Artificial Neural Networks for Content-based Web Spam Detection

Renato M. Silva¹, Tiago A. Almeida², and Akebo Yamakami¹

¹School of Electrical and Computer Engineering, University of Campinas – UNICAMP, Campinas, SP, Brazil
{renatoms,akebo}@dt.fee.unicamp.br
²Department of Computer Science, Federal University of São Carlos – UFSCar, Sorocaba, SP, Brazil
talmeida@ufscar.br

Abstract—Web spam has become a big problem in the lives of Internet users, causing personal injury and economic losses. Although some approaches have been proposed to automatically detect and avoid this problem, the high speed the techniques employed by spammers are improved requires that the classifiers be more generic, efficient and highly adaptive. Despite of the fact that it is a common sense in the literature that neural based techniques have a high ability of generalization and adaptation, as far as we know there is no work that explore such method to avoid web spam. Given this scenario and to fill this important gap, this paper presents a performance evaluation of different models of artificial neural networks used to automatically classify and filter real samples of web spam based on their contents. The results indicate that some of evaluated approaches have a big potential since they are suitable to deal with the problem and clearly outperform the state-of-the-art techniques.

Keywords: web spam; spam classifier; artificial neural network; pattern recognition

1. Introduction

The web has becoming an essential tool in the lives of their users. The steady growth in volume of available information requires the use of search engines that help users to find desired data and presenting the results in an organized, quick and efficient way. However, there are several malicious methods that try to circumvent the search engines by manipulating the relevance of web pages. This deteriorates the search results, leaves users frustrated and exposes them to inappropriate and insecure content. Such technique is known as web spamming [1].

Web spam is composed by content spam and/or link spam. According to Gyongyi and Garcia-Molina [2], a very simple example of content spam is a web page with pornography and thousands of invisible keywords that have no connection with the pornographic content. Sheng et al. [3] explain that link spam is a kind of web spamming with thousands of links to web pages that are intended to promote. Such method increases the relevance of pages in search engines that rank the importance of pages using the relation of the amount of links pointing to it.

The minor problems caused by web spamming is frightfully increasing. Eiron et al. [5] have classified 100 million web pages and they found that, in average, 11 of the 20 best results were pornographic pages that achieved a high relevance through the manipulation of links. According to Gyongyi and Garcia-Molina [2], there are more than 20% of web spam in the search results. Furthermore, the same study has indicated that about 9% of the search results have at least one link from a spam page in the 10 best results, while 68% of all queries have some kind of spam in the four best results presented by the search engines [2].

According to annual reports, the amount of web spam is frightfully increasing. Eiron et al. [5] have classified 100 million web pages and they found that, in average, 11 of the 20 best results were pornographic pages that achieved a high relevance through the manipulation of links. According to Gyongyi and Garcia-Molina [2], there are more than 20% of web spam in the search results. Furthermore, the same study has indicated that about 9% of the search results have at least one link from a spam page in the 10 best results, while 68% of all queries have some kind of spam in the four best results presented by the search engines [2].

Unlike the large amount of available approaches to deal with email spams [6], [7], [8], [9], [10], [11], [12], [13], [14], there few methods in the literature to automatically detect web spam. In general, all the techniques employ one of the following strategies:

- To analyze only the relation of web links [3], [15];
- To analyze only the content of the web pages [16], [17]; or
- To extract features from both contents and links [18], [19], [20].

Among all the approaches presented in the literature, machine learning methods are the most used ones, such as ensemble selection [21], [22], clustering [20], [23], random forest [21], boosting [21], [22], support vector machine [1], [24], and decision trees [15], [20]. However, the main conclusions presented in the literature indicate that the high speed the techniques employed by spammers are improved requires that the classifiers be more generic and adaptive.

To the best of our knowledge, the artificial neural network, that is the one of the most popular and successful technique for pattern recognition, has not been evaluated for classifying web spam. To fill this important gap, this paper presents a performance evaluation of different models of artificial
neural networks used to automatically classify and filter real and public samples of web spam.

This paper is organized as follows: Section 2 introduces the background of the evaluated artificial neural networks. The experiment protocol and main results are presented in Section 3. Finally, Section 4 offers the main conclusions and guidelines for future work.

2. Artificial neural network

Artificial neural network (ANN) is a parallel and distributed method made up of simple processing units called neurons, which has computational capacity of learning and generalization. In this system, the knowledge is acquired through a process called training or learning that is stored in strength of connections between neurons, called synaptic weights [25].

A basic model of ANN has the following components: a set of synapses, an integrator, an activation function, and a bias. So, there are different models of ANN depending on the choice of each component [25].

In the following, we briefly present each model we have evaluated in this work.

2.1 Multilayer perceptron neural network

A multilayer perceptron neural network (MLP) is a perceptron-type network that has a set of sensory units composed by an input layer, one or more intermediate (hidden) layers, and an output layer of neurons. This type of network is fully connected, in other words, all neurons in any layer are connected to all neurons of the previous layer [25].

By default, MLP is a supervised learning method that uses the backpropagation algorithm which can be summarized in two stages: forward and backward.

In the forward stage, the signal propagates through the network, layer by layer, as follows:

\[ u_{ji}^l(n) = \sum_{i=0}^{m_l-1} w_{ji}^l(n)y_{i}^{l-1}(n), \]

where \( l = 0, 1, 2, ..., L \) are the indexes of network layers. So, \( l = 0 \) represents the input layer and \( l = L \) represents the output layer. On the other hand, \( y_{i}^{l-1}(n) \) is the output function relating to the neuron \( i \) in the previous layer, \( l-1 \), \( w_{ji}^l(n) \) is the synaptic weight of neuron \( j \) in layer \( l \) and \( m_l \) corresponds to the numbers of neurons in layer \( l \) for \( i = 0, y_{i}^{0-1}(n) = +1 \) and \( w_{ji}^0(n) \) represent the bias applied to neuron \( j \) in layer \( l \) [25], [26].

The output of neuron \( j \) in layer \( l \) is given by:

\[ y_{j}^l(n) = \varphi_{j}(u_{j}^l(n)), \]

where \( \varphi_{j} \) is the activation function of \( j \). Then, the error can be calculated by:

\[ e_{j}^l(n) = y_{j}^l(n) - d(n), \]

where \( d(n) \) is the desired output for an input pattern \( x(n) \).

In the backward stage, the derivation of the backpropagation algorithm is performed starting from the output layer, as follows:

\[ \delta_j^L(n) = \varphi_j'(u_j^L(n))e_j^L(n), \]

where \( \varphi_j' \) is the derivative of the activation function. For \( l = L, L-1, ..., 2 \), is calculated:

\[ \delta_j^{l-1}(n) = \varphi_j'(u_j^{l-1}(n)) \sum_{i=1}^{m_l} w_{ji}^l(n) \ast \delta_j^l(n), \]

for \( j = 0, 1, ..., m_l - 1 \) [25], [26].

According to Haykin [25], one of the most common stopping criteria for the MLPs and other types of ANNs is the amount of training iterations. The algorithm can also be stopped when the network reaches a minimum error, which can be calculated by the Mean Square Error:

\[ MSE = \frac{1}{n} e^T e, \]

where \( n \) is the number of input patterns and \( e \) an array that stores the relative error for all input patterns in the network.

2.1.1 Levenberg-Marquardt algorithm

The Levenberg-Marquardt algorithm is usually employed to optimize and accelerate the convergence of the backpropagation algorithm [27]. It is considered a second order method because it uses information about the second derivative of the error function.

Considering that the error function is given by MSE, the equation used by Gauss-Newton method to update the network weights and to minimize the value of MSE is:

\[ W_{i+1} = W_i - H^{-1}\nabla f(W). \]

The gradient \( \nabla f(W) \) can be represented by:

\[ \nabla f(W) = J^T e \]

and the Hessian matrix can be calculated by:

\[ \nabla^2 f(W) = J^T J + S, \]

where \( J \) is the Jacobian matrix

\[ J = \begin{bmatrix} \frac{\partial e_1}{\partial x_1} & \frac{\partial e_1}{\partial x_2} & \cdots & \frac{\partial e_1}{\partial x_n} \\ \frac{\partial e_2}{\partial x_1} & \frac{\partial e_2}{\partial x_2} & \cdots & \frac{\partial e_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_m}{\partial x_1} & \frac{\partial e_m}{\partial x_2} & \cdots & \frac{\partial e_m}{\partial x_n} \end{bmatrix}, \]

\( x_i \) is the \( i \)-th input pattern of the network and
where $w$ is the input pattern.
The learning rate, $\alpha$, where

\[
\nabla^2 f(W) \approx J^T J.
\]

Therefore, updating the weights in Gauss-Newton method can be done by:

\[
W_{i+1} = W_i - (J^T J)^{-1}J^T e.
\]

One limitation of the Gauss-Newton method is that a simplified Hessian matrix cannot be reversed. Thus, the Levenberg-Marquardt algorithm updates the weights by:

\[
W_{i+1} = W_i - (J^T J + \mu I)^{-1}J^T e,
\]

where $I$ is the identity matrix and $\mu$ a parameter that makes the Hessian a positive definite matrix.

More details about the Levenberg-Marquardt algorithm can be found at [26], [27], [28].

### 2.2 Kohonen’s self-organizing map

The Kohonen’s self-organizing map (SOM) is based on unsupervised competitive learning. Its main purpose is to transform an input pattern of arbitrary dimension in a one-dimensional or two-dimensional map in a topologically ordered fashion [25], [29].

The training algorithm for a SOM can be summarized in two stages: competition and cooperation [25], [29].

In the competition stage, a random input pattern $(x_j)$ is chosen, the similarity between this pattern and all the neurons of the network is calculated by the Euclidean distance, defined by:

\[
id = \arg \min_{i} \|x_j - w_i\|,
\]

where $i = 1, \ldots, k$, and the index of the neuron with lowest distance is selected.

In the cooperation stage, the synaptic weights $w_{id}$ that connect the winner neuron in the input pattern $x_i$ is updated. The weights of neurons neighboring the winner neuron are also updated by:

\[
w_i(t+1) = w_i(t) + \alpha(t)h(t)(x_i - w_i(t)),
\]

where $t$ is the number of training iterations, $w_i(t+1)$ is the new weight vector, $w_i(t)$ is the current weight vector, $\alpha$ is the learning rate, $h(t)$ is the neighborhood function and $x_i$ is the input pattern.

The neighborhood function $h(t)$ is equal to $1$ when the winner neuron is updated. This is because it determines the topological neighborhood around the winning neuron, defined by the neighborhood radius $\sigma$. The amplitude of this neighborhood function monotonically decreases as the lateral distance between the neighboring neuron and the winner neuron increases. There are several ways to calculate this neighborhood function, and one of the most common is the Gaussian function, defined by:

\[
h_{ji}(t) = \exp\left(-\frac{d_{ji}^2}{2\sigma^2(t)}\right),
\]

where $d_{ji}$ is the lateral distance between winner neuron $i$ and neuron $j$. The parameter $\sigma(t)$ defines the neighborhood radius and should be some monotonic function that decreases over the time. So, the exponential decay function

\[
\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau}\right)
\]

can be used, where $\sigma_0$ is the initial value of $\sigma$, $t$ is the current iteration number and $\tau$ is a time constant of the SOM, defined by

\[
\tau = \frac{1000}{\log \sigma_0}.
\]

The competition and cooperation stages are carried out for all the input patterns. Then, the neighborhood radius $\sigma$ and learning rate $\alpha$ are updated. This parameter should decrease with time and can be calculated by:

\[
\alpha(t) = \alpha_0 \exp\left(-\frac{t}{\tau}\right),
\]

where $\alpha_0$ is the initial value of $\alpha$, $t$ is the current iteration number and $\tau$ is a time constant of the SOM which can be calculated as presented in the cooperation stage.

### 2.3 Learning vector quantization

The learning vector quantization (LVQ) is a supervised learning technique that aims to improve the quality of the classifier decision regions, by adjusting the feature map through the use of information about the classes [25].

According to Kohonen [29], the SOM can be used to initialize the feature map by defining the set of weight vectors $w_{ij}$. The next step is to assign labels to neurons. This assignment can be made by majority vote, in other words, each neuron receives the class label in that it is more activated.

After this initial step, the LVQ algorithm can be employed. Although, the training process is similar to the SOM one, it does not use neighborly relations and updates only the winner neuron. Therefore, it is checked if the class label of the input vector $x$ is equal to the label of the winner neuron. If the labels are equal, this neuron is moved towards the input vector $x$ by:

\[
w_{id}(t+1) = w_{id}(t) + \alpha(t)(x_i - w_{id}(t)),
\]
where $\alpha$ is the learning rate, $id$ is the index of the winner neuron and $t$ is the current iteration number. However, if the label of the winner neuron is different from the label of the input pattern $x$, then it is moved away by:

$$w_{id}(t + 1) = w_{id}(t) - \alpha(t)(x_i - w_{id}(t)).$$

### 2.4 Radial basis function neural network

A radial basis function neural network (RBF), in its most basic form, has three layers. The first one is the input layer which has sensory units connecting the network to its environment. The second layer is hidden and composed by a set of neurons that use radial basis functions to group the input patterns in clusters. The third layer is the output one, which is linear and provides a network response to the activation function applied to the input layer [25].

The activation functions consists of radial basis functions whose values increase or decrease in relation to the distance for a central point. The decreasing radial basis function most common is the Gaussian, defined by:

$$h(x) = \exp\left(-\frac{(x - c)^2}{r^2}\right),$$

where $x$ is the input vector, $c$ is the center point and $r$ is the width of the function. On the other hand, the increasing radial basis function generally is represented by a multi-quadratic function, defined by [30]:

$$h(x) = \sqrt{r^2 + (x - c)^2}.$$

The procedure for training a RBF is performed in two stages. In the first one, the parameters of the basic functions related to the hidden layer are determined through some method of unsupervised training, as $K$-means.

In the second training phase, the weights of the output layer are adjusted, which corresponds to solve a linear problem [27]. According to Bishop [27], considering an input vector $x = [x_1, x_2, ..., x_n]$, the network output is calculated by:

$$y_k = \sum_{j=1}^{m} w_{kj}h_j,$$

where $x = [w_{k1}, w_{k2}, ..., w_{km}]$ are the weights, $h = [h_1, h_2, ..., h_m]$ are the radial basis functions, calculated by a function of radial basis activation.

After calculating the outputs, the weights should be updated. A formal solution to calculate the weights is given by:

$$w = h^\dagger d,$$

where $h$ is the matrix of basis functions, $h^\dagger$ represents the pseudo-inverse of $h$ and $d$ is a vector with the desired responses [27].

Consult [25], [27], [30] for more information.

### 3. Experiments and results

To give credibility to the found results and in order to make the experiment completely reproducible, all the tests were performed with the public and well-known WEBSPAM-UK2006 collection. It is composed by a set of 105,896,555 web pages hosted in 114,529 hosts in the UK domains. It is important to note that this corpus was used in Web Spam Challenge Track I and II, that are the most known competitions of web spam detection techniques.

In our experiments, we have followed the same competition guidelines. In this way, we have used a set of 8,487 feature vectors employed to discriminate the hosts as spam or ham. This kind of information was provided by the organizers. Each feature vector is composed by 96 features proposed by Castilho et al. [20], where 24 features were extracted from the host home page, other 24 ones from the page with the highest PageRank, 24 ones from the mean of these values for all pages, and the remaining 24 ones from the variance of all web pages. Furthermore, each host is labeled as spam, not spam (ham) or indefinite. Thus, the vectors related to hosts that are labeled as undefined were discarded.

### 3.1 Protocol

We evaluated the following well-known artificial neural networks (ANNs) algorithms to automatically detect web spam based on its content: multilayer perceptron (MLP) trained with the gradient descent and Levenberg-Marquardt methods, Kohonen’s self-organizing map (SOM) with learning vector quantization (LVQ) and radial basis function neural network (RBF).

We have implemented all the ANNs with a single hidden layer and with one neuron in the output layer. In addition, we have employed a linear activation function for the neuron of output layer and an hyperbolic tangent activation function for the neurons of the intermediate layer. Thus, we have initialized the weights and biases with random values between $+1$ and $-1$, and normalized the data to this interval.

The desired output for all networks is $-1$ (ham) or $+1$ (spam). So, for RBFs and MLPs, if the output is greater than or equal to 0, the host is considered spam, otherwise it is considered ham. The same not happens with SOMs since the data receive the same label of the neuron that represents them.

Regarding the parameters, in all simulations, we have employed the following stopping criteria: maximum number of iterations be greater than a threshold $\theta$, the mean square error (MSE) of the training set be smaller than a threshold $\gamma$ or when the MSE of the validation set increases (checked every 10 iterations).

The parameters used for each ANN model were empirically calibrated and are the following:

- Multilayer perceptron with gradient descent method:
  - $\theta = 10,000$
  - $\gamma = 0.001$
  - Step learning $\alpha = 0.005$
  - Number of neurons in the hidden layer: 100
- Multilayer perceptron with Levenberg-Marquardt method:
  - $\theta = 500$
  - $\gamma = 0.001$
  - Step learning $\alpha = 0.001$
  - Number of neurons in the hidden layer: 50
- Kohonen’s self-organizing map with learning vector quantization:
  - Competition stage:
    - One-dimensional neighborhood function with initial radius $\sigma = 4$
  - Cooperation stage:
    - $\theta = 2,000$
    - Step learning $\alpha = 0.01$
    - Number of neurons in the hidden layer: 120
- Radial basis function neural network:
  - Number of neurons in the hidden layer: 10

Note that, for the simulations using the RBFs, we have not employed any stopping criteria because the training method is not iterative, as pointed out in Section 2.4.

To address the algorithms performance, we divided each simulation in 10 tests and calculated the arithmetic mean and standard deviation of the following well-known measures: accuracy rate (Acc%), spam recall rate (Rcl%), specificity (Spc%), spam precision rate (Pcs%), and F-measure (FM).

In each test, we have randomly selected 80% of the samples of each class to be presented to the algorithms in the training stage and the remaining ones were separated for testing.

### 3.2 Results

In this section, we report the main results of our evaluation. Table 1 presents the performance achieved by each artificial neural network. Bold values indicate the highest score.

Table 1: Results achieved by each evaluated artificial neural network for WEBSPAM-UK2006 dataset.

<table>
<thead>
<tr>
<th></th>
<th>Gradient</th>
<th>Levenberg</th>
<th>RBF</th>
<th>SOM + LVQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc</td>
<td>Mean</td>
<td>Mean</td>
<td>Mean</td>
<td>Mean</td>
</tr>
<tr>
<td>Rcl</td>
<td>86.0±0.1</td>
<td>88.6±1.4</td>
<td>79.7±0.6</td>
<td>80.6±0.8</td>
</tr>
<tr>
<td>Spc</td>
<td>92.0±0.6</td>
<td>94.2±1.2</td>
<td>95.8±0.7</td>
<td>96.3±0.6</td>
</tr>
<tr>
<td>Pcs</td>
<td>77.5±2.7</td>
<td>77.6±4.6</td>
<td>65.8±3.3</td>
<td>70.4±4.0</td>
</tr>
<tr>
<td>FM</td>
<td>0.656±0.039</td>
<td>0.731±0.032</td>
<td>0.379±0.030</td>
<td>0.412±0.030</td>
</tr>
</tbody>
</table>

According to the results, it is clear that the MLP trained with Levenberg-Marquardt method achieved the best performance. On the other hand, the Kohonen’s self-organizing map and radial basis function neural network acquired the worst performances with a very low spam recall rate. Furthermore, in all results we noted the contrast between the high specificity rate and a not too high spam recall rate, which indicates that the classifiers have more successful to identify ham hosts than spam ones.

Based on these first results, we considered the hypothesis that there are redundancies in the feature vectors that could be affecting the classifiers performance. In this way, we employed the principal component analysis (PCA) [25] to analyze and reduce the dimensionality of the feature space. Using the PCA, we observed that with only 23 dimensions is possible to represent about 99% of the information presented in the original 96 feature vectors. According to the analysis, the most relevant information are in the first 21 dimensions. Therefore, it is possible to conclude that just the attributes extracted from the host home page have enough information to characterize the host as spam or ham.

After reducing the feature vectors from 96 to 23 dimensions, we performed a new simulation. The found results are presented in Table 2.

Table 2: Results achieved by each evaluated artificial neural network for WEBSPAM-UK2006 dataset after a step of dimensionality reduction using principal component analysis.

<table>
<thead>
<tr>
<th></th>
<th>Gradient</th>
<th>Levenberg</th>
<th>RBF</th>
<th>SOM + LVQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc</td>
<td>Mean</td>
<td>Mean</td>
<td>Mean</td>
<td>Mean</td>
</tr>
<tr>
<td>Rcl</td>
<td>38.8±2.4</td>
<td>55.5±1.4</td>
<td>2.3±1.3</td>
<td>3.1±0.8</td>
</tr>
<tr>
<td>Spc</td>
<td>95.1±0.6</td>
<td>94.8±1.2</td>
<td>99.5±0.3</td>
<td>99.8±0.1</td>
</tr>
<tr>
<td>Pcs</td>
<td>71.1±2.8</td>
<td>77.0±4.0</td>
<td>88.6±19.8</td>
<td>88.3±6.5</td>
</tr>
<tr>
<td>FM</td>
<td>0.502±0.024</td>
<td>0.644±0.027</td>
<td>0.047±0.024</td>
<td>0.060±0.014</td>
</tr>
</tbody>
</table>

Based on the found results we can see that, although PCA indicates that the new feature vectors represent more than 99% of the information of the original data, reducing the dimensionality of the feature space clearly affected the classifiers performance for all evaluated techniques, mainly for Kohonen’s self-organizing map and radial basis function neural network which achieved spam recall rate lower than 5%. Therefore, it is conclusive that in this scenario reducing the dimensionality of the feature space is not indicated.

Analyzing the results achieved in all experiments, we also suspected that the difference between the achieved high specificity rates and low spam recall rates for all evaluated techniques could be explained by the large difference between the number of ham over the amount of spam samples. This could be doing the classifiers more specialized to identify ham hosts. Thus, to test this hypothesis, we decided to present to the classifiers the same number of data in the two classes during the training process. In this way, 1,978 samples of each class were randomly selected to be trained.
With this new setup, we performed a new simulation using again all the feature vectors. Table 3 presents the classification results.

Table 3: Results achieved by each evaluated artificial neural network for WEBSPAM-UK2006 dataset using classes of equal size in the training stage.

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>Pcs</th>
<th>Rcl</th>
<th>FM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient</td>
<td>Mean</td>
<td>Mean</td>
<td>±0.016</td>
</tr>
<tr>
<td>Levenberg</td>
<td>84.7</td>
<td>87.6</td>
<td>±0.15</td>
</tr>
<tr>
<td>RBF</td>
<td>82.9</td>
<td>86.5</td>
<td>±0.20</td>
</tr>
<tr>
<td>SOM + LVQ</td>
<td>86.4</td>
<td>88.8</td>
<td>±0.19</td>
</tr>
<tr>
<td>MLP + Gradient</td>
<td>86.1</td>
<td>89.1</td>
<td>±0.24</td>
</tr>
<tr>
<td>MLP + Levenberg</td>
<td>86.1</td>
<td>89.1</td>
<td>±0.24</td>
</tr>
<tr>
<td>RBF + SOM + LVQ</td>
<td>86.4</td>
<td>89.1</td>
<td>±0.24</td>
</tr>
</tbody>
</table>

The comparison indicates that the MLP with Levenberg-Marquardt method is very competitive and suitable to deal with the problem. Note that, although the Bagging algorithm presented by Ntoulas et al. [16] achieved higher spam recall rate, the spam precision rate is lower and, consequently, the specificity rate is also lower than those one achieved by MLP. Furthermore, the results indicate that the MLP neural networks outperformed established methods as the decision trees proposed by Castilho et al. [20] and SVM presented by Svore et al. [1].

4. Conclusions and future work

In this paper, we have presented a performance evaluation of different models of artificial neural networks used to automatically classify real samples of web spam based on their contents.

The results indicate that the multilayer perceptron neural network trained with the Levenberg-Marquardt method is the best evaluated model. Such a method also outperformed established techniques available in the literature such as decision trees [20] and support vector machine [1].

Furthermore, since the data is unbalanced, the results also indicate that all the evaluated techniques are superior when trained with the same amount of samples of each class. It is because the models tend to converge to the benefit of the class with the largest number of representatives, which increases the rate of false positives or false negatives.

We have also observed that the Kohonen’s self-organizing map and radial basis function neural network were inferior than the multilayer perceptron neural networks in all simulations. So, we can conclude that such a models are not indicated to classify web spam.

Finally, an analysis on the feature vectors originally proposed by Castilho et al. [20] indicate that they have many redundancies, since with the principal component analysis is possible to reduce them from 96 to 23 dimensions that represents about 99% of the information presented in the original feature set. However, using the reduced feature vectors clearly hurts the classifiers accuracy.

For future work, we intend to propose new features to enhance the classifiers prediction, and hence increase its ability to discriminate the hosts as spam or ham. In this way, we aim to use the relation of links presented in the web pages.

References


