An Analysis of Machine Learning Methods for Spam Host Detection

Renato M. Silva, Akebo Yamakami  
School of Electrical and Computer Engineering  
University of Campinas – UNICAMP  
Campinas, São Paulo, Brazil  
{renatoms, akebo}@dt.fee.unicamp.br

Tiago A. Almeida  
Department of Computer Science  
Federal University of São Carlos – UFSCar  
Sorocaba, São Paulo, Brazil  
talmeida@ufscar.br

Abstract—The web is becoming an increasingly important source of entertainment, communication, research, news and trade. In this way, the web sites compete to attract the attention of users and many of them achieve visibility through malicious strategies that try to circumvent the search engines. Such sites are known as web spam and they are generally responsible for personal injury and economic losses. Given this scenario, this paper presents a comprehensive performance evaluation of several established machine learning techniques used to automatically detect and filter hosts that disseminate web spam. Our experiments were diligently designed to ensure statistically sounds results and they indicate that bagging of decision trees, multilayer perceptron neural networks, random forest and adaptive boosting of decision trees are promising in the task of web spam classification and, hence, they can be used as a good baseline for further comparison.

Keywords—spamdexing; web spam; spam host; classification.

I. INTRODUCTION

Search engines are powerful allies of the Internet users to find information and therefore are responsible for a significant percentage of accesses to most Web sites. Thus, to succeed, it is important that the sites increase their relevance ranking in search engines. In this way, they are better positioned in search results when users consult terms related to their contents. To achieve this purpose, strategies, known as search engine optimization (SEO), should be used [1].

There are many ethical SEO strategies, however, Ledford [1] states that is required a lot of time and hard work to learn the most successful ones. All elements of the Web site should be created to maximize your search engine rankings. But, as mentioned in the SEO guide published by Google[1], it should be considered that the ultimate consumers of the Web site are the users, not search engines. It is important that the site offers relevant content, navigation facilities and other features that benefit your visitors. The problem is that several sites prefer to invest in unethical SEO techniques and circumvent the search engines to achieve relevance without merit. The most affected are the users that when make their queries in search engines, receive unexpected and irrelevant answers, often infected with malicious content. Such techniques are known as web spamming or spamdexing [2], and according to Gyongyi and Garcia-Molina [3], they can be divided into two main categories: content spam and link spam. The first one is related to the creation of pages with irrelevant keywords and the second one corresponds to the addition of links to the Web page that is intended to promote.

The main reason that contribute to the growth of web spam is the economic incentive. The spammers create Web pages with irrelevant content, increase their ranking in search engines through spamming techniques and advertise products. So, they get some money for each click of visitors [3].

Recent studies indicate that the amount of web spam is dramatically increasing. According to John et. al [4], 36% of the results generated by the most popular search engines contain malicious URLs. A report produced by Websense\(^2\) shows that 22.4% of the search results about entertainment present malicious links. Furthermore, a report published by McAfee\(^3\) informs that 49% of the most popular search terms return some malicious site in the top 100 search results. The same study found that 1.2% of the queries return links of the malicious sites in the top 100 results.

Given this scenario, this paper presents a comprehensive performance evaluation of several well-known machine learning techniques employed to automatically detect spam hosts in order to provide good baseline results for further comparison.

This paper is organized as follows: in Section II we briefly describe the related papers available in the literature. Section III presents the basic concepts of the methods evaluated in this paper. In Section IV we present the database and settings that we used in the experiments. Section V presents the main results. Finally, Section VI describes the conclusions and guidelines for future work.

II. RELATED WORK

Some papers that address the problem of web spamming are found in the literature.

Silva et al. [5], [6], [7] evaluated different neural-based approaches for web spam detection. They also analyzed how different sets of features impact on the classifiers accuracy.

Largillier and Peyronnet [8] presented several node aggregation methods for the demotion of the effects of web spamming on the PageRank algorithm [1].

Liu et al. [9] proposed a number of user-behavior features for separating spam pages from normal pages. They also presented a framework that combines machine-learning techniques assisted by user behavior to detect spam pages.

Rungswang et al. [10] proposed an ant colony optimization algorithm that exploits both content and link features for spam host detection.

Castilho et al. [11] proposed content-based features that were used in several other relevant works and in important events. In addition, they presented a spam detection system that combines content-based and link-based features, and uses the topology of the Web graph by exploiting the link dependencies among the Web pages.

Svore et al. [2] presented a method for detecting web spam that uses content-based features and the rank-time.

Gyongyi and Garcia-Molina [3] described several web spamming techniques used by spammers.

Nevertheless, as each mentioned paper used a different experiment setup, it is still unclear what are the best machine learning methods available in the literature that can be used to automatically detect spam hosts.

III. METHODS

This section presents concepts regarding the following well-known methods that we evaluated in this paper: multilayer perceptron neural networks, support vector machines, methods based on trees, such as decision trees, random forest, bagging and adaptive boosting of trees, and k-nearest neighbor. Such methods were chosen because most of them have been evaluated and presented as the best machine learning and data mining techniques currently available [12].

A. Multilayer perceptron neural network (MLP)

A multilayer perceptron neural network 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 [13]. By default, MLP is a supervised learning method that uses the backpropagation algorithm which can be summarized in two stages: forward and backward [14].

In the forward stage, the signal propagates through the network, layer by layer, as follows: \( u_{j}^{l}(n) = \sum_{i=0}^{m^{l-1}} u_{ji}^{l}(n) y_{i}^{l-1}(n) \), 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, \( l - 1, u_{ji}^{l}(n) \) is the synaptic weight of neuron \( j \) in layer \( l \) and \( m^{l} \) corresponds to the number of neurons in layer \( l \). For \( i = 0 \), \( y_{0}^{l-1}(n) = +1 \) and \( u_{j0}^{l}(n) \) represent the bias applied to neuron \( j \) in layer \( l \) [13].

The output of neuron \( j \) in layer \( l \) is given by \( y_{j}^{l}(n) = \phi_{j}(u_{j}^{l}(n)) \), where \( \phi_{j} \) 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 backward stage, the derivation of the backpropagation algorithm is performed starting from the output layer, as follows: \( \delta_{j}^{L}(n) = \phi_{j}^{'}(u_{j}^{L}(n)) e_{j}^{L}(n) \), where \( \phi_{j}^{'} \) is the derivative of the activation function. For \( l = L, L - 1, ..., 2 \), is calculated: \( \delta_{j}^{l-1}(n) = \phi_{j}^{'}(u_{j}^{l-1}(n)) \sum_{i=1}^{m^{l}} u_{ji}^{l}(n) * \delta_{j}^{l}(n) \), for \( j = 0, 1, ..., m^{l} - 1 \).


1) Levenberg-Marquardt algorithm: the Levenberg-Marquardt algorithm is usually employed to optimize and accelerate the convergence of the backpropagation algorithm [14]. It is considered a second order method because it uses information about the second derivative of the error function. Details can be found in Bishop [14] and Hagan and Menhaj [15].

B. Support vector machines (SVM)

Support vector machines (SVM) [16], [17], [18], [19], [20] is a machine learning method that can be used for pattern classification, regression and others learning tasks [13], [21]. This method was conceptually implemented following the idea that input vectors are non-linearly mapped to a high dimension feature space. In this feature space is constructed a linear decision surface which separates the classes of the input patterns.

One of the main elements that the SVM uses to separate the patterns of distinct classes is a kernel function. Through it, the SVM constructs a decision surface nonlinear in the input space, but linear in the features space [13]. Table I presents the most popular SVM kernel functions, where \( \gamma \), \( r \) and \( d \) are parameters that must be set by the user.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Formula</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>( k(x_i, x_j) = x_i^T x_j )</td>
<td>-</td>
</tr>
<tr>
<td>RBF</td>
<td>( k(x_i, x_j) = \exp(-\frac{\gamma}{2\sigma^2}</td>
<td></td>
</tr>
<tr>
<td>Polynomial</td>
<td>( k(x_i, x_j) = (\gamma x_i^T x_j + r)^\alpha ), ( \gamma &gt; 0 )</td>
<td>-</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>( k(x_i, x_j) = tanh(\gamma x_i^T x_j + r) )</td>
<td>-</td>
</tr>
</tbody>
</table>

To assist the choice of the SVM parameters, Hsu et al. [22] recommend the employment of a grid search. For instance, considering the SVM with RBF kernel, in which is necessary to define the regularization parameter \( C \) and \( \gamma \), the authors suggest that the grid search could be used to evaluate exponential sequences: \( C = 2^{-5}, 2^{-4}, 2^{-3}, ..., 2^{15} \) and \( \gamma = 2^{-15}, 2^{-14}, ..., 2^{3} \).
C. Decision trees (C4.5)

The C4.5 [23] is one of the most classical decision tree algorithms and uses both categorical and continuous attributes. It uses a divide-and-conquer approach to increase the predictive ability of the decision trees. Thus, a problem is divided in several sub-problems, by creating sub-trees in the path between the root and the leaves of the decision tree.

D. Random forest

A random forest [24] is a combination of decision trees in which each tree depends on the values of a random vector sampled independently and equally distributed for all trees in the forest. In this method, after generating a large number of trees, each one votes for one class of the problem. Then, the class with more number of votes is chosen.

E. K-nearest neighbors (IBK)

The IBK is an instance-based learning algorithm (IBL) [25]. Such method, derived from the k-nearest neighbors (KNN) classifier, is a non-incremental algorithm and aims to keep a perfect consistency with the initial training set. On the other hand, the IBL algorithm is incremental and one of its goals is maximizing classification accuracy on new instances [25].

As well as in the KNN, in IBK, the classification generated for the sample i is influenced by the outcome of the classification of its k-nearest neighbors, because similar samples often have similar classifications [25], [26].

F. Adaptive boosting (AdaBoost)

The adaptive boosting [27] is a boosting algorithm widely used in pattern classification problems. In general, as any boosting method, it makes a combination of classifiers. However, it has some properties that make it more practical and easier to implement than the boosting algorithms that preceded it. One of these properties is that it does not require any prior knowledge of the predictions achieved by weak classifiers. Instead, it adapts to the bad predictions and generates a weighted majority hypothesis in which the weight of the each prediction achieved by weak classifiers it is a function of its prediction.

G. Bagging

The bagging [28] is a method for generating multiple versions of a classifier that are combined to achieve an aggregate classifier. The classification process is similar to the boosting methods, but according to Witten and Frank [26], unlike what occurs in the second one, in the bagging, the different models of classifiers get the same weight in the generation of a prediction.

H. LogitBoost

The LogitBoost method [29] is a statistical version of the boosting method and, according to Witten and Frank [26], it has some similarities with Adaboost, but it optimizes the likelihood of a class, while the Adaboost optimizes an exponential cost function. Friedman et. al [29] defines this method as an algorithm for assembly of additive logistic regression models.

IV. DATABASE AND EXPERIMENT SETTINGS

To give credibility to the found results and in order to make the experiments reproducible, all the tests were performed with the large, public and well-known WEbspam-UK2006 collection4. It is composed by 77.9 million web pages hosted in 11,000 hosts in the UK domains. It is important to note that this corpus was used in Web Spam Challenge5 I and II, that are the most known competitions of web spam detection techniques.

In our experiment, we followed the same competition guidelines. In this way, we used three sets of 8,487 feature vectors employed to discriminate the hosts as spam or ham. Each set is composed by 6,509 hosts labeled as ham and 1,978 labeled as spam. The organizers provided three sets of features: the first one composed by 96 content-based features [11], the second one composed by 41 link-based features [30] and the third one composed by 138 transformed link-based features [11], which are the simple combination or logarithm operation of the link-based features.

To address the algorithms performance, we used a random sub-sampling validation, which is also known as Monte Carlo cross-validation [31]. Such method provides more freedom to define the size of training and testing subsets. Unlike the traditional k-fold cross-validation, the random sub-sampling validation allows to do as many repetitions were desired, using any percentage of data for training and testing. In this way, we divided each simulation in 10 tests and calculated the arithmetic mean and standard deviation of the following well-known measures: spam recall, spam precision and F-measure [26], [32], [33]. In each test, we 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.

A. Settings

In the following, we describe the main parameters used for each classifier evaluated in this work.

1) MLP: we evaluated the following well-known artificial neural networks; multilayer perceptron trained with the gradient descent method (MLP-GD) and multilayer perceptron trained with Levenberg-Marquardt method (MLP-LM).

4Yahoo! Research: “Web Spam Collections”. Available at http://barcelona.research.yahoo.net/webspam/datasets/
5Web Spam Challenge: http://webspam.lip6.fr/
We implemented all the MLPs 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 a hyperbolic tangent activation function for the neurons of the intermediate layer. Thus, we normalized the data for the interval $[-1, 1]$. Furthermore, one of the stopping criteria that we have used was the increase of the validation set error (checked every 10 iterations). The others parameters were empirically calibrated and they are presented in Table II.

Table II: Parameters used in the multilayer perceptron artificial neural networks.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MLP-GD</th>
<th>MLP-LM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max number of iterations</td>
<td>10000</td>
<td>500</td>
</tr>
<tr>
<td>Min limit for the mean square error (MSE)</td>
<td>0.001</td>
<td>0.001</td>
</tr>
<tr>
<td>Step learning</td>
<td>0.005</td>
<td>0.001</td>
</tr>
<tr>
<td>Neurons in the hidden layer</td>
<td>100</td>
<td>50</td>
</tr>
</tbody>
</table>

2) SVM: we implemented the SVM using the LIBSVM library [21] available for the MATLAB tool. We performed simulations with the linear, RBF, sigmoid and polynomial kernel functions and we used grid search to define the parameters. However, in the SVMs with polynomial and sigmoid kernel, that have a larger number of parameters, we performed the grid search only on the parameters $C$ and $\gamma$, due to excessive computational cost. In this case, we set the default values of the LIBSVM for the others parameters.

We performed the grid search using the random subsampling validation with 80% of the samples for training and 20% for test. Then, we chosen the best parameters – those that achieved the highest f-measures, and we used them to perform the experiments with SVM. In such experiments we evaluated all types of kernels. However, due to space limit, we decided to present only the results achieved by SVM with RBF kernel since it achieved the best performance. Table III presents the parameters used in the simulations with the SVM.

Table III: Parameters achieved by grid search and used by SVM with RBF kernel according to the set of features and balance the classes.

<table>
<thead>
<tr>
<th></th>
<th>Content</th>
<th>Links</th>
<th>Transf. links</th>
<th>Content+links</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C_{\gamma}$</td>
<td>$C_{\gamma}$</td>
<td>$C_{\gamma}$</td>
<td>$C_{\gamma}$</td>
</tr>
<tr>
<td>Balanced</td>
<td>$2_{15}^{14}$</td>
<td>$2_{15}^{12}$</td>
<td>$2_{15}^{12}$</td>
<td>$2_{15}^{10}$</td>
</tr>
<tr>
<td>Unbalanced</td>
<td>$2_{15}^{12}$</td>
<td>$2_{15}^{10}$</td>
<td>$2_{15}^{10}$</td>
<td>$2_{15}^{8}$</td>
</tr>
</tbody>
</table>

3) Remaining methods: we implemented the remaining classifiers using the WEKA tool [34]. The AdaBoost and bagging algorithms were trained with 100 iterations and both of them employ aggregation of multiple versions of C4.5 method. For all the other approaches we used the default parameters.

V. Results

For each evaluated method and feature set, we have performed a simulation using unbalanced classes, as originally provided in the web spam competition, with 6,509 (76.6%) samples of ham hosts and 1,978 (23.4%) samples of spam ones. Moreover, to evaluate if the unbalance of the data impacts on the classifiers accuracy, we have also performed simulations using the same number of samples in each class. For this, both classes were composed by 1,978 representatives, randomly selected.

Table IV and Table V present the results achieved by each classifier exploring the content-based features, links, transformed links, and the combination of links and content using unbalanced and balanced data, respectively. In each table, the results are sorted by F-measure for each feature set and bold values indicate the highest score for each performance measure. Further, values preceded by the symbol "*" indicate the highest score considering all the simulations.

In general, the evaluated classifiers achieved good results, since they have detected a significant amount of spam hosts with high precision, regardless of the used feature set. Furthermore, the results indicate that bagging of trees, in general, presented the best results. In average, it was able to detect 83.5% of the spam hosts with a precision rate of 82.9%. On the other hand, the SVM classifier, in general, achieved the worst performance.

It is important to note that, since the data we used in our experiment 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 be biased to the benefit of the class with the largest amount of samples.

If we compare the used set of features, we can observe that the best average result for all classifiers was achieved when transformed link-based features were employed. However, the highest F-measure was achieved by AdaBoost with the combination of content-based features and relation of links (Table V). Therefore, we can conclude that the most appropriate feature set varies according to the classifiers.

In order to support our claims, we also performed a statistical analysis of the results (Table VI). For that, we ranked the methods by F-measure using the Wilcoxon rank-sum test [35] and 95% of confidence interval. Such method is a statistical hypothesis test which is also sometimes called the Mann-Whitney test [35]. The methods that are at the same level in the table have statistically equivalent results. Note that, in our simulations, bagging of trees is statistically superior than other evaluated approaches for unbalanced data but statistically equal to AdaBoost, MLPs and random forest for balanced data. Therefore, it is safe to conclude that such methods can be used as a good baseline for further comparison. On the other hand, SVM is statistically inferior than all evaluated approaches independent on the balance.
and feature set.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a comprehensive performance evaluation of different established machine learning algorithms used to automatically identify spam hosts based on features extracted from their web pages. For this, we employed a real, public, non-encoded and large database composed by samples represented by content-based features, link-based features, the combination of both and transformed link-based features.

The results indicated that, in general, all evaluated techniques achieved good performance, regardless of the used attribute. This indicates that, besides to be efficient, they have good capability of generalization.

Furthermore, since the data we used in our experiment is unbalanced, the results also indicated 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 be biased to the benefit of the class with the largest amount of samples.

Among all evaluated classifiers, the aggregation techniques, such as bagging and boosting of trees, statistically achieved the best performances, demonstrating to be promising to identify spam hosts and consequently they are recommended to be used as a good baseline for further comparison.

For future work, we intend to adapt the most promising methods in order to optimize their performances and we aim to study and propose new set of features in order to increase the capacity of the classifiers prediction.
Table VI: Statistical analysis of the results using the Wilcoxon rank-sum test.

<table>
<thead>
<tr>
<th>Level</th>
<th>Unbalanced classes</th>
<th>Balanced classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bagging</td>
<td>bagging, AdaBoost, MLP-LM, MLP-GD and random forest</td>
</tr>
<tr>
<td>2</td>
<td>AdaBoost and MLP-LM</td>
<td>C4.5, IBK and LogitBoost</td>
</tr>
<tr>
<td>3</td>
<td>MLP-LM and MLP-GD</td>
<td>SVM</td>
</tr>
<tr>
<td>4</td>
<td>Random forest</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>C4.5 and LogitBoost</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>LogitBoost and IBK</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SVM</td>
<td></td>
</tr>
</tbody>
</table>

ACKNOWLEDGMENT

The authors would like to thank the financial support of Brazilian agencies FAPESP, Capes and CNPq.

REFERENCES


