. 2.



Fig, 2. A Single Node with Summation Unit and Activation Function

Fig. 2 shows the neuron used as fundamental building block for ANN, A set of inputs $X(x_1,x_2,....,x_n)$ are applied either from external or previous layer. Each of these inputs is multiplied by a connecting weight w_i, and the weighted sum is given by

$$NET = x_1 w_1 + x_2 w_2 + ... x_n w_n$$

$$= \sum_{i=1}^{n} x_i \ w_i \qquad \rightarrow (3)$$

After the output 'NET' is calculated activation function F is applied to it there by producing the final output signal 'OUT'.

$$OUT = F(NET) = \frac{1}{1+e^{-NET}} \qquad \rightarrow (4)$$

Where, F is sigmoid function shown in fig. 3.

$$F'(NET) = OUT \ (1-OUT) \qquad \rightarrow (5)$$

Equation (4) shows that sigmoid function is differentiable and provides the required non linearity and automatic gain control.

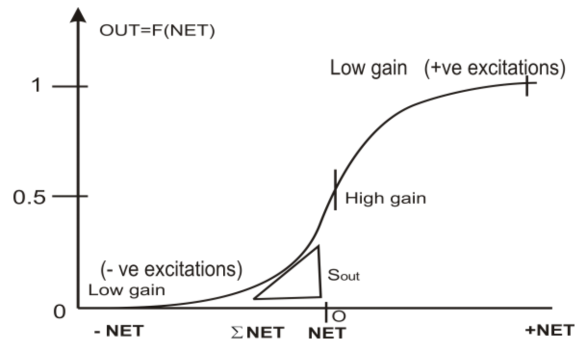


Fig. 3 Sigmoid Function

3.1Back Propagation Learning Algorithm for MLP

The difficulty with the multilayer perceptron is adjusting the weights of hidden layers during training process. . More number of hidden layers increases the difficulties. So, back propagation learning is a better substitute to train Multi layer perceptron network.

It adopts supervised learning. It is one of the ways to train artificial neural networks used along with an optimization method such as steepest descent. The method calculates the derivative of a loss function with respect to all the weights in the network, so that the derivative is fed to the optimization method which in turn uses it to update the weights in hidden layers. Hence, it attempts to minimize the loss function.

Back propagation requires a known, desired output (target vector) for each input vector in order to calculate the loss function gradient. It is therefore usually considered to be a supervised learning method; although it is also used in some unsupervised networks such as auto encoders etc. It is a generalization of the delta rule to multi-layered feed-forward networks, made possible by using the chain rule to iteratively compute gradients for each layer. Back propagation requires that the activation function used by the nodes be differentiable.

Training is accomplished by sequentially applying input vectors while adjusting network weights according to predetermined procedure. Here we adopt supervised training. In supervised training we present both input and desired (target) output vectors to the network.

3.2 Preceptron Training

The training set consists of a set of input vectors each with its desired target vector. Input vector components take on a continuous range of values and target vector components are binary valued. Perceptron training cannot be applied for multilayer networks. However, multi layer network provides increase in computational power over a single layer network by having non linear activation function between layers.

To update the weights one must calculate the error, i.e., difference between the actual output and the desired (target) output. This error is easily calculated at the output layer. At the hidden layers, however, there is no idea about the target vector. So, some other technique must be used to calculate an error at the hidden layers that will cause optimization of the output error.

MLP algorithm is a systematic method for training multilayer neural network. It considers only feed forward networks. MLP provides automatic gain control where,

(i) Large signals can be accommodated by the network without saturation.

(ii) While the small signals are allowed to pass through without excessive attenuation.

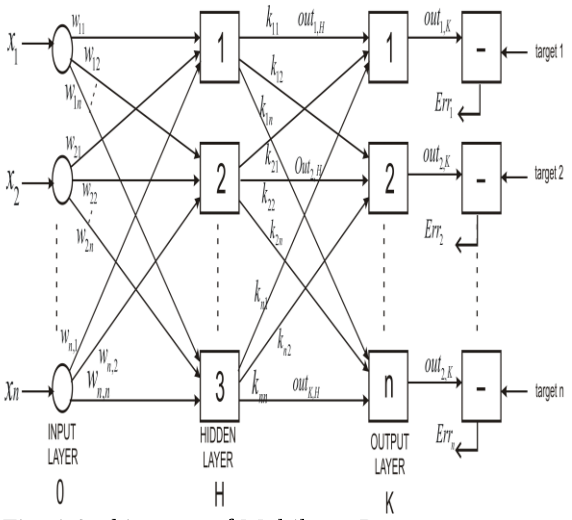


Fig. 4 Architecture of Multilayer Perceptron

A typical MLP starts as a network of nodes arranged in three layers, input, hidden, output layers. The architecture of MLP is shown in Fig. 4. There is no theoretical limit on number of hidden layers but 2 or 3 layers are preferable. First layer neurons serve as distribution points. They perform no computation. Each neuron in the subsequent layers produces 'NET' and output signals after applying sigmoid function. The input layer is designated as 0. The weights in layer 0 terminate on the neurons of subsequent layer H and so on.

3.3 MLP Algorithm

3.3.1 Two Passes of Computation

The algorithm constitute a 'forward pass', in that signal propagates from the network input layer to its output layer and a 'reverse pass', here, calculates error signal propagates backward through the network where it adjusts the weights. Thus, the MLP learning algorithm cycles through two distinct phases a 'forward pass', followed by 'backward pass' through the layers of network. The algorithm alternates between these phases several times as it scans the training data.

Calculations are performed on a layer by layer basis. First the outputs of the neuron in hidden layer H are calculated. These are then used as inputs to subsequent hidden layers followed by output layer K. Then layer K neurons outputs are calculated. Each of the network output is subtracted from its corresponding component of the target vector. This error is used to adjust the weights of the network in all the layers. After enough repetitions, the error between actual output and target outputs should be reduced to an acceptable value. At this point, the network is used for recognition and weights are not changed further.

i) Forward Pass: The first pass is referred to as forward pass. In this pass the signal propagates from the network input layer to its output layer. During this the synaptic weights remain unaltered throughout the network and the function signals are computed on a node by node basis. The 'NET' value of each node in the first layer is calculated as weighted sum of its node's output. The final output of each node is computed using activation function. The error is calculated at the output layer as follows.

$$NET_{i,H} = \sum_{i,j=1}^{n} x_i . w_{i,j} \quad \rightarrow (6)$$

$$OUT_{i,H} = F\left(NET_{i,H}\right) = \frac{1}{1+e^{-NET_{I,H}}} \quad \rightarrow (7)$$

Where, w $_{i,j}$ is the synaptic weight connecting neuron i to neuron j in the hidden layer. x_i's are the input signal of node i.

The output vector of one layer is the input vector to the next layer. So calculating the output of final layer requires the application of eqn. 7 in each layer starting from the network input layer to its output layer.

ii) Reverse Pass: This phase begins with the computation of error at each node in the output layer. The error in the output layer is calculated by subtracting the computed output from the desired output. The reverse pass on the other hand, starts at the output layer by pushing the computed error signals backward through the network layer by layer. Because a target value is available for each neuron in the output layer only adjusting the associated weights is usually accomplished using a modified delta rule. So, we compute local gradient δ for each node.

ii. a) Weight adjustment for output layer

For a node located in the output layer, the δ is simply equal to the error signal of that node multiplied by first derivative of sigmoid function . If K is the output layer, error of the nqth node in the Kth layer is given by

$$e_{q,K} = (target\ output - campted \quad output) \rightarrow (8)$$

$$Local\ gradient,\ \delta_{q,K} = F'(NET_{q,K})\ e_{q,K}$$

$$= out(1 - out).e_{q,K} \rightarrow (9)$$

$$\Delta w_{pq,K} = \eta\ \delta_{q,k}.out_{P,H} \rightarrow (10)$$

$$w_{pq,K}(t+1) = w_{pq,K}(t) + \Delta w_{pq,K} \rightarrow (11)$$

$\Delta w_{pq,K}$: Weight factor from node p in the hidden layer to node q , in the output layer

$w_{pq,K}(t)$:The value of weight from neuron p in the hidden layer to node q in the output layer at step t.

$w_{pq,K}(t+1)$: Value of the weight at step t+1 (after adjustment)

$\delta_{q,K}$:Local gradient from node q in the output layer

Out $_{p,,H}$: the value of out from node p in the hidden layer H.

Out : output of node in output layer K.

η : learning rate.

ii. b) Weight adjustment for hidden Layer

MLP trains the hidden layers by propagating the output error back through the network layer by layer adjusting the weights at each layer. Since hidden layers have no target vector, the above training process may not work. MLP trains hidden layers by propagating the output layer 'error', δ, back through the network layer by layer, there by adjusting weights in each layer.

Consider a single node p, in the hidden layer just before the output layer. In the forward pass, this node propagates its output (OUT) to nodes in the output layer through the inter connecting weights. During training these weights operate in reverse direction, passing value of δ from the output layer back to the hidden layer.

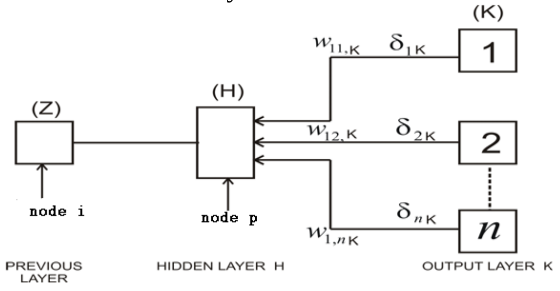


Fig. 5 Weight Adjustments in Hidden Layer

Local gradient $\delta_{p,H}$ from node p in hidden layer H is obtained as a product of the associated derivative $F'(NET_{P,H})$ and the weighted sum of the local gradients $(\delta_{1,K}, \delta_{2,K}, ..., \delta_{n,K})$ computed for the neurons in the next hidden layer or output layer that are connected to node p.

$$\delta_{P,H}(t) = F'(NET_{P,H}) \sum_{q=1}^{n} \delta_{q,K}.w_{pq,K}$$

$$= out_{p,H}(1 - out_{p,H}) \sum_{q=1}^{n} (\delta_{q,K}, w_{pq,K}) \rightarrow (12)$$

With δ in hand, the weights feeding the first hidden layer can be adjusted as

$$\Delta w_{pq,H} = \eta\ \delta_{P,H}.out_{i,H} \rightarrow (13)$$

Updated weight of hidden layer is given by

$$w_{pq,H}(t+1) = w_{pq,H}(t) + \Delta w_{pq,H} \rightarrow (14)$$

Where $w_{pq,H}(t).\ w_{pq,H}(t+1), \delta_{p,H}, out_{i,H}$ are defend as above

For each node in a given hidden layer, the local gradient and all weight associated with that layer must be adjusted. This is repeated moving back towards the input layer by layer, until the weights are adjusted.

The performance of MLP law depend on the initial setting of the weights, learning rate parameter, activation formation of the units and presentation of data.

3.4 Weight Initialization

Owing to its gradients descent nature, backdrop is very sensitive to initial conditions if w^0 is close to local minima then the convergence will be very fast. On the other hand back drop convergence very slow if w^0 starts the search in a relatively flat region of the error surface. (for away from local minima). In practice the weight are normally initiated to small random numbers that are uniformly distributed in a small range of values. The range is typically $\left[-\alpha/\sqrt{N_i}, +\alpha/\sqrt{N_i}\right]$ where N_i is the number of inputs to the ith unit. The value of α is it typically in the range (1, 3)

3.5 Learning Rate

Better convergence in learning can be achieved by adopting learning rate, η, which are suitable for each iteration. For this the change is made proportional to the negative gradient of the instantaneous error with respect to η.

$$\Delta\eta_{ji}(t+1) = -\gamma \frac{\partial e^2(t)}{\partial \eta_{ji}(t)} \qquad \rightarrow (15)$$

Where γ is proportionality constant

3.6 Momentum Term

One way to increase the rate of learning is by using a momentum term in the weight change (Rumel hart, Hinton and Williams 1986). The method involves adding a term to the weight factor that is proportional to the amount of previous weight change. Once an adjustment is made, it is 'remembered' and serves to modify all the subsequent weight adjustments. The adjustment eqns. are

$$\Delta w_{pq,K}(t+1) = \eta\left(\delta_{q,K}\ .out_{\ p,H}\right) + \alpha\left[\Delta w_{pq,K}(t)\right] \qquad \rightarrow (16)$$

α : is momentum coefficient is commonly used set to around 0.9

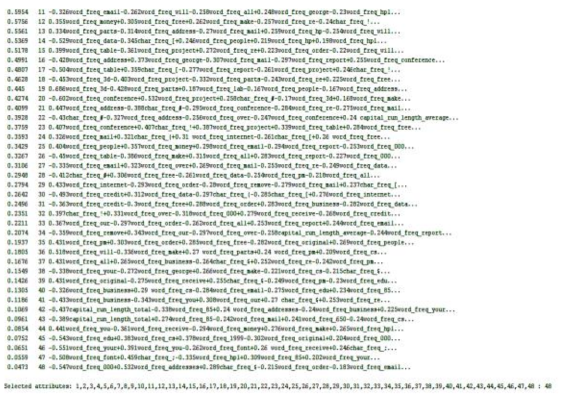
3.7 Implementation of the Model

3.7.1 Dataset Collection

Data is collected from the UCI Machine Learning Repository. It is a standard dataset which are applied to this model. It contains both samples of spam and non-spam e-mail data. The dataset has 57 continuous attributes and 1 nominal class label. There are 4501 number of instances among that 1813 are spam's which is around 39.4%.
Source:
http://www.ics.uci.edu/~mlearn/MLRepository.html

3.7.2 Preprocessing

The data that is collected is in Comma Separated Version (.CSV) file format. For applying the neural network techniques, a tool called Weka is used. This tool supports Attribute Relation File Format (ARFF). So the data had been converted to ARFF. The original data set contains 58 attributes but only the important attributes needs to be selected. This is called as attribute selection mechanism. To achieve this, an algorithm called Principal Component Analysis (PCA) is used along with Ranker Search method which is shown in Fig.6. The resultant of this is 47 continuous attributes and 1 nominal class label.



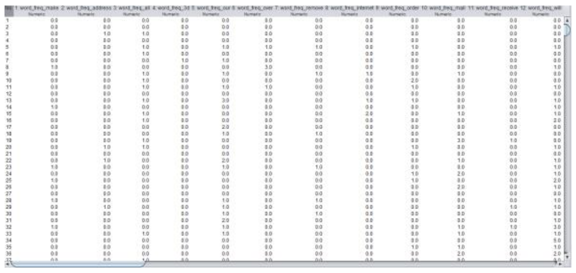
```
0.5954   11 -0.326word_freq_email-0.262word_freq_will-0.258word_freq_all+0.248word_freq_george-0.23word_freq_hpl...
0.5756   12 0.355word_freq_money+0.305word_freq_free+0.262word_freq_make-0.257word_freq_re-0.24char_freq_!...
0.5561   13 0.334word_freq_parts-0.314word_freq_address-0.27word_freq_mail+0.259word_freq_hp-0.254word_freq_will...
0.5369   14 -0.529word_freq_data-0.345char_freq_[+0.246word_freq_people+0.219word_freq_hp+0.198word_freq_hpl...
0.5178   15 0.399word_freq_table-0.361word_freq_project+0.272word_freq_re+0.223word_freq_order-0.22word_freq_will...
0.4991   16 -0.428word_freq_address+0.373word_freq_george-0.307word_freq_mail-0.297word_freq_report+0.255word_freq_conference...
0.4807   17 -0.504word_freq_table+0.359char_freq_[-0.277word_freq_report-0.261word_freq_project+0.246char_freq_!...
0.4628   18 -0.453word_freq_3d-0.403word_freq_project-0.332word_freq_parts-0.243word_freq_re+0.229word_freq_free...
0.445    19 0.686word_freq_3d-0.428word_freq_parts+0.187word_freq_lab-0.167word_freq_people-0.167word_freq_address...
0.4274   20 -0.602word_freq_conference+0.532word_freq_project+0.258char_freq_#-0.17word_freq_3d+0.168word_freq_make...
0.4099   21 0.447word_freq_address-0.388char_freq_#-0.295word_freq_conference-0.284word_freq_re-0.275word_freq_mail...
0.3928   22 -0.43char_freq_#-0.327word_freq_address-0.256word_freq_over-0.247word_freq_conference+0.24 capital_run_length_average...
0.3759   23 0.407word_freq_conference+0.407char_freq_!+0.387word_freq_project+0.339word_freq_table+0.284word_freq_free...
0.3593   24 0.328word_freq_mail+0.321char_freq_(+0.31 word_freq_internet-0.261char_freq_[+0.26 word_freq_free...
0.3429   25 0.404word_freq_people+0.357word_freq_money+0.298word_freq_email-0.294word_freq_report-0.253word_freq_000...
0.3267   26 -0.45word_freq_table-0.386word_freq_make+0.315word_freq_all+0.283word_freq_report-0.227word_freq_000...
0.3106   27 -0.335word_freq_email+0.323word_freq_over+0.269word_freq_mail-0.253word_freq_re-0.249word_freq_data...
0.2948   28 -0.412char_freq_#+0.308word_freq_free-0.261word_freq_data-0.254word_freq_pm-0.218word_freq_all...
0.2794   29 0.433word_freq_internet-0.293word_freq_order-0.28word_freq_remove-0.279word_freq_mail+0.237char_freq_[...
0.2642   30 -0.493word_freq_credit+0.312word_freq_data-0.297char_freq_(-0.285char_freq_[+0.276word_freq_internet...
0.2496   31 -0.363word_freq_credit-0.3word_freq_free+0.288word_freq_order+0.283word_freq_business-0.282word_freq_data...
0.2351   32 0.397char_freq_!+0.331word_freq_over-0.318word_freq_000+0.279word_freq_receive-0.268word_freq_credit...
0.2211   33 0.367word_freq_our-0.297word_freq_order-0.262word_freq_all+0.253word_freq_report+0.244word_freq_email...
0.2074   34 -0.359word_freq_remove+0.343word_freq_our-0.297word_freq_over-0.258capital_run_length_average-0.244word_freq_report...
0.1937   35 0.431word_freq_pm+0.303word_freq_order+0.285word_freq_free-0.282word_freq_original+0.269word_freq_people...
0.1805   36 0.518word_freq_will-0.336word_freq_make+0.27 word_freq_parts+0.24 word_freq_pm+0.209word_freq_cs...
0.1676   37 0.431word_freq_all+0.265word_freq_business-0.264char_freq_$+0.252word_freq_re-0.242word_freq_pm...
0.1549   38 -0.338word_freq_your-0.272word_freq_george+0.266word_freq_make-0.221word_freq_cs-0.215char_freq_$...
0.1426   39 0.431word_freq_original-0.275word_freq_receive+0.255char_freq_$-0.249word_freq_pm-0.23word_freq_edu...
0.1305   40 -0.326word_freq_business+0.29 word_freq_cs-0.284word_freq_email-0.275word_freq_edu+0.234word_freq_85...
0.1186   41 -0.433word_freq_business-0.343word_freq_you+0.308word_freq_our+0.27 char_freq_$+0.253word_freq_re...
0.1069   42 -0.437capital_run_length_total-0.338word_freq_85+0.24 word_freq_addresses-0.24word_freq_business+0.225word_freq_your...
0.0961   43 -0.389capital_run_length_total+0.274word_freq_85-0.242word_freq_mail+0.241word_freq_650-0.24word_freq_cs...
0.0854   44 0.441word_freq_you-0.361word_freq_receive-0.294word_freq_money+0.276word_freq_make+0.265word_freq_hpl...
0.0752   45 -0.543word_freq_edu+0.383word_freq_cs+0.378word_freq_1999-0.302word_freq_original+0.204word_freq_000...
0.0651   46 -0.551word_freq_your+0.391word_freq_you-0.262word_freq_font+0.26 word_freq_receive+0.246char_freq_;...
0.0559   47 -0.508word_freq_font+0.459char_freq_;-0.335word_freq_hpl+0.309word_freq_85+0.202word_freq_your...
0.0473   48 -0.547word_freq_000+0.532word_freq_addresses+0.289char_freq_$-0.215word_freq_order-0.183word_freq_email...

Selected attributes: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48 : 48
```

Fig.6: Result of Principal Component Analysis

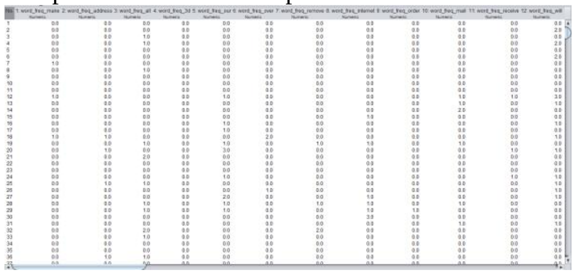
After this, the data is partitioned into two set-70% for training and 30% for testing.
The experiments are also conducted by collecting variant samples of the datasets which are demonstrated in Table-1, Table-2, Table-3, and Table-4.

TABLE I
Sample1: 25% of the Training samples and 805 Instances



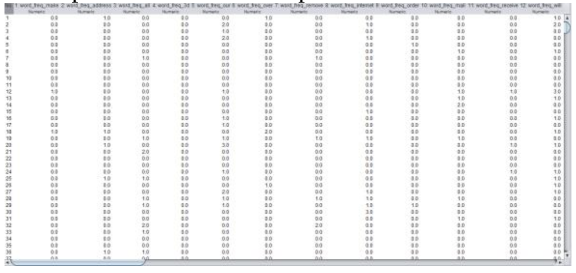
No.	1: word_freq_make	2: word_freq_address	3: word_freq_all	4: word_freq_3d	5: word_freq_our	6: word_freq_over	7: word_freq_remove	8: word_freq_internet	9: word_freq_order	10: word_freq_mail	11: word_freq_receive	12: word_freq_will
	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0
6	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	1.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	1.0
9	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0
10	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0
11	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	1.0	0.0	3.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0
14	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
15	0.0	0.0	1.0	0.0	0.0	0.0	0.0	2.0	0.0	1.0	0.0	1.0
16	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
17	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	1.0	0.0
20	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	0.0	1.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
23	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0	1.0
25	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	2.0
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	1.0
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
29	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
30	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
32	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	3.0
33	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0
35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0
36	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	2.0

TABLE III
Sample 2: 50% of the samples and 1610 Instances



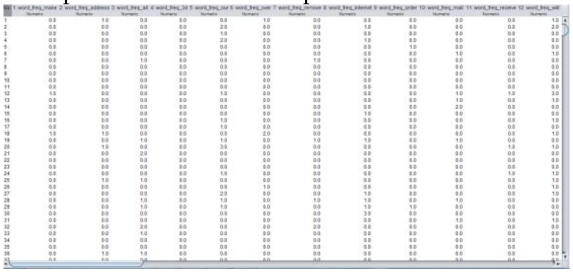
No.	1: word_freq_make	2: word_freq_address	3: word_freq_all	4: word_freq_3d	5: word_freq_our	6: word_freq_over	7: word_freq_remove	8: word_freq_internet	9: word_freq_order	10: word_freq_mail	11: word_freq_receive	12: word_freq_will
	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
7	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	3.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
17	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	1.0	1.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	1.0
19	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0
20	0.0	1.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
21	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
25	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
26	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
27	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
28	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0
29	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
32	0.0	0.0	2.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
36	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TABLE IIIII
Sample 3:75% of the samples and 2416 Instances



No.	1: word_freq_make	2: word_freq_address	3: word_freq_all	4: word_freq_3d	5: word_freq_our	6: word_freq_over	7: word_freq_remove	8: word_freq_internet	9: word_freq_order	10: word_freq_mail	11: word_freq_receive	12: word_freq_will
	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric
1	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
2	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	2.0
3	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
7	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	3.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
17	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	1.0	1.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	1.0
19	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0
20	0.0	1.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
21	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
25	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
26	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
27	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
28	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0
29	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
32	0.0	0.0	2.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
36	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TABLE IVV
Sample 4:100% of the samples and 3221 Instances



No.	1: word_freq_make	2: word_freq_address	3: word_freq_all	4: word_freq_3d	5: word_freq_our	6: word_freq_over	7: word_freq_remove	8: word_freq_internet	9: word_freq_order	10: word_freq_mail	11: word_freq_receive	12: word_freq_will
	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric	Numeric
1	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
2	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	2.0
3	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
7	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	3.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
17	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	1.0	1.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	1.0
19	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0
20	0.0	1.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
21	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
25	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
26	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
27	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
28	0.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0
29	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0
31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
32	0.0	0.0	2.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0
33	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
36	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

3.7.3 MLP Training Phase (Classification)

Once pre-processing phase is completed, the spam-based data set is ready to be trained by MLP. The aim of this phase is to train MLP based on the amount of samples of spam and non-spam (ham) messages to be able to distinguish between them. The samples are selected randomly from spam-based data sets. The weights of all input nodes are initialized to small random numbers and given to the network.

i) MLP-1 Training Implementation

MLP is a form of supervised learning for multilayer perceptron. It is most often used as training algorithm. Four different neural network structures for MLP learning are proposed in our work.

The first structure coined as *MLP-1*, consists of input layer with 46 nodes and 25% of the training spam-based dataset. There are two hidden layers in this structure - 8 nodes at the first hidden layer, 6 nodes at the second hidden layer. There are two nodes at the output layer i.e., spam and non-spam. Fig.7 demonstrates MLP-1 structure. The algorithm in section 3.3 clarifies how MLP-1 operates for all the below mentioned structure MLP-1, MLP-2, MLP-3, and MLP-4.shown in Fig.7, Fig.8, Fig.9, Fig.10.

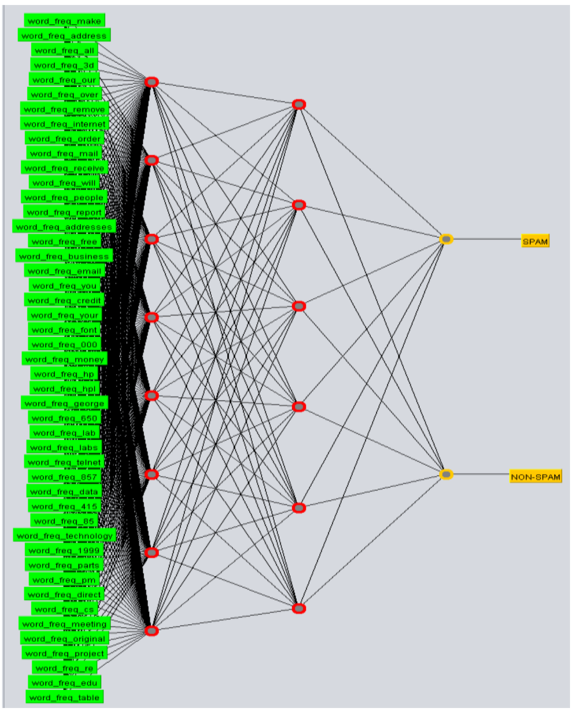


Fig. 7 Input Layer, Hidden Layers, and Output Layer of MLP-1

ii) MLP-2 Training Implementation

The second structure named *MLP-2*, consists of input layer with 46 nodes and 50% of the training spam-based dataset. We have used two hidden layers for this structure - 8 nodes at the first hidden layer, 6 nodes at the second hidden layer. There are two nodes at the output layer i.e., spam and non-spam. Fig.8 demonstrates MLP-2 structure.

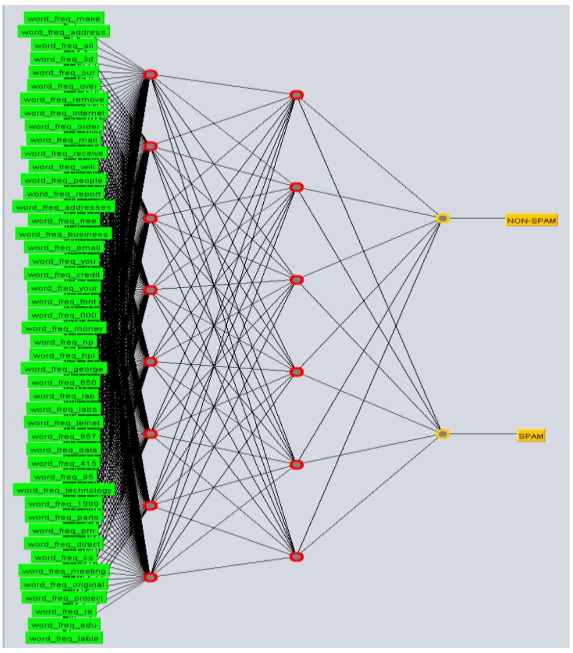


Fig. 8 Input Layer, Hidden Layers, and Output Layer of MLP-2

iii) MLP-3 Training Implementation

The third structure coined as *MLP-3,* consists of input layer with 46 nodes and 75% of the training spam-based dataset. There are two hidden layers in this structure - 8 nodes at the first hidden layer, 6 nodes at the second hidden layer. There are two nodes at the output layer i.e., spam and non-spam. Fig.9 demonstrates MLP-3 structure.

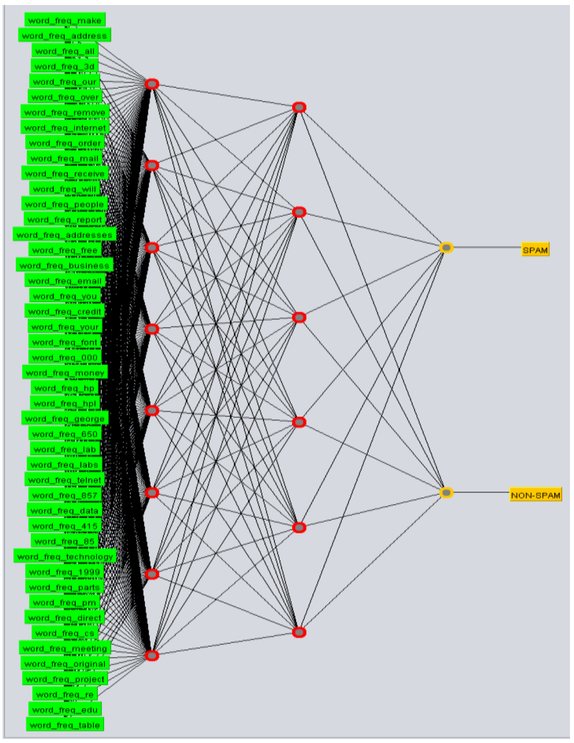


Fig. 9 Input Layer, Hidden Layers, and Output Layer of MLP-3

iv) MLP-4 Training Implementation

The fourth structure coined as *MLP-4*, consists of input layer with 46 nodes and 100% of the training spam-based dataset. There is only one hidden layer with multiple nodes. However there are two nodes at the output layer i.e., spam and non-spam. Fig.10 demonstrates MLP-4 structure.

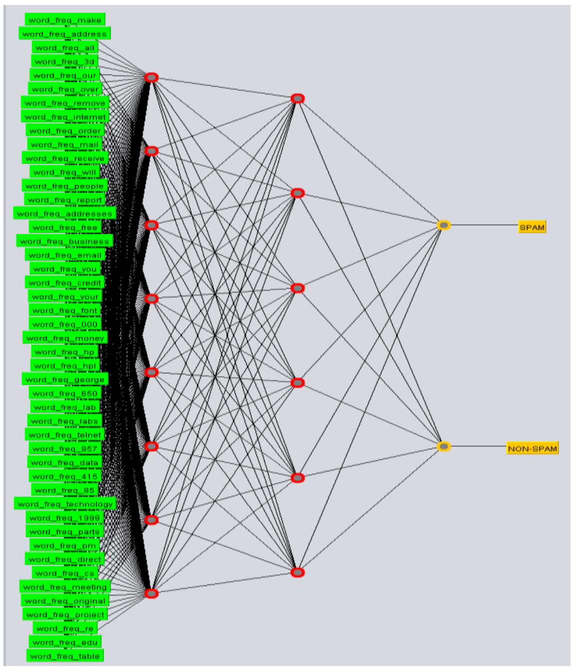


Fig. 10 Input Layer, Hidden Layers, and Output Layer of MLP-4

3.7.4 Testing Phase

When the training phase for four network structures is completed, the Neural Network (NN) models are ready to identify if the incoming e-mail is legitimate or spam. The testing phase uses these NN models to determine the type of message as a spam or an e-email. Testing is done for checking the accuracies of a model.

In this paper, 30% of the total dataset are selected for testing. The test samples should be pre-processed using PCA algorithm. When the pre-processing is completed, the test sample propagates forward the network on every in the layer. Then the weight sum of the inputs to the node is calculated, and the bias value is added to the sum, finally the activation function to the node (i.e., the desired output) is calculated. The network decides if the test sample is legitimate email or a spam depends on the desired output.

Various samples of the datasets are applied to the above NN models and classification of the spam and non-spam (ham) mails are achieved.

II. RESULT ANALYSIS

There are four networks in our model. The input for each model is a sample dataset and outputs are the desired classes. For these models, the performance metrics during training phase is demonstrated on Table 5.

TABLE V
Performance metrics of the four models during Training Phase.

Performance Metrics	MLP-25%	MLP-50%	MLP-75%	MLP-100%
Correctly classified instances	705-87.58 %	1460-90.68%	2226-92.14%	2971-92.23 %
Incorrectly classified instances	100-12.42 %	150-9.31 %	190-7.86%	250-7.77 %
Mean statistic	0.7514	0.7642	0.7701	0.6867
Mean absolute error	0.169	0.1517	0.1249	0.1361
Root mean squared error	0.2838	0.269	0.2425	0.2527
Relative absolute error	33.79%	39.03%	36.27%	58.44%
Root Relative squared error	56.76%	61.02%	58.43%	74.07%
Total number of instances	805	1610	2416	3221

In MLP-1, the correctly classified and incorrectly classified instances are 87.58% and 12.42% respectively during the training phase. In MLP-2, the correctly classified and incorrectly classified instances are 90.68% and 9.31% respectively during the training phase. In MLP-3, the correctly classified and incorrectly classified instances are 92.14% and 7.86% respectively during the training phase. In MLP-4, the correctly classified and incorrectly classified instances are 92.23% and 7.77% respectively during the training phase.

The performance metrics of the models during the testing phase is demonstrated on Table 6.

TABLE VI
Performance metrics of the four models during Testing Phase.

Performance Metrics	MLP-25%	MLP-50%	MLP-75%	MLP-100%
Correctly classified instances	309-89.83%	593-85.95%	913-88.29%	1127-81.66 %
Incorrectly classified instances	35-10.17%	97-14.05%	121-11.71%	253-18.34 %
kappa statistics	0.7887	0.6445	0.7674	0.4806
Mean absolute error	0.185	0.1933	0.1797	0.2525
Root mean squared error	0.2943	0.3049	0.3061	0.3626
Total number of instances	344	690	1034	1380

In MLP-1, the correctly classified and incorrectly classified instances are 89.83% and 10.17% respectively during the testing phase. In MLP-2, the correctly classified and incorrectly classified instances are 85.95% and 14.05% respectively during the testing phase. In MLP-3, the correctly classified and incorrectly classified instances are 88.29% and 11.71% respectively during the testing phase. In MLP-4, the correctly classified and incorrectly classified instances are 81.66% and 18.34% respectively during the testing phase.
The performance metrics of the models during the testing phase is demonstrated on Table 6.

The models can be compared in terms of various accuracy measures. The comparison of the models based on precision is shown in Fig. 11

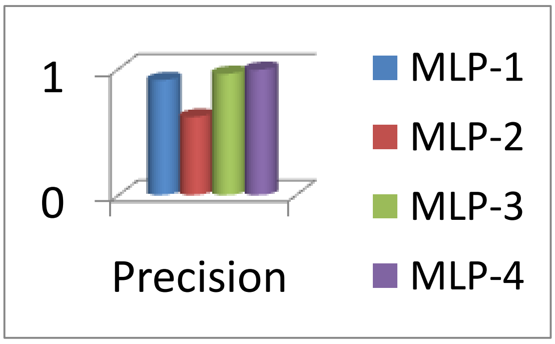


Fig.11: Comparison of MLP-1, MLP-2, MLP-3, and MLP-4 based on Precision

The precision of MLP-1 is 0.910, MLP-2 is 0.619, MLP-3 is 0.960, and MLP-4 is 0.992.
The comparison of the models based on recall is shown in Fig. 12.

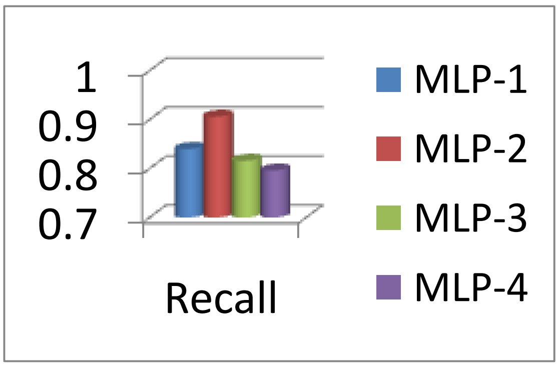


Fig.12: Comparison of MLP-1, MLP-2, MLP-3, and MLP-4 based on Recall

The precision of MLP-1 is 0.840, MLP-2 is 0.906, MLP-3 is 0.816, and MLP-4 is 0.796.
The comparison of the models based on ROC Area is shown in Fig. 13.

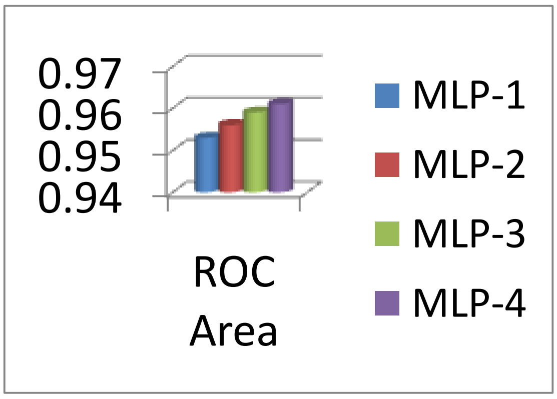


Fig.13: Comparison of MLP-1, MLP-2, MLP-3, and MLP-4 based on ROC Area

The ROC Area of MLP-1 is 0.953, MLP-2 is 0.956, MLP-3 is 0.959, and MLP-4 is 0.961.

III.CONCLUSION AND FUTURE WORK

Spam e-mails have become a big problem in the Internet world and it creates several problems for e-mail users. These spam e-mails may come from fake institutions which pretend to be authentic institutions like bank, government organizations or some others financial institutions. In this paper, we have presented several Neural Network techniques to handle the e-mail data, pre-process, and classify them. This model is beneficial to the e-mail data miners. We have tested the model with large number of instances and the outcome of the model is impressive. On an average, all of the four models have 95% of correctly classified instances. The results of the tested spam-based dataset with MLP is comparable in terms of TP Rate, FP rate, Accuracy, Precision, Recall, F-measure, MCC, ROC Area, PRC Area. From the results, it can be determined that all the models outperformed and can be successfully implemented to classify the spam e-mails.

The future work would include large number of e-mail clients. Moreover, one can modify the model to set the parameters like to change the number of hidden layers instead of trial and error settings of the nodes. Additionally, the convergence time of MLP is slow which can be improved by using appropriate optimization techniques for fast convergence. A rigorous research would be carried to detect and prohibit the spam senders.

ACKNOWLEDGMENT

The authors of this paper would like to thank the reviewers of the paper who would read this manuscript and give us valuable suggestions.

REFERENCES

[1] Christina V. et al, “A Study on Email Spam Filtering Techniques”, International Journal of Computer Application, Dec 2010, Vol. 12, pp. 7-9.

[2].Priyanka Sao, Pro. Kare Prashanthi, "E-mail Spam Classification Using Naïve Bayesian Classifier", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), June 201, Vol. 4 Issue 6., pp 2792 - 2796.

[3].Anjali Sharma, Manisha, Dr. Manisha, Dr. Rekha Jain, "Unmasking Spam in Email Messages", International Journal of Advanced Research in Computer and Communication Engineering, Feb. 2015, Vol. 4, Issue 2, pp 35 - 39

[4].Shuang Hao," A Thesis on Early Detection of Spam-related Activity", Georgia Institute of Technology, Dec 2014.

[5]."MAAWG email metrics program: The network operators perspective, report 15." http://www.maawg.org/sites/maawg/files/news/MAAWG_2011_Q1Q2Q3_Metrics_Report_15.pdf, 2011.

[6]."Email spam record activity" http://www.guardian.co.uk/technology/ 2011/jan/10/email-spam-record-activity, 2011.

[7]."Symantec Internet security threat report" http://www.symantec.com/content/en/us/enterprise/other_resources/b istr_main_report_ v18_2012_21291018.en-us.pdf, 2013.

[8].Microsoft releases new threat data on Rustock." http://blogs.microsoft. com/blog/microsoft-releases-new-threat-data-on-rustock/, 2011.

[9] "Email Statistics Report, 2015-2019", The Radicati Group, Inc. A Technology Market Research Firm Palo Alto, CA, USA, http://www.radicati.com.

[10] Sujeet More, Dr S A Kulkarni, " Data Mining With Machine Learning Applied for Email Deception", IEEE Proceedings of International Conference on Optical Imaging Sensor and Security, Coimbatore, Tamil Nadu, India, July 2-3, 2013.

[11] H.S. Hota, et al, “Tuned Artificial Neural Network Model for E-mail Data Classification with Feature Selection”, International Journal of Computer Applications, April 2013, Vol. 67, pp.20-25.

[12] Herrero, A., Snasel, et al, “Combined Classifiers with Neural Fuser for Spam Detection”. [Advances in Intelligent Systems and Computing] International Joint Conference CISIS12-ICEUTE12-SOCO12 Special Sessions, 2013, Vol.189.

[13] Idris, I, Selamat, A. et al, “A Combined Negative Selection Algorithm-Particle Swarm Optimization for an Email Spam Detection System”, Engineering Applications of Artificial Intelligence, 2015, vol.39, pp.33-44.

[14].Simon, D. “Biogeography-Based Optimization”,IEEE Transactions on Evolutionary Computation, 2008, vol.12, pp.702-713.

[15] Ali Rodan, et al, “Optimizing Feed forward Neural Networks Using Biogeography Based Optimization for E-Mail Spam Identification”, Int. Jl. of Communications, Network and System Sciences, 2016, vol. 9, pp. 19-28.

[16] Ren WangI, Amr M. Youssef , Ahmed K. Elhakeem, "On Some Feature Selection Strategies for Spam Filter Design", The 2006 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE/CCGEI),Ottawa, May 2006, pp. 2186 - 2189.

[17] Qiuming Ma, et al, “Text Spam Neural Network Classification Algorithm”, IEEE International Conference on Communications, Circuits and Systems, 2010, vol. 9, pp. 466-469.

[18] Ismaila Idris, “E-mail spam classification with ANN and Negative selection algorithms”, International Journal of Computer Science & Communication Networks,2011, Vol.1(3), pp 227-231.

[19] El-Sayed M. El-Alfy et al., “Using GMDH-based networks for improved spam detection and email feature analysis”, Applied soft computing, 2011, vol. 11, pp. 477-488.

[20] Harshal Deshmukh, “Spam Mail Detection Using Artificial Neural Network”; Imperial Journal of Interdisciplinary Research (IJIR), 2016, Vol-2, Issue-5, pp.1233-1236.

[21] Masurah Mohamad, Ali Selamat, " An Evaluation on the Efficiency of Hybrid Feature Selection in Spam Email Classification ", IEEE 2015 International Conference on Computer, Communication , and Control Technology(14 ICT 2015), Sarawak, Malaysia, April 21-23 2015, pp. 227 – 231.

[22] L. Firte, C. Lemnaru and R. Potolea, ―Spam Detection Filter Using KNN Algorithm and Resampling‖, in Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing, IEEE-2010, pp. 27-33.

[23] A. Nossier, khaled kagati, and Islam Taj-Eddin, ―Intelligent Word- Based Spam Filter Detection Using Multi Neural Networks‖, International Journal of Computer Science, March 2013, Vol. 10, Issues 2, No. 1, 1664- 0784

[24] Reena Sharma, et al, “E-mail Spam Detection Using SVM and RBF”, I.J. Modern Education and Computer Science, April 2016, vol.4, pp.57- 63.

[25] Josin Thomas, Nisha S. Raj , Vinod P. "Towards Filtering Spam Mails using Dimensionality Reduction Methods", 5th IEEE International Conference - Confluence The Next Generation Information Technology Summit - 2014, pp. 163 - 168.

[26].Mohammad-Ali Oveis-Gharan, Kaamran Raahemifar, " Multiple Classifications for Detecting Spam email by Novel Consultation Algorithm", The IEEE-2014 Canadian Conference on Electrical and Computer Engineering (IEEE-CCECE), 14-15 May 2014, Toronto, Canada, pp. 1- 5.

[27] Arram, A., Mousa, H. and Zainal, A. “Spam Detection Using Hybrid Artificial Neural Network and Genetic Algorithm”, 13th International Conference on Intelligent Systems Design and Applications (ISDA), IEEE- 2013, pp.336-340.