Context-sensitive and keyword density based supervised machine learning techniques for malicious webpage detection

Betul Altay[a], Tansel Dokeroglu[b,*], Ahmet Cosar[c]

[a]*HAVELSAN, Hava Elektronik Sanayi, Ankara, TURKEY*
[b]*Computer Engineering Department, THK University, Ankara, TURKEY*
[c]*Computer Engineering Department, Middle East Technical University, Ankara, TURKEY*

Abstract

Conventional malicious webpage detection methods use blacklists in order to decide whether a webpage is malicious or not. The blacklists are generally maintained by third party organizations. However, keeping a list of all malicious web sites and updating this list regularly is not an easy task for the frequently changing and rapidly growing number of webpages on the web. In this study, we propose a novel context-sensitive and keyword density based method for the classification of webpages by using three supervised machine learning techniques, Support Vector Machine (SVM), Maximum Entropy (MaxEnt), and Extreme Learning Machine (ELM). Features (words) of webpages are obtained from HTML contents and information is extracted by using feature extraction methods: existence of words, keyword frequencies, and keyword density techniques. The performance of proposed machine learning models are evaluated by using a benchmark data set which consists of one hundred thousand webpages. Experimental results show that the proposed method can detect malicious webpages with an accuracy of 98.24%, which is a significant improvement compared to state-of-the-art approaches.

Keywords: Malicious, Webpage, Classification, SVM, Maximum Entropy, Extreme Learning Machines, Keyword Density

1. Introduction

The Internet usage has become essential for our common daily activities such as shopping, education, and entertainment. In accordance with the statistics of International Telecommunication Union (ITU), the number of individuals using the Internet was over three billion throughout the world in 2015 [1]. Unfortunately, the huge number of users of the Internet and its facilities may cause great danger in security because of cyber criminal activities. Webpages used to have static HTML content but now they include technologies that interact with users. This situation may cause significant risks in online security of the computers.

Webpages containing threats for users are called malicious webpages and the most important security threats included in these pages are Phishing and Cross Site Scripting (XSS). Phishing (Fishing), is an attempt to obtain personal information of the Internet users by making use of social engineering. Malicious webpages that use phishing try to steal user names, passwords, e-mail addresses, phone numbers, photos, social security numbers and even credit card details of victims [2]. The operation mechanism of phishing is based on impersonating another user and/or official web site. For example, webpages may include links to web sites different from where the user thinks he/she is reaching. These links can download a harmful executable to victims' computers or can open another malicious or undesirable document. The phishing webpages may request personal information and users can be easily deceived (see Figure 1). The users may be exposed to fake advertising and counterfeit products purchases because of the fake webpages. If the user buys a product, the sold item can be an imitation, an illegal one or even an empty box.

*Corresponding author
Email address: tdokeroglu@thk.edu.tr (Tansel Dokeroglu)

The technique, XSS, gives opportunities to attackers to inject malicious code into webpages. After the injection, the victim's browser/computer becomes vulnerable to further attacks and/or sensitive information leaks [3]. It has become a major issue especially after the advances in webpage design by using the scripting technologies. The drive-by-download techniques make the web service development easier, powerful and flexible [4]. However, the power and flexibility in recent webpages provides a new tool that can be misused by attackers. A recent study shows that there are too many malicious webpages in search results [5]. Because of their importance , variety, and intensity, filtering malicious webpages has become an important research topic.

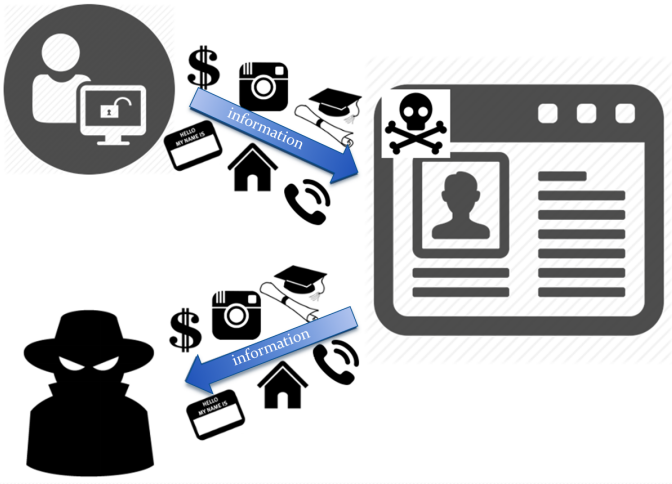

Figure 1: Flow of Phishing process [6]

In order to handle this problem, various solutions have been proposed. The most widely known technique is the blacklist approach. Browsers and security tools have these lists that contain malicious web domains and URLs. If a requested URL is found on the blacklist of Google Safe Browsing, browser does not accept the page. However, blacklist approach has deficiencies; (1) the lists include only the crawled webpages, (2) crawlers can not reach intranets, (3) crawled pages may be hacked after the crawling, (4) they need a malicious webpage detection mechanism or human assistance during the production of the list [7]. The second method is the creation of honeypots with Virtual Machines (VM). By using VM environments, visiting a webpage is simulated and its effects may be observed. It is a successful method but not efficient due to its high execution time. Therefore, this method may help creation of blacklists but it is not a suitable real time classification processes [8]. The third method is signature check that is implemented only for classifying executable codes, not for phishing or scripting. Its performance is not good [9] [10]. Recent studies have focused on automated solutions using Machine Learning (ML) methods. In this study, we propose novel context-sensitive and keyword density based supervised ML algorithms using Support Vector Machine (SVM), Maximum Entropy (MaxEnt), and Extreme Learning Machine (ELM).

Since it is easy to prepare data and training is fast with a small number of features, non-content based features have been preferred generally on malicious webpage detection with ML techniques. However, in our opinion, each word of an HTML content can provide some clues about the behavior of the webpage. Therefore, we focus on the content of the webpages rather than adding other features like URLs, screen-shots, DNS server relationships etc. These mentioned features may have totally different characteristics/semantics and they may degrade the expected results. By considering the data as not only a text but also a webpage, we propose a new method (keyword density) for deciding weights of features while the other studies use conventional methods, existence or frequency of keywords. In addition to feature modifications, we study the ML techniques, SVM, MaxEnt, and ELM.

SVM has been used in most of the related works and proved its success on text classification. We propose MaxEnt because of its success on document classification [11] [12] and it has not been implemented for any malicious webpage detection study until now. ELM provides faster learning speed and less human intervention than SVM [13] [14]. We study with ELM because of its learning speed with 100 thousand webpages and 800 thousand features. By increasing

the efforts on data processing phase, we are able to increase the accuracy level of detecting the malicious webpages up to 98.28% true positive ratio [10]. Even the most successful recent studies provide 97.8% true positive ratio with 2.2% false positive. Therefore, our method can be reported among the best performing state-of-the-art approaches.

The outline of the study is as follows. In section 2, we explain related works about malicious webpage detection studies. In section 3, we give an overview of the supervised ML techniques used in this study. Section 4 presents information about the experimental setup, data preparation and the observed performances of the proposed ML algorithms. Finally, we give our concluding remarks and future work in Section 5.

2. Related Work

In this section, we give information about the related studies and existing approaches to detect malicious webpages. This summary combines utterly diverse concepts which have different evolution processes, usage areas, history, and effect types. We explain related works of each sub topic in four different parts:

(1) *Malicious Webpage Identification and Detection without Using Machine Learning Techniques*: Chen and Guo show that phishing emerges as a recent type of network attack using webpages that cheats users in order to reach personal information in 1990s [15]. They suggest an end-host based anti-phishing algorithm to detect and prevent phishing attacks by using generic characteristics of the hyperlinks. They successfully detect 195 out of 203 phishing e-mail attacks. Moshchuk et al. design a proxy-based anti-malware tool that uses Virtual Machines (VM) [16][14]. User reaches the webpage after the tool renders it in a VM. Execution-based detection is a new approach instead of signature control of other anti-malware tools. Invernizzi et al. do not filter webpages in client side in run-time [17] [18] [19], instead, they focus on crawling malicious webpages. Therefore, they search the web by starting from a known malicious webpage and crawl only malicious ones by comparing with an initial seed. These studies put forward the importance of malicious webpage problem and provide different perspectives for the solution of the problem.

(2) *Document and Webpage Classification (Web Filtering) Using ML with Content Features*: Malicious webpage detection may be thought as a sub class of document classification and webpage filtering. There are too many studies related with these topics but we choose to focus on three of them because of their high correlation with our study. Nigam et al. propose a Maximum Entropy usage in text classification because of its similarity with language modeling, part-of-speech tagging and text segmentation [11]. They show that Maximum Entropy is a valid technique for text classification for obtaining better results in some conditions when compared with Naive Bayes. Pang et al. publish a research about binary classification of documents and it attracted so much attention [12]. The study compares three successful ML techniques for sentiment classification of HTML documents. The performance of SVM, Maximum Entropy, and Naive Bayes are similar. Chau and Chen focus on the search of related webpages and implement neural network and SVM techniques with content and link features in order to be used for topic specific search engines [20]. They use HTML tags in order to decide the importance of words in a webpage document. Because, the location of the words in an HTML document also provide considerable information. Similar to these studies, we extract valid ML techniques and importance of HTML tags for the calculation of keyword density.

(3) *Using Machine Learning with Non-Content Features*: Malicious web sites include other common features as well as content. There are studies that use the count or length of some static features because feature count is small and execution times of classification are low when the count of items are selected as features. Six DNS and server related features are used by Seifert et al. [21]. These features are the number of unique HTTP servers, redirects, redirects to different countries, domain name extensions and unique DNS servers. Decision Tree ML technique is used with nearly 20 thousand samples and resulted with 74.5% true positive and 97.4% true negative results. Seifert et al. use eight different static features that are the numbers of applet and object tags, script tags, XML processing instructions, frames and iFrames, indications of redirects, source script tags, functions that indicate script obfuscation, visibility of iFrames [22]. Decision Tree ML technique is used with almost 22 thousand samples and resulted with 94.1% true positive and 53.8% true negative results. 171 features are used by [6]. 154 of them are counts of native JavaScript functions such as abs(), acos(), apply(), etc. Nine of them are static features of HTML documents such as word count, line count, symmetry of tags, etc. Eight of them are the counts of the use of ActiveX objects such as Scripting.FileSystemObject, WScript.Shell, Adodb.Stream, etc. Naive Bayes, Decision Tree, SVM and Boosted Decision Tree algorithms are trained and tested with almost 1,100 samples and Boosted Decision Tree showed best results with 92.6% true positive and 92.4% true negative rates. 77 static features are used by Canali et al. [23]. 19

of them are HTML related features such as iFrame tag count, hidden element count, the percentage of white spaces, known malicious pattern count. 25 of them are JavaScript related features such as the number of occurrences of eval(), setTimeOut(), setInterval() functions, number of built-in functions commonly used for obfuscation routine, the number of long variable names, the number of string modification functions, etc. 33 of them are URL link and host features such as suspicious URL pattern count, presence of IP address, presence of sub-domain, value of TTL, registration date, etc. SVM, Random Tree, Random Forest, Naive Bayes, Logistic, J48 and Bayes Net are used with almost 200 thousand samples. As a result, 94.5% true positive and 95.8% true negative results were obtained. 39 static features are used by [4]. Ten of them are URL features such as number of words, length of hostname, etc. 29 of them are page content related features such as applet count, embedded script count, abnormal visibility style, etc. SVM, Decision Tree, Naive Bayes, KNN and ANN are used with almost 30 thousand benign samples. As a result, 96.01% accuracy is obtained with ANN. These studies are related with our study considering the problem and the sample data sets. However, feature set size and the characteristics, algorithms and experiment processes are completely different.

(4) *Using Machine Learning with Content Features*: The most recent studies are listed in this part. Bannur et al. use conventional URL features, number of page links, semantic, and visual features of web contents [8]. They choose to implement SVM and Logistic Regression ML methods. They decrease the error rate down to 1.9% with SVM. Abbasi et al. put forward a similar research with medical webpages [24]. Their study is rich in classification method varieties because they implement 21 classification methods. Graph-based methods are listed from the most successful to the least successful as RTL-GC, QoC, Mass Estimation, QoL, TrustDistrust, TrustRank, AntiTrustRank, BadRank, Cautious Surf, PageRank, and ParentPenalty. On the other side, content-based methods are RTL-CC, AZProtect, SVM-Linear, Logistic Regression, SVM-RBF, SVM-Poly, Bayes Network, Neural Network, C4.5 and Naive Bayes sequentially. Last related study is done by Kazemian and Ahmed [10]. They use URL, page links, semantic and visual features together as previous studies did. They employ three supervised ML techniques KNN, SVM, and Naive Bayes, and two unsupervised ML techniques K-Means and Affinity Propagation. Their study proposes that RBF-SVM technique is the best one with the whole feature types. The true positive ratio is 97.8% and true negative ratio is 55.1%.

Non-content features have been preferred generally for malicious webpage detection with ML techniques. There are two main reasons; easy data preparation and fast training with small size of features. However, each word of an HTML content may give a clue about the meaning and behavior of web sites. Therefore, in this study, we focus on the content of the webpages. We do not prefer adding other features like URLs, screen-shots, DNS server relationships etc. These features have totally different characteristics and they may distort the expected results. By considering the data as not only a text but also as a webpage, we propose a new methodology (keyword density) for deciding weights of features while the other studies use conventional methods, existence or frequency of keywords. In addition to feature modifications, we work with different ML techniques, Maximum Entropy (MaxEnt) and Extreme Learning Machine (ELM). We propose MaxEnt because of its success on document classification [11] [12] and it has not been implemented for any malicious webpage detection study until now. Secondly, ELM provides faster learning speed and less human interference is required than SVM [13]. We study with ELM because of its learning speed (there are approximately 800 thousand features and 100 thousand webpages). Third ML technique is SVM that has been used in most of the related works and proved its success. Our aim by using SVM is that it can be a base classification technique in order to obtain meaningful comparison with less studied ML techniques, MaxEnt and ELM. Last contribution of this study is the accuracy level of the obtained results. By increasing the efforts on data processing step phase, we achieve to improve the level of accuracy, even when compared to a recent and similar study by [10] which has achived 97.8% true positive ratio with 2.2% false positive.

3. Applied Machine Learning Models

In this section, we give an overview of the supervised ML techniques used in this study. The first ML technique is Support Vector Machine (SVM) which has been widely used in the literature for malicious web page detection. Second one is Maximum Entropy (MaxEnt) which has never been used for malicious web page detection before but it is reported in the literature with very good results provided for document and web page classification. Third one is Extreme Learning Machine (ELM) which has never been used for web page classification even though its learning speed is very high. SVM obtains the best results compared to most of the recent/popular classification models such as Logistic Regression (LR), Bayes Network, Neural Network, Naive Bayes, K-Nearest Neighbors, K-Means, Affinity Propagation etc. Therefore, SVM can be reported as one of the best binary classification methods by producing best results in similar malicious webpage detection studies [8] [10]. Besides, we intend to experiment with previously unused techniques, MaxEnt and ELM. Although rationale behind these three supervised ML techniques are quite different, the reasons for using them can be summarized with their efficiencies and similar uses in previous studies. MaxEnt has been shown to be an effective method on text categorization and document classification but it has not been used for webpage classification so far [12]. ELM is a popular classification method and very suitable for text classification because of its learning speed and it has not been used as a web content classification tool either [25].

3.1. Support Vector Machines (SVM)

SVM is one of the most widely used data classification techniques for binary classification of high dimensional data. The concept of SVM is introduced by Boser et al. [26]. SVM is a binary classification technique that finds optimal margin between the training patterns and the decision boundary on separable data. Costes and Vapnik designed the present and more convenient form of SVM for non-separable data [27]. The main target of the SVM model is to determine an optimal hyperplane which separates the examples of different classes for given training data points. The decision hyperplane is constructed by maximizing the distance of hyperplane and the nearest examples from different classes that are called support vectors. SVM can be formulated as given below [28]:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \tag{1}$$

for (sample, class label) pairs of training set (x_i, y_i) where $i = 1, ..., l$, $x_i \in R^n$ and $y \in 1, -1^l$,

subject to $y_i(w^T \phi(x_i) + b)$>=$1 - \xi_i$, ξ_i>=0.

Training vectors x_i are mapped into a higher dimensional space by using the function ϕ. SVM produces a hyperplane with the maximal margin in this space. Lastly, C is the penalty parameter and $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is the kernel function. The kernel function of SVM is important in many cases because it is crucial for non-separable patterns. If a pattern has separable data, SVM can find an optimal hyperplane between two classes easily. However, if the pattern is non-separable, SVM needs to map the current data into new space by transformations in order to make the current pattern separable. The transformation is processed with a predefined Kernel Function.

Mostly used kernel functions are Radial Basis Function (RBF), Linear, Sigmoid and Polynomial. In this study we prefer Linear and RBF functions Linear Function Radial Basis Function(RBF). Because RBF is the most popular and commonly used function for SVM and Linear function is suggested for solving large-scale classification problems such as text classification [29].

$$K(x_i, x_j) = x_i^T x_j \qquad \text{(Linear Function)}$$

$$K(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2) \quad , \gamma > 0 \qquad \text{(Radial Basis Function(RBF))}$$

3.2. Maximum Entropy (MaxEnt)

MaxEnt is a statistical classification modeling technique which was introduced by Berger et al. (1996) [30] in order to solve several natural language processing problems. Since then, the method has been used for lots of text classification studies [11] [12] [31] [32] (see Figure 2).

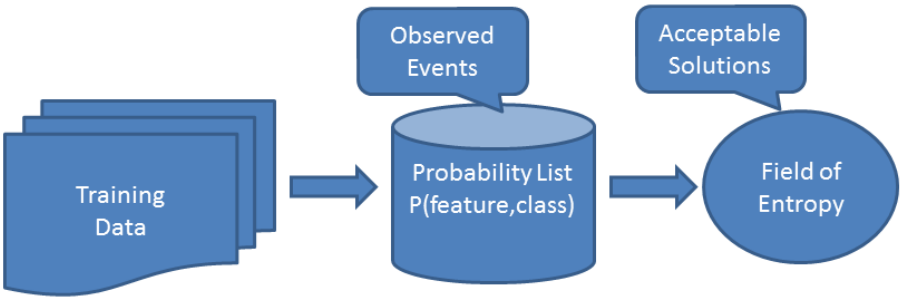

Figure 2: Maximum Entropy Model

Training data include relationship information between features and class types. The maximum entropy model uses these relationships in order to estimate probabilities. If training data is a text, this algorithm models conditional distribution of the words of texts in classes. Probabilistic distribution of a text classification model is computed as given in [30] :

$$p(c|d) = \frac{1}{Z(d)} exp(\sum_i \alpha_i f_i(d,c))) \tag{2}$$

where $Z(d)$ in Equation 2 is a partition function which makes normalization. It is computed as:

$$Z(d) = \sum_c exp(\sum_i \alpha_i f_i(d,c))) \tag{3}$$

In Equations 2 and 3, c indicates class type, d indicates document. The parameter α_i refers weight of feature and it must be learned by estimation. Various estimation algorithms could be used for this step such as Limited-Memory Variable Metric (L-BFGS) [33], Orthant Wise Limited-memory Quasi Newton (OWLQN) or Stochastic Gradient Descent (SGD) [34]. $f_i(d,c)$ indicates the impact of a feature i. The impact of the function could has binary or positive integer value. While binary value is used for occurrence of a word in text, integer value could give more information such as frequency of word. More precisely the function is formulated as [32]:

$$f_{(w,c')}(d,c) = \begin{cases} 0, & \text{if } c \neq c' \\ \frac{N(d,w)}{N(d)}, & \text{otherwise} \end{cases} \tag{4}$$

where N(d,w) in Equation 4 is the density of word w in document d and $N(d)$ is the total density of words d.

3.3. Extreme Learning Machine (ELM)

ELM is a learning algorithm for single-hidden layer feed-forward neural networks (SLFN) (see Figure 3 for SLFN) [25]. The main deficiency of feed-forward neural networks is the slowness problem because of slow gradient-based algorithms used in training step and tuning operation of all the parameters of the network iteratively. In order to handle this bottleneck, Huang et al. developed an algorithm that chooses the input weights randomly and decides analytically

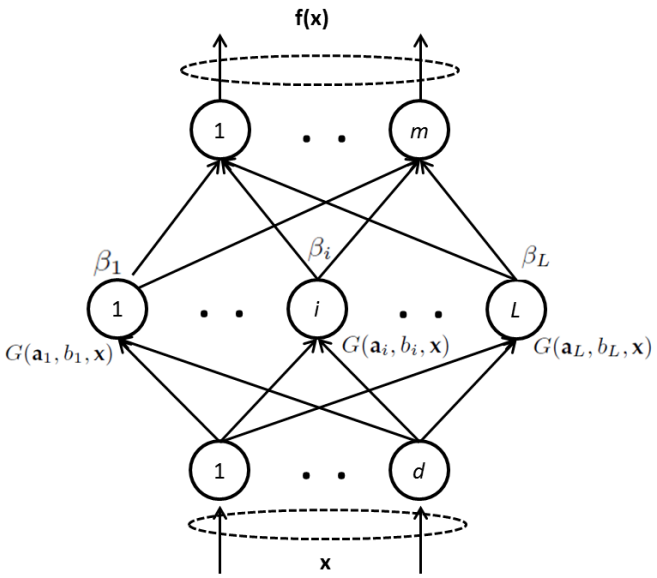

Figure 3: Single-hidden layer feed-forward network.

the output weights in order to obtain best generalization performance with extremely fast learning speed [13][25] [35] .

The output of SLFN having L number of hidden nodes can be represented with the formula below;

$$f_L(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x), x \in R^n, a_i \in R^n \tag{5}$$

where learning parameters of hidden nodes are a_i and b_i and β_i is the connection weight between the ith hidden node and the output node. $G(a_i, b_i, x)$ is the output of the hidden node with the input x. Generally, the additive hidden node based on activation function is $g(x) : R \rightarrow R$. $G(a_i, b_i, x)$ is given by

$$G(a_i, b_i, x) = g(a_i.x + b_i), b_i \in R, \tag{6}$$

where $a_i.x$ represents the inner product of $a_i \in R^n$ and $x \in R^n$ vectors. For the training samples $\{(x_i, t_i)\}_{i=1}^{N} \subset R^n \times R^m$, if output of the network is equal to the targets, we have

$$f_L(x_j) = \sum_{i=1}^{L} \beta_i g(a_i.x_j + b_i) = t_j, j = 1, ..., N \tag{7}$$

Equation 7 could be written as:

$$H\beta = T \tag{8}$$

$$\mathbf{H} = \begin{bmatrix} h(x_1) \\ . \\ . \\ . \\ h(x_N) \end{bmatrix} = \begin{bmatrix} G(a_1.b_1 + x_1) & . & . & . & G(a_L.b_L + x_1) \\ . & & . & & . \\ . & & . & & . \\ . & & . & & . \\ G(a_1.b_1 + x_N) & . & . & . & G(a_L.b_L + x_N) \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ . \\ . \\ . \\ \beta_L^T \end{bmatrix} \textit{ and } \mathbf{T} = \begin{bmatrix} t_1^T \\ . \\ . \\ . \\ t_L^T \end{bmatrix}$$

where H is the hidden layer output matrix of the SLFN, the ith column of H is the jth hidden node output that is relevant to the input $x_1, x_2, ..., x_N$. $h(x)$ is the hidden layer feature mapping. $h(x_i)$ is relevant to the the ith input x_i. It has been proved, if the activation function G is infinitely differentiable in any interval then the hidden layer parameters could be randomly generated [35].

3.4. Implementation of the Machine Learning techniques

There are some important issues that have significant influence on SVM performance/results [28]. First issue is *scaling*. If data is not scaled, greater numeric ranges may dominate smaller numeric ranges. Therefore, scaling ranges [-1,+1] and [0,1] are generally suggested and the range [0,1] is used in our study. Second issue is the model selection. RBF Kernel is suggested as the first choice because of its accuracy and popularity for SVM. Besides, Linear-SVM is also suggested for text classification with its speed and convenience for large-scale data. Codes of LIBSVM [36] for RBF-SVM and LIBLINEAR [37] for Linear-SVM are used in the experiments.

The last issue is finding the best parameters for these models. In order to handle this issue, we use cross-validation and grid-search solution. Basically, we divide our training set into five subsets and use four of them for training and one of them for testing with various pairs of parameters. After that, we chose C for Linear-SVM and (C, γ) for RBF-SVM which belongs to the best accuracy on grid-searches. They are $32, 64, 64, 128, 32$ for Linear-SVM and $(128.0, 0.125), (8, 8), (8, 8), (32, 8), (8, 8)$ for RBF-SVM in order to model training set sizes 500, 2500, 5000, 25000, and 50000 respectively. All values are powers of two because parsed values in grid are $2^{-5}, 2^{-3}, ..., 2^{15}$ for C and $2^{-15}, 2^{-13}, ..., 2^{3}$ for γ. After the parameters are obtained, models are created for each algorithm and training set. Feature vectors of each webpage are predicted with created model and predictions are saved. In the final phase, we obtain true positive, false positive, true negative and false negative ratios by comparing actual results and predicted results of webpages.

Maximum Entropy Settings : A MaxEnt classification library *MAXENT* that is implemented at Tsujii Laboratory in University of Tokyo [38] is used in our study. The library supports L1 and L2 regularization. Besides, OWLQN, LBFGS and SGD algorithm are used for optimization. First, we need to update input sets because the values of the method should be integers in range [0,255]. Binary representation is not a problem but other types of inputs including floats are scaled again. As recommended, we execute L1 regularization with OWLQN and L2 regularization with LBFGS. Iteration count is selected as 300. Bigger iteration counts consume more computation time and increases accuracy. This program, firstly, saves all training data in a model. Then, it performs training and checks test set like SVM. As a result, models are saved in order to use again and also true positive, false positive, true negative and false negative ratios are obtained. Models of this algorithm include actively used features and their effects with a list as given in Table 1.

Extreme Learning Machine Settings : ELM implementation used in the experiments is obtained from the web site of Nanyang Technological University [39]. Samples of this library suggests random data selection and more than one trial for each data set because each trial gives different results which is caused by random input weights on decision of output weights. Random selection of data samples is unnecessary for our problem because we already created different size sample sets. On the other hand, we get average performance for 50 trials of each sample set as recommended. The library is implemented for small size feature sets. However, feature set size is very large in our problem so that matrix is very big and sparse. Therefore, we modify the source code by using sparse matrices. We obtain averages of true positive, false positive, true negative and false negative ratios in 50 trials results.

Table 1: Small part of MaxEnt model.

Class Label	Feature ID	Lambda Weight of Feature
0	10496	0.122787
1	1052	-0.220663
0	1052	0.220663
1	1060	-0.043466
0	1060	0.043466
0	10730	0.033599
0	1076	0.118291
1	11	0.004689
0	11	-0.004689
0	1102	0.092983
1	1103	-0.317303

4. Data Preparation and Performance Evaluation of the Proposed Algorithms

In this section, experimental setup, data preparation and the observed performances of the proposed Machine Learning (ML) algorithms are presented. Data preparation tasks and the experiments are carried out on an Intel Core I5-4570 3.20 GHz computer having 8 GB RAM.

4.1. Preparation of the data to be used in the machine learning models

Two data sets are used in this study, Phistank (2016) [40] for malicious web sites and Alexa (2016) [41] for safe (benign) web sites. Both of the data sets have been used by most of the recent studies [4] [10] [17]. Phistank is a community site to submit, verify, track, and share phishing data. The community provides current malicious web site list of URLs. Alexa is an analytic tool and provides a list of top ranked one million web site URLs. We assume that pages of Alexa are benign because they are extremely popular and top ranked pages over the world. These data sets provide us two hundred thousand of the benign page URLs. Details of the data sets are given in Table 2.

Counts	Malicious	Safe(Benign)
URL	28848	200000
Crawled Webpage	26039	181665
English Webpage	20799	99974

Table 2: Details of the data sets used in the experiments [40][41].

Data preparation is a time consuming process of data mining. First, we extract meaningful data from the webpages. In order to handle this issue in a short time and with least error rate, we use a keyword density extractor library designed by Comodo Group ® [42]. Although the preparation of the data is not the main focus of our study, we observe that the data preparation phase before processing with machine learning techniques can be a time consuming process. Parallel computing and big data processing techniques can be efficiently used in this area. The data preparation tasks are explained in the following parts.

In the crawling step, 28,848 malicious and 200 thousand benign URLs are used in order to request HTML contents of the webpages. However, significant part of them are eliminated as shown in Table 2 (9% of the URLs are eliminated because thay are not reachable). Language detection is executed by JLangDetect [43] on crawled HTML contents and 42% of them are filtered since their language is not English or word count is smaller than five. If there is not sufficient content, both language detection and classification with content are meaningless issues. At the end of the crawling step, we obtain 20,799 malicious and 99,974 benign English webpage HTML content saved in text files. Getting the

HTML content of web sites, saving them to text files, detecting of their language, and delay of unreachable URLs take hours of computation even with 20 second timeout limit used for unreachable URLs.

We examine HTML content of webpages in order to extract keywords. In order to obtain correct feature set, these contents are parsed and conventional content processing methods are used. Implemented processes can be summarized as below:

Article Extraction: Webpages may contain both valuable information and irrelevant texts. Article extraction helps obtaining only valuable information from a webpage. In order to filter irrelevant texts, it removes navigation links, advertisements, menu items, selection items, videos, images etc.

Removing Some Special Characters: Removal of special characters, punctuations, apostrophes, words containing only one or two characters etc. increase clearness of text analyzing.

Stemming: This process is summarized briefly as reducing derived or inflected words in order to obtain base form of the word. For instance, 'stems', 'stemmer', 'stemming' and 'stemmed' have same root 'stem' so each of them can be considered as root word.

Stop Word Elimination: generally refers to most common words in a language. Using them in text processing does not express a meaning. Some example stop words are 'but', 'are', 'some', 'the', 'who', 'and', 'etc'.

The second issue of feature extraction is *scoring*. Recent studies generally use binary representation or TF-IDF because they are simple methods and they present satisfying results for text classification. However, webpages have more features than a regular text such as HTML tags so we obtain keyword frequencies of the HTML contents and compare them with conventional methods. In other words, we prefer to use both Binary Representation, TF-IDF and Keyword Density.

Binary Representation: This feature type is only interested in a keyword that occurs in a text or not. If the text T contains the keyword k, value of feature k in the feature vector of T is 1. Otherwise, its value is 0. This method is easy and efficient in size and time when values are binary.

Keyword Frequency and TF-IDF: Frequency of a word shows its importance in the text. However, some words create noise like stop words. Therefore, Term Frequency Inverse Document Frequency (TF-IDF) is one of the mostly used approaches in text mining rather than Term Frequency (TF). TF-IDF is the division of term frequency which is the frequency of the word within the document. It is used to inverse the document frequency which is the number of documents containing that word. This approach helps highlighting words that occur rarely in all data set but frequently in the document [8].

For a keyword frequency instance, Wikipedia keyword extraction link (https:// en. wikipedia. org/ wiki/ Keyword_extraction) is analyzed and 159 keywords are extracted. Frequencies of the top ranked ten keywords are listed in Table 3. According to the Table, 'languag', 'document', 'method', 'process' and 'text' keywords are more relevant with this webpage although frequency of 'term' is higher than their frequencies.

Rank	Keyword	TF	TF-IDF
1	keyword	14	0.00706714
2	extract	9	0.01345291
3	wikipedia	8	0.02346041
4	method	6	0.00215750
5	term	5	0.00017319
6	languag	4	0.00048309
7	text	4	0.00090930
8	assign	3	0.00262467
9	document	3	0.00029898
10	process	3	0.00046418

Table 3: Top ten Keyword Frequencies of a Wikipedia Webpage

Keyword Density: is a successful and commonly used feature for document classification but HTML contents include more valuable properties. Title, meta keywords, headers and other HTML tags are some of these valuable information holders. It is clear that a word found in header or title is more valuable than a word in text body. We analyze HTML content of a webpage considering tags of HTML, and score each keyword by considering its frequency

and tags. This score is used as *density* in the rest of this study.

Rank	Keyword	Density Score	Density Ratio
1	keyword	117.2	0.12752587
2	extract	111.5	0.12129783
3	edit	33.1	0.03600859
4	assign	27.3	0.02969893
5	languag	27.1	0.02948135
6	term	21.3	0.02317169
7	text	18.2	0.01979928
8	method	18.2	0.01979928
9	search	18.1	0.01969050
10	process	18.1	0.01969050

Table 4: Top 10 Keyword Densities of a Wikipedia Webpage

Feature Set Generation: In order to generate feature vectors, a feature set should be defined so that all keywords in feature files and document frequencies of them are saved. The most time consuming step is the construction of this table because of the large number of select and update transactions on the database. After all keywords are obtained, the table is simplified to keep only small and meaningful sets. As a result, we create five tables, one of them keeps all keywords which are found in keywords of one hundred thousand webpages and four of them are simplified versions of the all keywords table as shown in ER Diagram (see Figure 4).

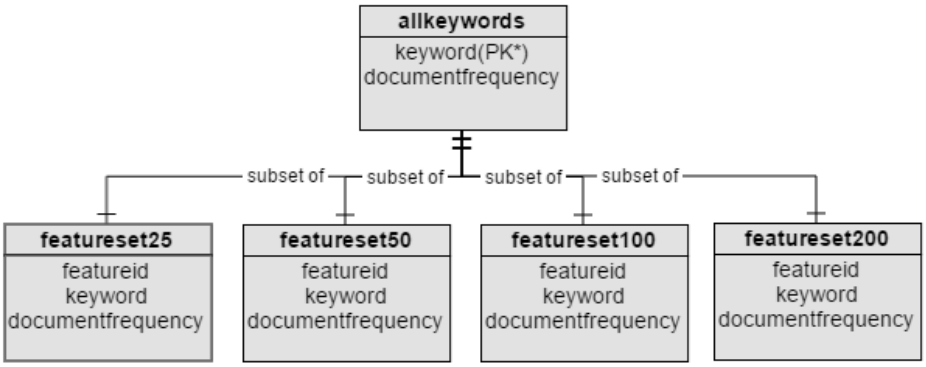

Figure 4: Entity Relationship Diagram

allkeywords table: is created by using the words used in all webpages and the table contains 800 thousand items. Each row keeps a unique keyword which will be used as feature and its document frequency. In order to decrease effort, this step could be handled by using a predefined common English keywords list instead of composing 'allkeywords' table but self constructed table contains lots of words which are not contained in English dictionary but exist in web. Sample row from the table shows that *download* keyword is found in 17,162 distinct documents as a keyword.

featuresetN tables: The whole rows in the *allkeywords* table are not considered as features because the table includes words with spelling errors, concatenated words, and language interfere such as 'installatiebedrijf', 'movingforwardtrademarklogo', 'strategisch'. Luckily, frequencies of such problematic words give a clue about their validity. In order to handle this problem, feature sets are created by simplifying *allkeywords* table. In other words, the feature set tables are subsets of *allkeywords* table. They are simplified tables containing the keywords whose document

frequencies are above the threshold n. We create and use featureset25, featureset50, featureset100, and featureset200 tables which contain 33148, 20638, 12988 and 8288 rows respectively. Each row indicates a feature by keeping id, keyword and document frequency.

Conversion into Feature Vectors: Last step is the expression of each webpage as feature vector in order to use on ML methods. On this step, we create three vectors for each webpage; existence based, TF-IDF based and keyword density based. All feature vectors contain maximum 100 features in order to reduce data size. Existence based feature vectors keep only distinct first 100 keywords in text because we do not have more meaningful data due to binary representation. TF-IDF based feature vectors contain top ranked 100 TF-IDF scores per webpage. Similarly, KD based feature vectors contains ratios of top ranked 100 keyword density for each webpage. Lastly, TF-IDF values are scaled to [0,1]. Existence values do not need scaling because their value set is 0,1. Also, keyword density ratio values do not need to be scaled since these values are already in range [0,1].

4.2. The performance evaluation of the Machine Learning algorithms

In this part of the experiments, supervised machine learning methods, SVM, ELM, and MaxEnt are executed with a benchmark for the detection of malicious webpages. For this process, we use extracted feature vectors for training and testing data sets. For the evaluation of these ML methods, models are created with training sets and class labels of test sets are predicted with a created model as given in Figure 5. The performance of algorithms are examined in the following parts.

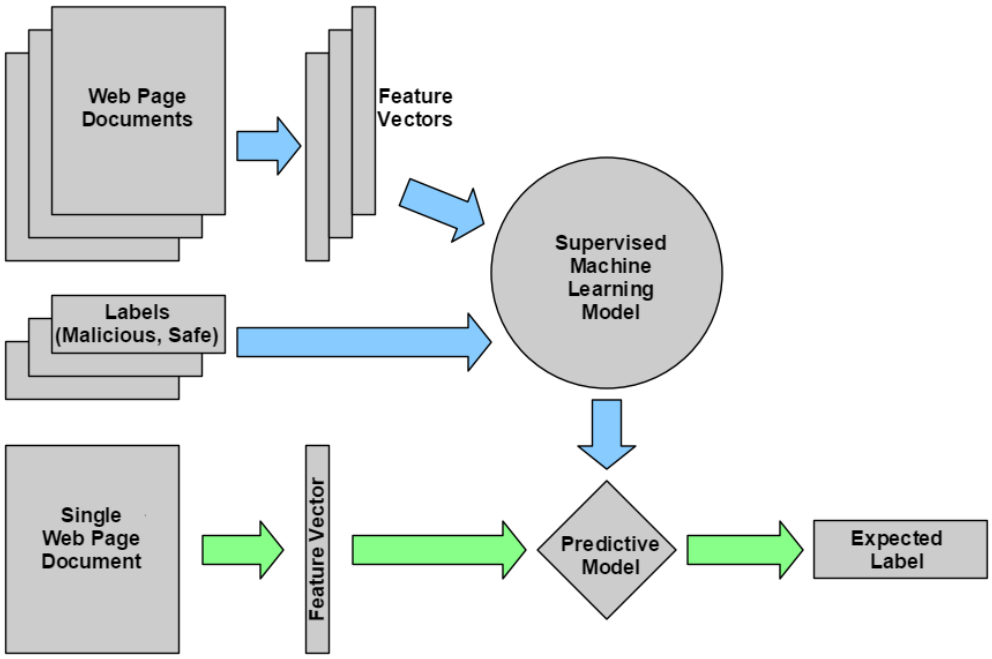

Figure 5: Supervised Machine Learning Model

The Effect of Data Set Size for the Machine Learning algorithms: Machine learning methods are executed with different data set sizes that are randomly selected as 20 thousand malicious and 80 thousand safe webpages. Properties of the data sets are described in Table 5.

Although researches show that the percentage of malicious webpages in the real world is roughly 0.1% [17], this percentage is not suitable to generate a predictive model with ML algorithms during the experiments. Therefore, we use 20% malicious webpages in our data set. 50% of the webpages are used for training and other half is used for testing. The same ratio of malicious webpages to safe webpages is used in testing and training sets. Keywords

	1000	5000	10000	50000	100000
Fold count	100	20	10	2	1
Train set size	500	2500	5000	25000	50000
Test set size	500	2500	5000	25000	50000
Malicious count	100	500	1000	5000	10000
Safe count	400	2000	4000	20000	40000

Table 5: Properties of input sets

density percentages in webpages are used as features so that feature values are between [0.0, 1.0] and their sum is 1 for each webpage feature vector. Feature vector size is limited with maximum weighted 100 features for each webpage. Besides, used feature set, featureset25, includes the features whose document numbers are bigger than 25. Averages of accuracies are listed in Table 6 where the accuracy is defined as successfully labeled webpages divided by all webpages.

	1000	5000	10000	50000	100000
RBF-SVM	94.60	97.24	97.44	**98.24**	98.01
Linear-SVM	94.80	97.36	97,26	**97.75**	97.55
ME-L1	94.80	96.56	96.72	**97.33**	96.94
ME-L2	95.00	96.72	97.20	**97.28**	96.81
ELM	87.29	88.26	88.13	88.74	88.05

Table 6: The Effect of Data Set Size and ML Algorithms on Accuracy percentage. Numbers in the column headings show the size of the data sets

Results are obtained by getting average accuracies of n different parts of 100000/n data sets. For instance, different 10 tests are done with the data including 10 thousand sample and averages are listed on the Table 6. We compare the algorithms with 50 thousand samples because most successful results are obtained with this sample size. Their accuracies percentages in descending order are RBF-SVM (98,24), Linear-SVM (97,75), MaxEnt-L1 (97,33), MaxEnt-L2 (97,28) and ELM (88,74).

Another concern for a successful detection is the confusion matrix that is used for describing the performances of implemented classification models. It contains not only successfully detected malicious webpage percentage but also the percentages of mislabeled webpages. False positive rate is an important value in this study. In Figure 6, the confusion matrix gives four sections. True positive section indicates the percentage of successfully detected malicious webpages. True negative section indicates the percentage of successfully detected safe webpages. False positive represents that safe webpages labeled as malicious incorrectly. False negative represents that malicious webpages labeled as safe incorrectly.

The success of the algorithm is determined with high True Positive (TP) and low False Positive (FP) ratios. Their TP ratios, also called recalls, in descending order is MaxEnt-L1 (94,68), MaxEnt-L2 (94,36), Linear-SVM (94,08), RBF-SVM (93,22) and ELM (48,93). On the other side, their FP ratios in ascending order is RBF-SVM (0.50), ELM (1.31), Linear-SVM (1.33), MaxEnt-L2 (1.99) and MaxEnt-L1. Even FP ratio of ELM is low, it does not show success of this method due to the very low TP ratio.

4.3. The effect of feature type

One of the biggest contributions of this study is the selection of feature type. The other studies generally use binary representation or TF-IDF. In addition to these conventional methods, we try to represent results of keyword densities. In this section, we compare their performances. These experiments are conducted with ten fold data sets including ten thousand samples that contain five thousand training and five thousand test samples. These tests are executed with Linear-SVM and MaxEnt-L1 algorithms.

Feature vector sizes are limited with 100 features which are scaled into [0,1] for each webpage. According to the Tables 7 and 8, Keyword Density is the best option as accuracy, result and f-measure and gaps are very small with binary representation.

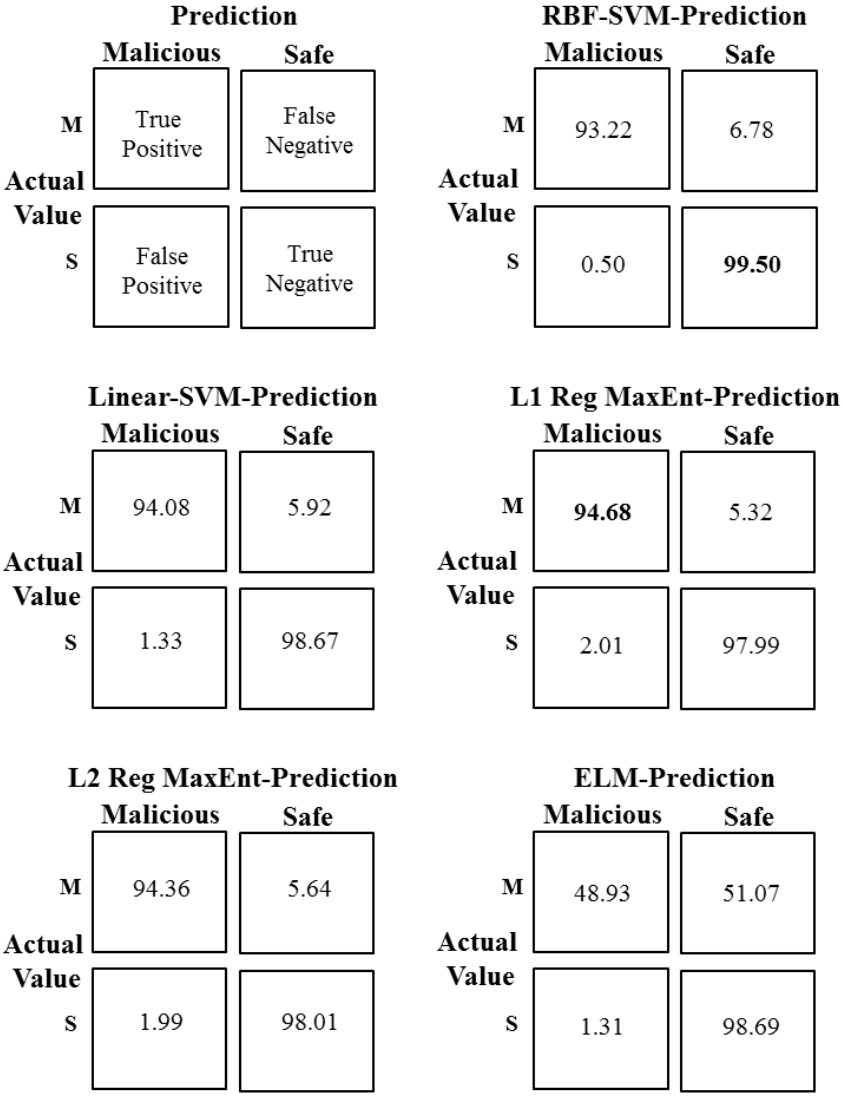

Figure 6: Confusion Matrix of the algorithms for 50 thousand webpages

Linear-SVM	Accuracy(%)	TP(%)	TN(%)	FP(%)	FN(%)
Binary Representation	96.98	91.70	98.30	1.70	8.30
TF-IDF	96.12	84.00	**99.15**	0.85	3.88
Keyword Density	97.26	92.00	98.58	1.42	2.74
MaxEnt-L1	Accuracy(%)	TP(%)	TN(%)	FP(%)	FN(%)
Binary Representation	96.26	91.10	97.55	2.45	3.74
TF-IDF	93.18	90.90	93.75	6.25	6.82
Keyword Density	**96.72**	**92.40**	97.80	2.20	3.28

Table 7: The Effect of Feature Types on Accuracy

Linear-SVM	Recall(%)	Precision(%)	F-Measure(%)
Binary Representation	91.70	93.10	92.39
TF-IDF	84.00	**96.11**	89.65
Keyword Density	92.00	94.19	**93.08**
MaxEnt-L1	Recall(%)	Precision(%)	F-Measure(%)
Binary Representation	91.10	90.29	90.69
TF-IDF	90.90	78.43	84.21
Keyword Density	**92.40**	91.30	91.85

Table 8: The Effect of Feature Types on Statistical Analysis

4.4. The effect of feature set size

We save extracted keywords from 10 thousand webpages in order to use them as our feature set. However, we need to put a lower limit to their document frequencies due to the problematic words. In order to check efficiency of this limit, different number of document frequency limits are put into the keywords Table and results are compared. These experiments are conducted by ten fold data sets including ten thousand samples which contain five thousand training and 5 thousand test samples. Feature vector sizes is limited with 100 features for each webpage. These tests are executed with Linear-SVM and MaxEnt-L1 Regularization. All keywords Table (FeatureSet0) is reduced to four Tables. As shown in Table 9, row count of these tables are decreased from 800 thousand to almost 8 thousand. Average number of different features in vectors of training set, that is used for modeling, is also decreased five times.

	Table Row Count	Number of Features
FeatureSet0	789,946	41,230
FeatureSet25	33,148	21,980
FeatureSet50	20,638	17,169
FeatureSet100	12,988	12,257
FeatureSet200	8,288	8,173

Table 9: Row counts and number of features for feature sets

According to the results in Tables 10 and 11, there is no difference on the accuracy of methods although feature set size is significantly decreased. Equality on results despite of relatively feature count could be explained by active feature count. Actually, even fewer features are used for classification actively. For example, due to the active feature lists of MaxEnt-L1 models approximately one thousand features are used for classification according to the Table 12 although training sets have up to 41,230 different features.

4.5. The execution time of the proposed algorithms

In this section, we compare the execution times of ML algorithms on the data set having 5,000 training and 5,000 test feature vectors of web pages obtained by using featureset200 and having a 1:4 ratio of malicious to safe web

Linear-SVM	Accuracy(%)	TP(%)	TN(%)	FP(%)	FN(%)
FeatureSet0	97.34	89.50	99.30	0.70	10.50
FeatureSet25	97.26	92.00	98.58	1.42	8.00
FeatureSet50	97.30	89.80	99.18	0.82	10.20
FeatureSet100	97.34	89.90	99.20	0.80	10.10
FeatureSet200	97.00	88.10	99.23	0.77	11.90
MaxEnt-L1	Accuracy(%)	TP(%)	TN(%)	FP(%)	FN(%)
FeatureSet0	96.74	90.70	98.25	1.75	9.30
FeatureSet25	96.72	92.40	97.80	2.20	7.60
FeatureSet50	96.60	90.10	98.23	1.77	9.90
FeatureSet100	96.74	91.50	98.05	1.95	8.50
FeatureSet200	96.64	90.20	98.25	1.75	9.80

Table 10: The Effect of Feature Set Size on Accuracy

Linear-SVM	Recall(%)	Precision(%)	F-measure(%)
FeatureSet0	89.50	96.97	93.09
FeatureSet25	92.00	94.19	93.08
FeatureSet50	89.80	96.48	93.02
FeatureSet100	89.90	96.56	93.11
FeatureSet200	88.10	96.62	92.16
MaxEnt-L1	Recall(%)	Precision(%)	F-measure(%)
FeatureSet0	90.70	92.84	91.76
FeatureSet25	92.40	91.30	91.85
FeatureSet50	90.10	92.71	91.39
FeatureSet100	91.50	92.15	91.82
FeatureSet200	90.20	92.80	91.48

Table 11: The Effect of Feature Set Size on Statistical Analysis

	Number of Features	Active Feature Count
FeatureSet0	41230	1008
FeatureSet25	21980	1048
FeatureSet50	17169	1060
FeatureSet100	12257	1059
FeatureSet200	8173	1071

Table 12: Active Feature Counts

pages. Since the execution times for larger data set sizes were prohibitely long (several days) we have limited our experiment to 10,000 data items and featureset200.

As shown in Table 13, the best algorithms considering the execution time are Linear-SVM and ELM as expected. MaxEnt with L1 shows a poor performance in training however, testing time is more important. Therefore, it can be accepted as successful. Performances of RBF-SVM algorithm and MaxEnt with L2 are really problematic so using them on a live system may not be suitable.

	ME-L1	ME-L2	Linear-SVM	RBF-SVM	ELM
Training with 5000 samples	12.94	8.21	**0.02**	5.13	**0.09**
Testing with 5000 samples	0.54	3.49	**0.03**	4.01	**0.04**

Table 13: Execution time of the proposed ML algorithms (in seconds)

4.6. Parameter settings of the proposed algorithms

In this part of our study, we present the parameter settings of our proposed algorithms in Tables 14, 15, 16, and 17.

Parameter	Value
-s Type of solver	2: L2-regularized L2-loss support vector classification (primal)
-p epsilon	the epsilon in loss function of epsilon-SVR (default 0.1)
-e epsilon	tolerance of termination criterion (default 0.01)
-B bias	if bias >= 0, instance x becomes [x; bias]; if < 0, no bias term added (default -1)
-c cost	Calculated before training with cross validation and grid search method of LIBLINEAR [37]

Table 14: Parameter settings of Linear SVM

Parameter	Value
-g gamma	Calculated before training with cross validation and grid search of LIBSVM [36]
-c cost	Calculated before training with cross validation and grid search of LIBSVM [36]
-s svm_type	type of SVM (default 0: C-SVC)
-t kernel_type	type of kernel function (default 2: radial basis function: exp(-gamma* $\|u - v\|^2$))
-d degree	degree in kernel function (default 3)
-r coef0	coef0 in kernel function (default 0)
-e epsilon	tolerance of termination criterion (default 0.001)
-m cachesize	cache memory size in MB (default 100)
-h shrinking	whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates	whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
-wi weight	the parameter C of class i to weight*C, for C-SVC (default 1)

Table 15: Parameter settings of LIBSVM

Parameter	Value
ELM_TYPE	1 (0 for regression; 1 for (both binary and multi-classes) classification)
Number of Hidden Neurons	20
Activation Function	'sig' for Sigmoidal function

Table 16: Parameter settings of ELM

Parameter	Value
Optimization Algorithm Selection	OWLQN for L1, LBFGS for L2
Iteration Count	300

Table 17: Parameter settings of MaxEnt

5. Conclusions and future work

In our opinion, that each word of a webpage gives important clues about the behavior and the content of webpages is more important than its features that are based on URL and/or domain name for the detection of malicious webpages. By making use of this idea, we propose a novel context-sensitive and keyword density based method for the classification of malicious webpages by using supervised Machine Learning techniques, Support Vector Machine (SVM), Maximum Entropy (MaxEnt), and Extreme Learning Machine (ELM). SVM is a well-known and efficient ML method for the text classification problems. MaxEnt is known to perform well on document classification and in this study, it is being used in malicious webpage detection for the first time. ELM provides faster learning speed for very large number of features (words). It can process on 800 thousand features and 100 thousand webpages very efficiently. By focusing on much its efforts on data processing phase, we are able to improve the accuracy of detecting the malicious webpages with up to 98.24% true positive ratio. Even the most successful recent work have reported 97.8% true positive and 2.2% false positive ratios [10].

By examining the experimental results, SVM and MaxEnt provide similar success rates. RBF-SVM gets the best accuracy with 98.24% and Linear-SVM, MaxEnt-L1, and MaxEnt-L2 methods closely follow RBF-SVM in terms of accuracy. MaxEnt-L1 gets the best (true positive, true negative) pair with (94.68%, 97.99%) ratio, MaxEnt-L2, Linear-SVM and RBF-SVM methods follow it. Considering the execution time, Linear-SVM, ELM and MaxEnt-L1 have similar execution times. Linear-SVM or MaxEnt-L1 show satisfactory results because of their high accuracies, low false positive rates, and low execution times. We show that these two methods maintain their speed and accuracy with large number of features and samples. The results of our study are significantly better than any reported method in the literature. Most similar and recent study [10] gets (true positive, false positive) pairs in the following order RBF-SVM (97.8%, 55.1%), Linear-SVM (92.4%, 83.2%), Nave Bayes (76.4%, 87.4%) and K-Nearest Neighbor (9.9%, 94.8%). Although, study of Kazemian and Ahmed uses additional features, URL, page link and visual, results of our study are significantly better than it. The probable reason of this difference is preprocessing the content and feature extraction issues.

Our contribution with this study can be summarized as below. Firstly, we clear up the keywords by using lots of conventional methods such as stemming, article extraction, stemming character elimination etc. Secondly, our study contributes a novel keyword extractor. While the conventional feature value types are binary representation and TF-IDF on text classification studies, we also extract keyword densities of web pages and used them as features. Feature value types do not affect very much hence keyword density shows the best results and TF-IDF shows the worst performance. Therefore, we indicate that preprocessing of web pages should not be considered as only text processing. Lastly, feature set elimination is applied because total feature count reaches almost eight hundred thousand keywords and used feature count in the sample set exceeds forty thousand keywords. However, most of these features are negligible because they are misspelled. In order to clean them, features existing in less than document frequency threshold are removed. As a result, accuracy or true positive rate does not change although the total feature count has been decreased from almost 800K to 8K, while in the experiments that we have performed a total of 8,000 features have been used out of 40,000 resulting in great savings of processing time.

As future work, we plan to apply parallel computation methods for the data preparation phase of our proposed malicious web site detection process. Therefore, more reliable data can be available and even better results will be obtained with a higher speed. Another area of research can be application of new machine learning techniques such as deep learning for the solution of this problem. A hybrid feature set system can be created including not only keyword densities but also JavaScript functions, ActiveX objects, DNS-Server relationships, and URL features. Each feature should be analyzed separately so that there is an additional work for weights of their effect on final decision. Feature selection can be added to this study. Because we showed that even one thousand keywords are sufficient for the solution of the problem. Decrease in the feature set size will positively improve the execution time of the algorithms.

Compliance with Ethical Standards:

This study is not funded by any institution.
Conflict of Interest: There is no conflict of interest between authors.

This article does not contain any studies with human participants performed by any of the authors.
This article does not contain any studies with animals performed by any of the authors.
This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent: There is no individual participant included in the study.

References

[1] International Telecommunication Union. Statistics. `http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2015/ITU\_Key\_2005-2015\_ICT\_data.xls`, 2015.

[2] Ram Basnet, Srinivas Mukkamala, and Andrew H Sung. Detection of phishing attacks: A machine learning approach. In *Soft Computing Applications in Industry*, pages 373–383. Springer, 2008.

[3] Gary Wassermann and Zhendong Su. Static detection of cross-site scripting vulnerabilities. In *2008 ACM/IEEE 30th International Conference on Software Engineering*, pages 171–180. IEEE, 2008.

[4] Abubakr Sirageldin, Baharum B Baharudin, and Low Tang Jung. Malicious web page detection: A machine learning approach. In *Advances in Computer Science and its Applications*, pages 217–224. Springer, 2014.

[5] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, Nagendra Modadugu, et al. The ghost in the browser: Analysis of web-based malware. *HotBots*, 7:4–4, 2007.

[6] Yung-Tsung Hou, Yimeng Chang, Tsuhan Chen, Chi-Sung Laih, and Chia-Mei Chen. Malicious web content detection by machine learning. *Expert Systems with Applications*, 37(1):55–60, 2010.

[7] Pawan Prakash, Manish Kumar, Ramana Rao Kompella, and Minaxi Gupta. Phishnet: predictive blacklisting to detect phishing attacks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5. IEEE, 2010.

[8] Sushma Nagesh Bannur, Lawrence K Saul, and Stefan Savage. Judging a site by its content: learning the textual, structural, and visual features of malicious web pages. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, pages 1–10. ACM, 2011.

[9] Mihai Christodorescu and Somesh Jha. Testing malware detectors. *ACM SIGSOFT Software Engineering Notes*, 29(4):34–44, 2004.

[10] HB Kazemian and S Ahmed. Comparisons of machine learning techniques for detecting malicious webpages. *Expert Systems with Applications*, 42(3):1166–1177, 2015.

[11] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67, 1999.

[12] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[13] Guang-Bin Huang, Dian Hui Wang, and Yuan Lan. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, 2(2):107–122, 2011.

[14] Ayça Deniz, Hakan Ezgi Kiziloz, Tansel Dokeroglu, and Ahmet Cosar. Robust multiobjective evolutionary feature subset selection algorithm for binary classification using machine learning techniques. *Neurocomputing*, 241:128–146, 2017.

[15] Juan Chen and Chuanxiong Guo. Online detection and prevention of phishing attacks. In *Communications and Networking in China, 2006. ChinaCom'06. First International Conference on*, pages 1–7. IEEE, 2006.

[16] Alexander Moshchuk, Tanya Bragin, Damien Deville, Steven D Gribble, and Henry M Levy. Spyproxy: Execution-based detection of malicious web content. In *USENIX Security*, 2007.

[17] Luca Invernizzi, Paolo Milani Comparetti, Stefano Benvenuti, Christopher Kruegel, Marco Cova, and Giovanni Vigna. Evilseed: A guided approach to finding malicious web pages. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 428–442. IEEE, 2012.

[18] Ajith Abraham, Yukio Ohsawa, and Yasuhiko Dote. Web intelligence and chance discovery. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 11(8):695–696, 2007.

[19] Ramón A Carrasco and Pedro Villar. A new model for linguistic summarization of heterogeneous data: an application to tourism web data sources. *Soft Computing*, 16(1):135–151, 2012.

[20] Michael Chau and Hsinchun Chen. A machine learning approach to web page filtering using content and structure analysis. *Decision Support Systems*, 44(2):482–494, 2008.

[21] Christian Seifert, Ian Welch, Peter Komisarczuk, Chiraag Uday Aval, and Barbara Endicott-Popovsky. Identification of malicious web pages through analysis of underlying dns and web server relationships. In *LCN*, pages 935–941. Citeseer, 2008.

[22] Christian Seifert, Ian Welch, and Peter Komisarczuk. Identification of malicious web pages with static heuristics. In *Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian*, pages 91–96. IEEE, 2008.

[23] Davide Canali, Marco Cova, Giovanni Vigna, and Christopher Kruegel. Prophiler: a fast filter for the large-scale detection of malicious web pages. In *Proceedings of the 20th international conference on World wide web*, pages 197–206. ACM, 2011.

[24] Ahmed Abbasi, Fatemeh Zahedi, Siddharth Kaza, et al. Detecting fake medical web sites using recursive trust labeling. *ACM Transactions on Information Systems (TOIS)*, 30(4):22, 2012.

[25] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 985–990. IEEE, 2004.
[26] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.
[27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
[28] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.
[29] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.
[30] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.
[31] Hai Leong Chieu and Hwee Tou Ng. A maximum entropy approach to information extraction from semi-structured and free text. *AAAI/IAAI*, 2002:786–791, 2002.
[32] Alaa El-Halees. Arabic text classification using maximum entropy. *The Islamic University Journal (Series of Natural Studies and Engineering)*, 15(1):157–167, 2007.
[33] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.
[34] Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 477–485. Association for Computational Linguistics, 2009.
[35] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.
[36] Chih-Chung Chang and Chih-Jen Lin. Libsvm – a library for support vector machines. `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`, 2015.
[37] Machine Learning Group at National Taiwan University. Liblinear – a library for large linear classification. `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`, 2015.
[38] Yoshimasa Tsuruoka. A simple c++ library for maximum entropy classification v3.0. *Software available at http:// www.nactem. ac.uk/ tsuruoka/ maxent/*, 2006.
[39] Zhu Qin-Yu and Huang Guang-Bin. Basic elm algorithms. `http://www.ntu.edu.sg/home/egbhuang/elm\_codes.html`, 2004.
[40] PhishTank. Join the fight against phishing. `https://www.phishtank.com/developer\_info.php`, 2016.
[41] Alexa. Alexa top sites. `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`, 2016.
[42] Comodo Group. Creating trust online. `https://www.comodo.com/`, 2017.
[43] Cdric Champeau. Jlangdetect. `https://github.com/melix/jlangdetect`, 2014.