LookAhead: Augmenting Crowdsourced Website Reputation Systems with Predictive Modeling

Sourav Bhattacharya[1(✉)], Otto Huhta[2], and N. Asokan[2,3]

[1] Bell Laboratories, Dublin, Ireland
sourav.bhattacharya@bell-labs.com
[2] Aalto University, Espoo, Finland
otto.huhta@aalto.fi
[3] University of Helsinki, Helsinki, Finland
asokan@acm.org

Abstract. Unsafe websites consist of malicious as well as inappropriate sites, such as those hosting questionable or offensive content. Website reputation systems are intended to help ordinary users steer away from these unsafe sites. However, the process of assigning safety ratings for websites typically involves humans. Consequently it is time consuming, costly and not scalable. This has resulted in two major problems: (i) a significant proportion of the web space remains unrated and (ii) there is an unacceptable time lag before new websites are rated. In this paper, we show that by leveraging structural and content-based properties of websites, we can reliably and efficiently predict their safety ratings, thereby mitigating both problems. We demonstrate the effectiveness of our approach using four datasets of up to 90,000 websites. We use ratings from Web of Trust (WOT), a popular crowdsourced web reputation system, as ground truth. We propose a novel ensemble classification technique that makes opportunistic use of available structural and content properties of web pages to predict their eventual ratings in two dimensions used by WOT: trustworthiness and child safety. Ours is the first classification system to predict such subjective ratings. The same approach works equally well in identifying malicious websites. Across all datasets, our classification achieves average F_1-score in the 74–90 % range.

1 Introduction

Internet scammers set up various types of "unsafe" websites to lure their victims. These include *malicious* sites, intended for *phishing*, *drive-by-downloads* of malware and misusing private user data, as well as sites that are *inappropriate* in some sense, e.g., websites hosting offensive, objectionable, hateful or illegal content.

A variety of mechanisms have been developed for steering unsuspecting users away from unsafe websites. Popular browsers present interstitial security warnings when users attempt to navigate to a known malicious website [1]. Several anti-virus vendors maintain website reputation systems (e.g., TrustedSource[1]).

[1] http://www.trustedsource.org/

M. Conti et al. (Eds.): TRUST 2015, LNCS 9229, pp. 143–162, 2015.
DOI: 10.1007/978-3-319-22846-4_9

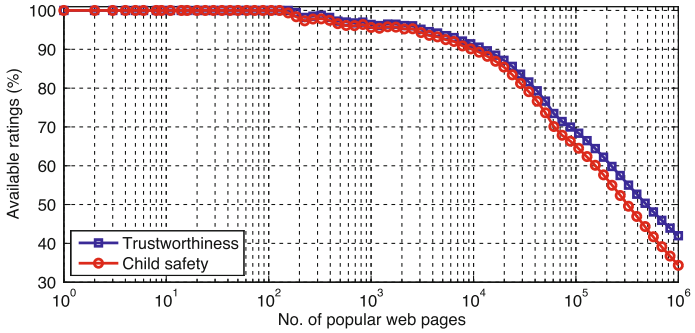


Fig. 1. Cumulative availability (%) of WOT ratings, trustworthiness and child safety, for the one million most popular webpages (as of July 2014).

These systems use a combination of machine learning techniques and manual expert evaluations to arrive at the rating for a given website. A popular subcategory of reputation systems use input ratings that are *crowdsourced* from the users of the system. *PhishTank*[2] and *Web of Trust* (WOT)[3] are examples of web reputation systems that rely fully or partly on crowdsourced ratings. An advantage of crowdsourced ratings is that the ratings can cover a broader class of unsafe websites, including those that are perceived to be inappropriate but not outright malicious [9].

All reputation systems, especially those that involve humans in the rating process, suffer from two major disadvantages: *insufficient coverage* and *time lag*. For example, Fig. 1, shows the cumulative availability of WOT reputation ratings (*trustworthiness* and *child-safety*) for one million most popular webpages (obtained from alexa.com) and indicates that the majority of the pages are unrated. The time gap between a new website coming online and the system assigning a rating can be often in the order of days to months.

A consequence of these drawbacks is that users, who rely on such reputation systems to protect them from unsafe websites, remain vulnerable when many unsafe websites are unrated. Although, machine learning techniques have been extensively used for detecting malicious websites based on the structure and content of web pages [8,10,27], in this work we address the following research question: *Can we reliably predict the eventual rating of an unrated website?*

We introduce *LookAhead*, a system that uses a combination of structural and content-based features to predict the eventual rating a website is likely to receive. In reality, the prediction task is non-trivial, as not all feature types are present on all webpages (see Sect. 4). To mitigate this feature unavailability problem, we propose an ensemble classification approach. We train different classifiers for each feature type and present different combination strategies to estimate the overall rating. For the structure of the websites, we consider HTML and JavaScript-based features. However, we show that structural features alone would not be

[2] http://www.phishtank.com/
[3] https://www.mywot.com/

sufficient for accurate predictions. Therefore, we introduce novel content-based feature sets, which are extracted from outgoing links with low ratings and the text present on a webpage. We make the following contributions:

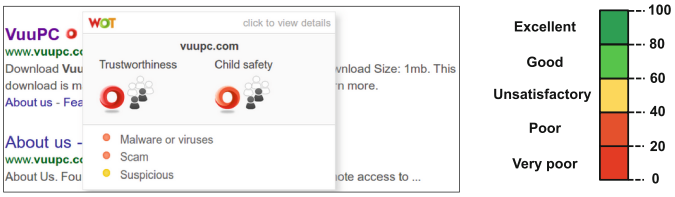
(i) Use of Content-Based Features for effectively predicting future ratings of websites. In particular, we propose a novel use of the *empirical cumulative distribution function* (ECDF) as a feature set to extract clues about the content of a web page based on ratings of outgoing hyperlinks in it (Sect. 4.2). We also propose how *topic modeling* techniques can be used to extract features that capture the theme of a webpage (Sect. 4.2). **(ii) LookAhead**, an adaptive ensemble classification technique that effectively combines several individual classifiers by learning combination weights from the data (Sect. 4.3). **(iii) Systematic Comparative Evaluation** of LookAhead on several datasets with up to 90,000 web pages (Sect. 5). We show that the performance can be improved significantly (statistically) when utilizing content-based features in addition to the structural features of web pages (Sect. 6). In particular, this holds across both subjective dimensions (trustworthiness and child safety), as well as maliciousness.

2 Related Work

A typical approach for helping users avoid malicious websites is to use blacklists of known bad websites. For example, Microsoft's Internet Explorer and Mozilla Firefox warn users when they try to visit a page present on a blacklist. Unfortunately, blacklists suffer from a number of shortcomings, e.g., they are required to be updated periodically, are often slow to reflect new malicious websites, and have poor coverage of malicious web space. To mitigate problems with blacklists, Felegyhazi et al. [16] propose a system that, given an initial blacklist of domains, tries to predict potentially malicious domains based on nameserver features and registration information. Prakash et al. [26] propose five different heuristics that allow synthesizing new URLs from existing ones. The authors use this idea to enlarge the existing blacklist of malicious URLs.

Going beyond blacklists, application of machine learning techniques to successfully identify malicious websites has become popular. Ma et al. [21] explore the use of lexical features, including the length and number of dots in URLs, host-based features, such as IP address, domain name and other data returned by WHOIS queries [13] to identify malicious web links. Another popular approach is to analyze the structural properties of webpages, especially looking for known malicious patterns within the embedded JavaScript, to identify malicious sites [10,12,15,20,27]. JSAND by Cova et al. [10] combines anomaly detection with emulation and uses a naive Bayes classifier for malicious web page and script detection. Cujo by Rieck et al. [27] considers both static and dynamic JavaScript features and classifies websites using Support Vector Machines (SVM). ZOZZLE by Curtsinger et al. [12] considers over 1.2 million JavaScript samples and achieves FPR and TPR in the range of 1.2–5.1 %.

Closest to our work is Prophiler by Canali et al. [8], which identifies malicious websites by considering only static features related to the URL and the



(a) (b)

Fig. 2. (a) WOT user interface showing aggregated user ratings. (b) WOT divides the reputation rating range into five (color-coded) levels.

structure of a page. For example, they consider 37 URL-based, 20 HTML and 26 JavaScript features and train three different classifiers, one for each feature type. While systems like Prophiler [8] and JSAND [10] report good results for detecting malicious websites, we consider a much broader and non-trivial problem of predicting subjective rating dimensions like trustworthiness and child-safety of a website. In addition, we consider not only the structure and URL of a web page but also the content presented on that page.

3 Web Reputation System WOT

WOT provides reputation ratings of the domain of a given URL in two dimensions, trustworthiness and child safety as integers in the range [0–100]. WOT builds the reputation ratings of a web domain mainly based on crowdsourced input ratings from a large user base and then applying a proprietary aggregation algorithm. It also uses input from other *trusted sources*, but the identities of these sources are not public. WOT has seen well over 100 million downloads. It is also used by large scale services like *Facebook* and *Mail.ru*. It is reasonable to assume that the user base of WOT and similar rating systems runs into tens of millions.

The front-end of WOT is a browser extension that scans the page being rendered in the browser for URLs, looks up their reputation ratings in the WOT back-end, and shows the results as color-coded glyphs. For example, Fig. 2(a) shows a red glyph next to a website deemed unsafe by WOT. The rating space is divided into five levels, with a color code assigned to each level, see Fig. 2(b). WOT's confidence in a rating is also indicated by a set of dark figurines (up to five, Fig. 2(a)).

Our objective is to see if we can use information found on a hitherto unrated web page to predict what rating it will receive. In this paper we use WOT as the target reputation system. However, our proposed method is generic and would work with any web reputation system. We therefore use existing WOT ratings as the ground truth, and apply a supervised learning-based algorithm for model building. Instead of building a regression model, we formulate the web page reputation prediction as a binary classification task [4]. We divide

reputation ratings into two (coarse) groups by applying a suitable threshold on the reputation ratings. This helps to minimize the effect of subjective variations among users in their ratings. Given a reputation rating $r \in [0, 100]$ of a URL, the class information of the URL is computed using the following simple rule:

$$class(r) = \begin{cases} bad & \text{if } r < \mathcal{T}_h \\ good & \text{otherwise} \end{cases} \quad (1)$$

In our experiments (Sect. 6) we present results for $\mathcal{T}_h = 40$. Results for $\mathcal{T}_h = 60$ can be found in Appendix A.5 of the full version of this paper [2].

4 LookAhead: Predicting Safety Ratings

Our predictive approach utilizes existing reputation ratings of a large number of webpages to learn a mapping function from various webpage features to a set of target classes, in our case, either *good* or *bad* (see Eq. 1). Figure 3 illustrates an overview of our web safety prediction approach combined with WOT. The LookAhead part, highlighted in the figure, is composed of a web crawler, a database, and a predictive model. The web crawler extracts various features from webpages and stores them, along with reputation ratings in the two WOT dimensions[4], to a database. The predictive model learns a classification model and uses it for predicting web safety of unrated URLs. We consider two types of features to represent websites: (i) *structural* features, which are extracted from the HTML and embedded JavaScript code, and (ii) *content* features that capture ratings of outgoing web links and the thematic structure of page text.

4.1 Structural Features of Web Pages

For structural features, we mainly rely on past research that has identified and successfully validated a large set of features (extracted from HTML and embedded JavaScript code) to identify malicious webpages. Specifically, we adopt the handcrafted and domain specific features used by Canali et al. for their Prophiler system [8]. In the evaluation section (see Sect. 5.2), we consider Prophiler as our main baseline algorithm.

HTML-Based Features: We adopt the same 20 HTML features[5] used by the Prophiler. Examples of the features include the number of *iframe* tags, the number of hidden elements, the number of script elements, the percentage of unknown tags, and the number of malicious patterns, e.g., presence of the *meta* tag [8].

JavaScript-Based Features: We use the same 24 JavaScript-based features used by the Prophiler, which are extracted by analyzing either the JavaScript

[4] WOT ratings are obtained using their web API (https://www.mywot.com/wiki/API).

[5] See [8] for an exhaustive and in-depth description of all the HTML features.

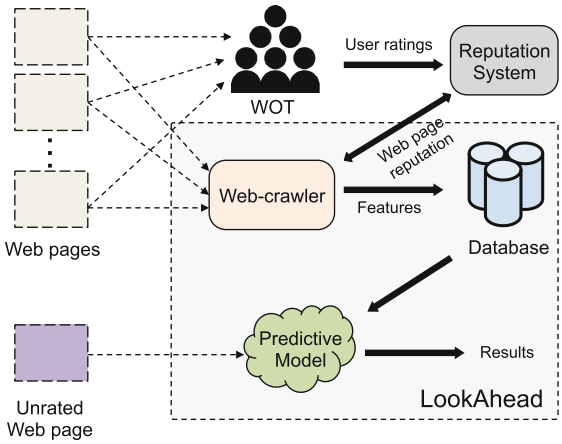


Fig. 3. An architectural overview of LookAhead in association with a crowdsourced web reputation system WOT.

file or the <script> element embedded within the HTML text. Examples of JavaScript-based features include the number of times the *eval()* function is used, the number of occurrence of the *setTimeout()* and *setInterval()* functions, the number of *DOM* modification functions, and the length of the script in characters [8].

4.2 Content Features of Web Pages

Contrary to the state-of-the-art approaches, in this paper we propose the use of a novel set of features based on (1) *empirical cumulative distribution function* (ECDF) of the reputation ratings of embedded outgoing links and (2) *topic modeling*. The main intuition behind using these features is that by learning (unsupervised) webpage content properties, we avoid the need for handcrafted features based on domain knowledge. In our evaluation, we show that the proposed novel features improve the recognition performance significantly (see Sect. 6).

Embedded Link-Based Features: To extract simple yet effective clues about the content of a web page, we hypothesize that the content of a page is related to the content of the pages it links to. In other words, we conjecture that the adage "*You are the company you keep*" is applicable here. This saying is based on the fact that often knowledge about an unknown person's friends provides some idea about the person's interests or personality. Similar ideas have been successfully applied in recommender systems [6] and in detecting susceptibility of mobile devices for malware infections [31].

Building on this idea, we propose a feature extraction scheme utilizing the available reputation ratings of embedded links. However, web pages may contain an arbitrary number of embedded links, ranging from none to several hundreds

or more. Moreover, the range of the reputation ratings can be arbitrary. Thus we need a feature representation scheme that can compactly represent an arbitrary number of outgoing links, while remaining robust in the face of arbitrary ranges of ratings.

ECDF-based feature extraction has been previously explored in the field of ubiquitous computing and mobile sensing to represent human motion characteristics from continuous accelerometer data streams [3,18,25]. However, the method has attracted very little attention outside the sensing domain. The simplicity and fast computation time of the ECDF features make it a viable option for using it in static web page analysis. Contrary to mobile sensing, in this paper we primarily focus on discrete reputation ratings.

More formally, let $R = \{r_1, r_2, \ldots, r_n\}$ denote the set of available reputation ratings of all the embedded web links on a page, where $r_i \in \mathbb{I}_{[0,100]}$, $\forall i \in \{1, \ldots, n\}$. The ECDF $\mathcal{P}_c(r)$ of R can be computed as:

$$\mathcal{P}_c(r) = p(X \leq r), \tag{2}$$

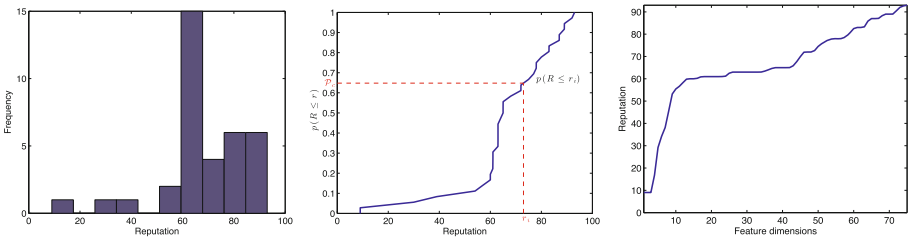
where, $p(X = r)$ is the probability of observing an embedded web link with a reputation rating of r, and X is a random variable that takes values from R (uniformly at random). For example, Fig. 4(a) shows an exemplary histogram of reputation ratings of web links found within a web page and Fig. 4(b) shows the corresponding ECDF computed using Eq. 2. Note that $\mathcal{P}_c(r)$ is defined on the entire range of the reputation ratings for embedded web links and is a monotonically increasing function.

Often the distribution of reputation ratings for embedded links is multimodal, e.g., as in our example shown in Fig. 4(a). In order to learn from such distributions, a recognition system should extract descriptors that relate to the shape and spatial position of the modes [18]. The shape of the distribution is captured as $\mathcal{P}_c$ increases from 0 to 1 (see Fig. 4(b)). To extract a feature vector $\boldsymbol{f} \in \mathbb{R}^k$ from the distribution, we first divide the range of $\mathcal{P}_c$, i.e., $[0, 1]$, into k equally sized bins with centers respectively at $[b_1, b_2, \ldots, b_k]$. The i^{th} feature component $f_i \in \mathbb{R}$ is then computed as:

$$f_i = \mathcal{P}_c^{-1}(b_i) \tag{3}$$

Thus the feature vector $\boldsymbol{f}$ accurately captures the shape and positions from the underlying probability function $p(r)$, while the ECDF $\mathcal{P}_c$ can be computed efficiently using Kaplan-Meier estimator [11]. For completeness, Fig. 4(c) shows the extracted ECDF-based feature vector for $k = 75$. The only parameter for the ECDF-based feature extraction method is the number of bins k, which controls the granularity with which the shape of the underlying distribution is captured. In our experiments we also append the mean of ratings in R as a feature value to the extracted ECDF feature vector.

Adversarial Implications: If ECDF features were based on all outgoing links, a malicious website may attempt to evade detection by embedding a large number of links to pages with high ratings. To deter such an attack, while constructing the set R (see above), we only allow ratings $r \leq C_r$, where C_r is the



(a) (b) (c)

Fig. 4. Exemplary illustration of (a) the distribution of reputation ratings, (b) their Empirical Cumulative Distribution Function, and (c) ECDF-based features.

critical rating threshold. The choice of C_r can be application dependent and ideally should be adapted based on the overall costs of making false negative predictions.

Topic Model-Based Features: To gain further insight into the type of content on a web page, we analyze the text in the page and extract a set of features that captures the summary of the text as a distribution over a set of predefined topics. A topic is defined as a probability distribution over a fixed set of words. In order to learn the topics in an unsupervised manner, we employ the well established *Latent Dirichlet Allocation* (LDA) model [5]. The main objective of LDA, or in general in any topic modeling algorithm, is to extract short descriptions of documents, while preserving statistical relationships that are useful, e.g., for document summarization and classification. In this work, we only focus on text in English. As a significant portion of the webpages in our evaluation dataset (see Sect. 5.1) is non-english we use Google translation APIs, as part of the web crawler, to convert text into english. To avoid translation errors, we use an english dictionary to validate words before they are included in the *vocabulary* set V used by the LDA model.

The main objective of the LDA model is to learn model parameters, such as K topics $\beta_{1:K}$, the topic proportions θ_d in the document d, and topic assignments $z_{n,d}$ of observed word w_n in document d from the corpus of webpages. A brief overview of the topic model and the definitions of the parameters are given in Appendix A.1 of the full version [2]. Once the LDA parameters are learned, given the set of words w present on a webpage and the topics $\beta_{1:K}$, the topic model-based feature set for the webpage is computed as: $p(\theta_d|w, \beta_{1:K})$, i.e., the estimated topic proportions.

Adversarial Implications: Similarly to the ECDF-based features, the topic model-based features can be exploited by an adversary. As the topic proportion term, i.e., $p(\theta_d|w, \beta_{1:K})$ captures the relative weight of various topics being described within the text w, an attacker can simply add random words that can boost the probability of certain topics. In Sect. 7 we propose a possible solution to prevent this attack.

4.3 Ensemble Classification

One challenge in the feature extraction procedures, described above, is that often one or more feature types are missing from a web page. For example, in reality, not all web pages use JavaScript, contain embedded outgoing links, or use textual descriptions, although the HTML features are always available. Thus, a new classification technique is required that is able to overcome the problem of feature unavailability. Existing approaches such as [8,21,29,30], do not address this problem and therefore have limited generalizability.

According to Bayesian theory [19], given HTML ($\boldsymbol{f}_H$), JavaScript ($\boldsymbol{f}_J$), ECDF ($\boldsymbol{f}_E$), and Topic ($\boldsymbol{f}_T$) feature vectors, a URL should be assigned to the class $c_j \in \{bad, good\}$, if the posterior probability for class c_j is maximum, i.e.

$$\begin{aligned} &assign \quad URL \rightarrow c_j \quad if \\ &p(c_j|\boldsymbol{f}_H, \boldsymbol{f}_J, \boldsymbol{f}_E, \boldsymbol{f}_T) = \max_i \; p(c_i|\boldsymbol{f}_H, \boldsymbol{f}_J, \boldsymbol{f}_E, \boldsymbol{f}_T) \end{aligned} \tag{4}$$

The computation of $p(c_j|\boldsymbol{f}_H, \boldsymbol{f}_J, \boldsymbol{f}_E, \boldsymbol{f}_T)$ depends on the joint probability functions (likelihood) $p(\boldsymbol{f}_H, \boldsymbol{f}_J, \boldsymbol{f}_E, \boldsymbol{f}_T|c_j)$ and the prior probability $p(c_j)$, i.e.:

$$p(c_j|\boldsymbol{f}_H, \boldsymbol{f}_J, \boldsymbol{f}_E, \boldsymbol{f}_T) \propto p(\boldsymbol{f}_H, \boldsymbol{f}_J, \boldsymbol{f}_E, \boldsymbol{f}_T|c_j)\, p(c_j) \tag{5}$$

The likelihoods are difficult to infer when one or more features are unavailable. The likelihood computation can be simplified by combining decision support of individual classifiers on different feature types [19]. Accordingly, we train four classifiers $\mathcal{C}_H$, $\mathcal{C}_J$, $\mathcal{C}_E$, and $\mathcal{C}_T$ using valid $\boldsymbol{f}_H$, $\boldsymbol{f}_J$, $\boldsymbol{f}_E$, and $\boldsymbol{f}_T$ features respectively, where each classifier returns a posterior probability distribution over the *bad* and *good* classes. However during prediction, if a feature type is unavailable, we do not include the corresponding classifier while computing the overall posterior probabilities.

A number of strategies can be adopted to combine the posterior probabilities of the classifiers to generate the overall belief. In this paper we propose a linear combination rule that determines the combination weights of individual classifiers using the Fukunaga class separability score [17]. Our adaptive weight selection method is based on the intuition that a classifier should be given more importance if it is easy to separate among the *bad* and *good* classes in the corresponding feature space. See Appendix A.2 of the full version [2] for the definition of class separability we use and other popular combination rules. For each classifier, we compute the separability score after correlation based feature subset selection. The separability scores, after normalization, are then used as the respective weight w_k for the classifier $\mathcal{C}_k$. The final belief of the class c_j is estimated as:

$$p^*(c_j|\boldsymbol{f}_H, \boldsymbol{f}_J, \boldsymbol{f}_E, \boldsymbol{f}_T) = \sum_{k \in \{H,J,E,T\}} w_k \; p(c_j|\boldsymbol{f}_k) \tag{6}$$

The final predicted class c_j is inferred by applying the decision rule given in Eq. 4 using the computed belief above. Figure 5 shows the data adaptive ensemble classification technique used by LookAhead.

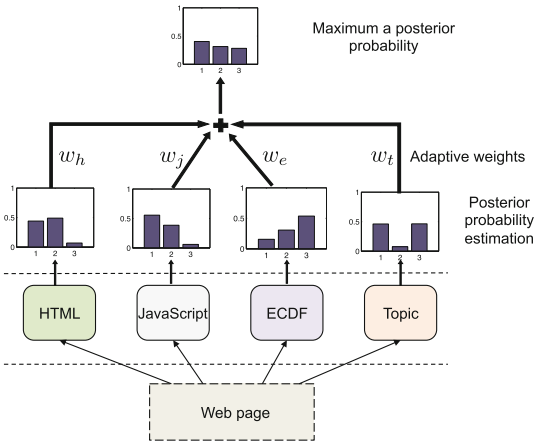


Fig. 5. Overview of the ensemble classification approach used by LookAhead.

5 Experimental Settings

5.1 Datasets

To perform an extensive and systematic study, we generated a pool of over 140,000 URLs and obtained their reputation ratings in both dimensions using the WOT API. Out of these, 80,000 URLs have positive reputation ratings, and 60,000 have negative ratings. For each URL we crawl the web page to extract HTML, JavaScript, ECDF and topic model features where available. Figure 6 illustrates the histograms of reputation ratings for all webpages in our dataset. The dataset, where at least HTML features and WOT ratings are available, is referred to as the *opportunistic* dataset. Out of 140,000 URLs, 89,220 web pages have trustworthiness ratings, and 84,714 have ratings for child safety. However, the number drops to 31,995, in case of trustworthiness, and to 38,118 for child safety, when validity of all feature types are considered (for $\boldsymbol{\mathcal{T}_h} = 40$). We refer to this second dataset as the *all-valid* dataset. The significant drop in the size of the all-valid dataset further highlights that feature unavailability is intrinsic to web data analysis.

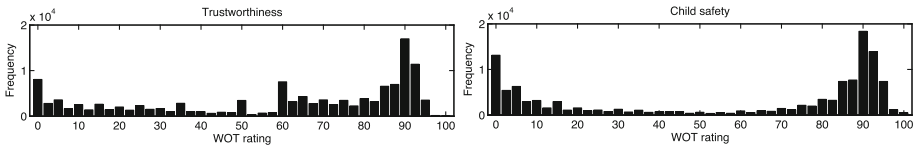


Fig. 6. Histograms of all webpages in our dataset in two reputation dimensions: trustworthiness (left) and child safety (right).

Existing research primarily focused on detecting if a webpage is malicious. However, the malware dataset used in [8] is no longer available, which makes exact replication of Prophiler results difficult. "Trustworthiness" in WOT does not directly correspond to malware. In addition to the reputation ratings, WOT provides category information, such as 'malware', 'scam', 'suspicious' and 'good site', of websites based on votes from users and third parties. From the all-valid dataset we generate a malware dataset consisting of 2,784 webpages that were categorized by WOT as 'malware or virus'. To generate a dataset containing both malware and benign webpages, we include an equal number of webpages that got very high trustworthiness ratings and have all feature types. We refer to this dataset as the *malware* dataset.

Lastly, we construct another dataset by considering only the URLs that fall either in the top most or the bottom most trustworthiness rating categories, see Fig. 2(b) for definitions of various rating categories used by WOT. As with malware dataset, we only consider webpages for which all feature types are available. Our *two-category* dataset consists of 10,118 sites with very poor ratings and 13,539 with excellent ratings.

5.2 Baseline Algorithms

In our experiments, we report comparison results against Prophier [8]. Prophiler relies on HTML, JavaScript, and URL/HOST features to detect if a webpage is malicious. However, it uses APIs to a proprietary WHOIS [14] system and uses a private database for blacklisted URLs to extract URL/HOST features. Neither of these are available openly, which makes the corresponding URL/HOST feature vectors invalid for our datasets. API inaccessibility and unavailability of suitable blacklisted database covering WOT URLs used in our dataset makes the majority of the URL/HOST feature vectors in our experiments invalid. Consequently, we do not use URL/HOST features in our ensemble classification system. Note that it is very easy to incorporate additional feature types in our classification system, e.g., training a classifier $\mathcal{C}$ using the new feature type and then considering the posterior probabilities in Eq. 6. Contrary to our approach, i.e., assigning data driven weighting of classifiers to compute the final belief (see Sect. 4.3), Prophiler uses the 'OR' combination rule (see Appendix A.3 of [2]). We systematically compare the performance of LookAhead with the ensemble classification methodology considering different subsets of feature types.

5.3 Evaluation Metric

We use 10-fold stratified random cross validations when presenting classification performance for all the approaches. As the primary performance metric, we use Avg. F_1-score, False Negative Rate (FNR), and False Positive Rate (FPR). The definitions of all the evaluation metrics can be found in Appendix A.4 of [2].

6 Evaluation

We begin our evaluation by first considering classification performance on the all-valid dataset using Random Forest as the basic classifier[6]. Note that, all URLs considered within this dataset have valid HTML (H), JavaScript (J), ECDF (E) and Topic-based (T) features. This dataset allows us to systematically study the influence of various feature combinations on the overall classification performance of LookAhead. Table 1 summarizes the performance of LookAhead in both reputation dimensions with the parametric settings $\mathcal{T}_h = 40$ (see Eq. 1), and $C_r = 40$ (see Sect. 4.2). The table also includes the performance of Prophiler.

For trustworthiness, LookAhead achieves the highest Avg. F_1-score of 81.3 %, when all feature types are considered (highlighted in gray), at the same time achieving the lowest FNR (19 %) and FPR (18.3 %). Similarly for child safety, LookAhead with all feature types achieves the best performance (86.4 %), lowest FNR (11.6 %) and lowest FPR (16.2 %). In both reputation dimensions, the performance using all features, is significantly better (statistically) than all other feature combinations, i.e., $p \ll 0.01$ in McNemar χ^2 test with Yates' correction [22].

Prophiler shows a statistically weaker classification performance in both reputation dimensions compared to LookAhead (employing all feature types). However, it achieves a better FNR in prediction than LookAhead. This is due to the use of a conservative 'OR' classifier combination rule (see full version [2]) that is more likely to report a URL as bad. This higher likelihood of predicting web pages as bad improves the overall recall of the bad class, which consequently pulls down the FNR for Prophiler, however, at the expense of a higher FPR. Prophiler focuses solely on reducing FNR. In contrast, in use cases where overall usability in prediction is important, both FNR and FPR should be reduced. For example, in predicting safety ratings, a low FPR is also needed to avoid showing frequent warnings to users for actually good websites.

In reality, not all feature types are available for all URLs. To evaluate the performance of LookAhead under real life situations we next present results on the opportunistic dataset. In these experiments, we only present the performance of LookAhead while considering all available feature types. Moreover, we study the performance of various classifier combination rules and present the results in Table 2 for both reputation dimensions. In contrast to the all-valid dataset, $\mathcal{T}_h = 40$, generates a high degree of class imbalance in our opportunistic dataset (see Fig. 6). During the training phase the prevalence of one class affects the process of learning, and the learned classifier is often biased towards the over-represented class [23]. To mitigate class imbalances during training, we also report experimental results when a simple class balancing approach, i.e., reducing data from the prevalent class, is applied during classifier training. The data driven, adaptive classification combination rule of LookAhead generates the best classification performance, with a notable exception in the case of unbalanced dataset for trustworthiness, where the 'Product' rule achieves the highest Avg. F_1-Score.

[6] We also experimented using linear-SVM, SVM, KNN and C4.5 classifiers, and chose Random Forest for its superior performance.

Table 1. Performance of LookAhead (under various feature combinations H = HTML, J = JavaScript, E = ECDF, and T = Topic) and Prophiler on the all-valid dataset ($\mathcal{T}_h = 40$, $C_r = 40$, and **: Statistically significant with 99 % confidence).

All-valid dataset size: 31,995 URLs
Reputation dimension: Trustworthiness

Feature sets				Avg. F_1-Score	FNR	FPR
H	J	E	T	(%)	(%)	(%)
✓				75.6 **	25.2	23.7
	✓			74.3 **	25.6	25.9
		✓		66.9 **	33.5	32.7
			✓	74.5 **	26.8	24.3
✓	✓			76.9 **	23.3	22.8
✓		✓		77.3 **	23.7	21.7
✓			✓	78.5 **	21.8	21.3
	✓	✓		72.1 **	28.9	26.9
	✓		✓	77.1 **	24.0	21.9
		✓	✓	77.5 **	23.9	21.2
✓	✓	✓		78.8 **	21.6	20.7
✓	✓		✓	79.5 **	20.9	20.1
✓		✓	✓	80.4 **	20.2	19.0
	✓	✓	✓	79.6 **	21.4	19.4
✓	✓	✓	✓	**81.3**	**19.0**	**18.3**
Prophiler				74.5 **	14.2	35.9

All-valid dataset size: 38,118 URLs
Reputation dimension: Child safety

Feature sets				Avg. F_1-Score	FNR	FPR
H	J	E	T	(%)	(%)	(%)
✓				80.3 **	15.2	25.8
	✓			79.6 **	15.8	26.8
		✓		73.1 **	22.0	33.6
			✓	81.9 **	17.0	19.8
✓	✓			81.1 **	14.3	25.1
✓		✓		79.4 **	16.3	26.3
✓			✓	84.5 **	13.4	18.3
	✓	✓		77.3 **	18.1	29.1
	✓		✓	83.9 **	14.6	18.2
		✓	✓	83.9 **	14.9	17.7
✓	✓	✓		82.4 **	13.5	23.2
✓	✓		✓	85.2 **	12.2	18.4
✓		✓	✓	85.7 **	12.6	16.7
	✓	✓	✓	85.3 **	13.4	16.6
✓	✓	✓	✓	**86.4**	**11.6**	**16.2**
Prophiler				79.5 **	9.6	34.5

Table 2. Performance of LookAhead on the opportunistic dataset under various classifier combination rules ($\mathcal{T}_h = 40$ and $C_r = 40$, **: Significant with 99 % confidence, *: 95 % confidence).

Opportunistic dataset size: 89,220 URLs
Reputation dimension: Trustworthiness

Experiment		Avg. F_1-Score	FNR	FPR
Comb. Rule	Bal.	(%)	(%)	(%)
Adaptive		78.0	56.4	4.1
Sum		77.8	57.9	**3.5**
Product		**78.9** **	53.3	4.6
Or		32.4 **	**10.1**	84.2
Voting		71.4 **	38.8	25.2
Prophiler*		72.6 **	45.3	19.8
Adaptive	✓	**74.0**	22.3	29.1
Sum	✓	74.0	22.3	**29.0**
Product	✓	73.6 **	22.8	29.5
Or	✓	27.3 **	**2.3**	89.8
Voting	✓	57.3 **	11.1	57.0
Prophiler*	✓	62.0 **	14.4	49.8

Opportunistic dataset size: 84,714 URLs
Reputation dimension: Child safety

Experiment		Avg. F_1-Score	FNR	FPR
Comb. Rule	Bal.	(%)	(%)	(%)
Adaptive		**83.7**	29.8	7.2
Sum		83.4 *	31.2	**6.6**
Product		83.6	29.6	7.5
Or		40.7 **	**4.7**	82.3
Voting		73.9 **	19.7	30.7
Prophiler*		73.9 **	26.7	26.2
Adaptive	✓	**81.5**	25.3	**14.0**
Sum	✓	81.1 **	22.9	16.4
Product	✓	80.9 **	23.1	16.8
Or	✓	40.0 **	**3.0**	83.4
Voting	✓	68.1 **	12.3	44.3
Prophiler*	✓	69.4 **	15.6	40.5

Table 3. Performance of LookAhead and Prophiler on the malware and two-category datasets (**: Significant with 99 % confidence).

Malware dataset size: 5,568 URLs

Feature sets				Avg. F_1-Score	FNR	FPR
H	J	E	T	(%)	(%)	(%)
✓	✓	✓	✓	**89.0**	**10.3**	**11.6**
Prophiler				80.7 **	11.1	27.3

Tow-category dataset size: 23,657 URLs

Feature sets				Avg. F_1-Score	FNR	FPR
H	J	E	T	(%)	(%)	(%)
✓	✓	✓	✓	**89.8**	**13.9**	**7.4**
Prophiler				79.3 **	16.2	24.3

Prophiler has previously been shown to perform well in detecting malicious websites. To show how LookAhead (with all features) perform in such scenarios, we repeated the experiments on the malware and two-category datasets and present the results in Table 3. LookAhead achieves average F_1-scores of 89 % for the malware dataset and 89.8 % for the two-category dataset, which are significantly better ($p \ll 0.01$) than Prophiler's performance of 80.7 % for the malware dataset and 79.3 % for the two-category dataset. LookAhead also generates better FNR and FPR than Prophiler on both datasets.

7 Discussion

7.1 Feature Importance in Reputation Prediction

Our results show that the structural and content related properties of a website can be effectively used to predict not only its maliciousness, but also the more challenging properties of trustworthiness and child safety. In order to understand the overall classification results, we study the importance of individual features as computed by a Random Forest classifier[7]. In Fig. 7 we plot the average importance for all (120) features used in this work when training a Random Forest classifier (using 100 trees) on the all-valid dataset. The higher the value, the more important is the feature. Figure 7 further highlights that different features are assigned different relative importances, while separating good websites from bad ones in each reputation dimension.

Interestingly, the importance scores of the HTML and JavaScript-based features look very similar for both trustworthiness and child safety predictions. The most important features, shown by the dotted region A in the figure, are related to script tags in HTML, direct assignments in JavaScript, and the total character count in both. Although, a few structural (i.e., HTML and JavaScript) features are found to be important, a majority of them have little or no significance. Contrary to the structural features, ECDF features show significant differences in importance scores for the two reputation dimensions. For trustworthiness, low ratings of the embedded links (region B) play an important role in prediction. In child safety, the mean value of the embedded ratings (region C) plays a significant role also. For trustworthiness, the three most important topics (region E)

[7] Feature importance is defined as the total decrease in node impurity averaged over all the trees [7].

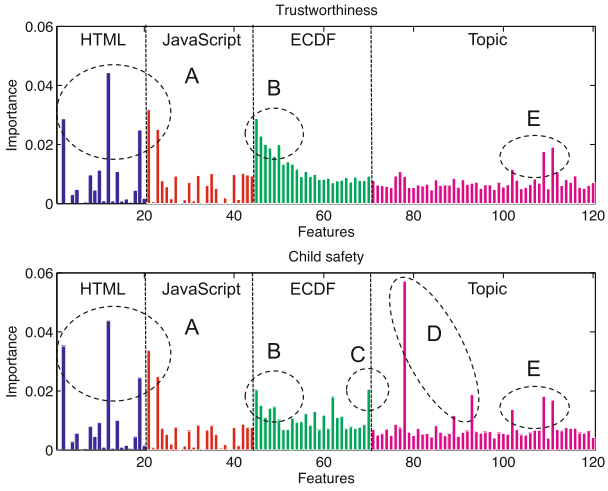


Fig. 7. Importance of individual features, while predicting trustworthiness and child safety, computed by the Random Forest classifier on the all-valid dataset.

are related to money-making, news, and weather. Among the rest of the topic features, none are significantly better or worse than the others. For child safety prediction there are three other topics (region D) that play a significant role and as expected, these topics correspond to adult content.

Although, we use the same feature set for predicting both reputation dimensions, the feature selection inherent to the Random Forest classifier learns very different mapping functions for each prediction task. Figure 7 provides evidence that our proposed ECDF and Topic-based features contribute consistently in predicting subjective ratings.

7.2 Tuning of Prediction Performance

Predictive performance of LookAhead can be primarily influenced by a number of factors: (i) the type of features considered (e.g., HTML and ECDF), (ii) the type of classifier used (e.g., Random Forest and SVM), (iii) strategies used to overcome class imbalances in the training data, and (iv) the combination rule used for computing the final posterior probability (e.g., Adaptive and Sum rule). Often, once the prediction pipeline is deployed, the factors (i)–(iii) are kept constant, as they are time consuming to re-build. However, the classifier combination strategy can be adapted in real time to control the overall performance of the LookAhead system. Based on the requirements, the system administrator can focus more on lowering the overall FNR by using the 'OR' combination rule, e.g., while predicting child safety a very low FNR is expected for parental filtering systems. As evident from Table 2, often emphasizing FNR inflates FPR. Our LookAhead system demonstrates a good balance of both FNR and FPR.

Table 4. Detection rates for various classifiers settings.

Dataset	System	FNR (%)	FPR (%)	$\mathcal{D}_r$ (%)
All-valid, Trustworthiness	LookAhead	19.0	18.3	52.5
All-valid, Trustworthiness	Prophiler	14.2	35.9	37.4
All-valid, Child safety	LookAhead	11.6	16.2	57.5
All-valid, Child safety	Prophiler	9.6	34.5	39.6
Opportunistic, Trustworthiness	LookAhead	22.3	29.1	40.0
Opportunistic, Trustworthiness	Prophiler	14.4	49.8	30.1
Opportunistic, Child safety	LookAhead	25.3	14.0	57.2
Opportunistic, Child safety	Prophiler	15.6	40.5	34.3

7.3 Detection Rate

When considering the implications of our results, in addition to the FNR and FPR, the proportion of good and bad websites in the wild should also be taken into account. In reality, this so-called base rate B_r, is biased towards good websites. Thus we look at the detection rate for bad websites, i.e., what percentage of web pages that our classifier predicts as bad are truly bad. From WOT statistics [28], we see that roughly 20 % of websites that have a rating are dangerous regarding either trustworthiness or child safety. We use this number as our estimate for B_r, and compute the detection rate as:

$$\mathcal{D}_r = \frac{(1 - FNR) \cdot B_r}{(1 - FNR) \cdot B_r + FPR \cdot (1 - B_r)}, \tag{7}$$

Table 4 presents detection rates of LookAhead and Prophiler on all-valid and opportunistic datasets for both reputation dimensions. We can see that due to the biased base rate, the detection rates are in the range of $30-40\%$ for Prophiler and $40 - 57\%$ for LookAhead, indicating better classification performances of LookAhead. For example, in case of a warning system for users, when all features are present, 52.5 % of possible warnings for untrustworthy web pages would be correct for LookAhead. The corresponding detection rate of 37.4 % for Prophiler is significantly lower. The results highlight that, while in general the problem of predicting reputation ratings is challenging, considering content-based features significantly improves the detection rate.

7.4 Applications

Fast-Tracking Publication of Ratings: Crowdsourced reputation rating services like WOT do not announce a rating for a web site until they have enough input ratings to reach a sufficient level of confidence. If a partially accumulated rating (that has not reached a sufficient level of confidence) matches the rating predicted by our classifier, the reputation service may choose to fast-track the publication of the rating.

Table 5. Time analysis for extracting various feature types.

Feature type	Average extraction time
HTML + JavaScript	3.1 s / link
ECDF	1.9 s / link
Topic + translation	3.4 s / link
Topic + without translation	1.3 s / link

Intermediate User Feedback: If a user attempts to navigate to an unrated page that is predicted by our classifier to have a potentially bad rating, the browser extension can warn the user accordingly. Earlier research [24] raised concerns about the usefulness of crowdsourcing for security and privacy applications. Nevertheless, given the popularity of systems like WOT, we argue that a tool like LookAhead is essential for the security of users who have chosen to rely on such systems. Also, note that although our analysis was done with WOT as the target rating system, the methodology is applicable to any website safety rating system, whether crowdsourced or expert-rated.

7.5 Performance Considerations

We summarize the performance of our various feature extraction techniques and report the average measured running time needed for computing them. For the purpose of computing the average extraction time we randomly selected 1,000 URLs from our dataset and measured the time required to extract different classes of features on a standard Linux desktop computer (8 Gb RAM, 2.4 GHz processor). In case of Topic model features we also recorded the time for performing translation of non-english web pages. Table 5 summarizes the time analysis of our feature extraction methods. The time of 3.1 s that LookAhead needs for extracting structural features is comparable to that of 3.06 s reported by Prophiler. When including the content based features, in total, LookAhead needs 6.3 s to extract all features from an English-language web page (and 8.4 s if translation is needed). Moreover, caching and pre-fetching of features can be employed to further reduce the feature extraction time.

7.6 Limitations

Perhaps the most significant limitation of any system using machine learning to detect bad websites is the potential for adversaries to manipulate the system: either by modifying their website to avoid detection or by manipulating the classifier itself. While the use of the ECDF-function protects against manipulation of outgoing links, as we pointed out in Sect. 4.2, the simplistic approach of using topic modeling is vulnerable to an attacker who attempts to influence the inferred topic model for a page he controls. Instead of directly using the probability distribution of topics as we do in Sect. 4.2, we could convert to a boolean

vector (indicating if the topic is present on the page). Such an approach will reduce false negatives (since an attacker can no longer gain by adding text to his page to make it appear to belong to an innocuous topic as the dominant topic), but will also raise false positives. We are currently investigating this avenue.

Another limitation is that, although the performance of LookAhead is comparable to previous solutions, real time use will require further speedup. One option here is to use server-side assisted feature extraction. Finally, an open question is how the use of predicted ratings will influence the actual rating. For example, if the predicted rating is used for intermediate user feedback as suggested above, it might sway future input ratings from the crowd towards the predicted rating.

7.7 Current Work

We are conducting a longitudinal study on a large number of websites that do not yet have a WOT rating. We plan to see (a) how well our predictions match those websites that do eventually get a rating and (b) how do our predictions as well as the actual ratings evolve over time.

Acknowledgments. This work was partially supported by the Intel Institute for Collaborative Research in Secure Computing (ICRI-SC) and the Academy of Finland project "Contextual Security" (Grant Number: 274951). We thank Web of Trust for giving access to their data which we used in this work. We also thank Timo Ala-Kleemola and Sergey Andryukhin for helping us understand the WOT data, Jian Liu and Swapnil Udar for helping to develop the web crawler. We would also like to thank Petteri Nurmi, Pekka Parviainen, and Nidhi Gupta for their feedback on an earlier version of this manuscript.

References

1. Akhawe, D., Felt, A.P.: Alice in warningland: a large-scale field study of browser security warning effectiveness. In: Proceedings of the 22Nd USENIX Conference on Security, SEC 2013, pp. 257–272. USENIX Association, Berkeley, CA, USA (2013)
2. Bhattacharya, S., Huhta, O., Asokan, N.: Lookahead: augmenting crowdsourced website reputation systems with predictive modeling (2015). http://www.arxiv.org/pdf/1504.04730.pdf
3. Bhattacharya, S., Nurmi, P., Hammerla, N., Plötz, T.: Using unlabeled data in a sparse-coding framework for human activity recognition. Pervasive and Mobile Computing, May 2014
4. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2007)
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
6. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52 (1998)
7. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)

8. Canali, D., Cova, M., Vigna, G., Kruegel, C.: Prophiler: a fast filter for the large-scale detection of malicious web pages. In: Proceedings of the 20th International Conference on World Wide Web, pp. 197–206. ACM (2011)
9. Chia, P.H., Knapskog, S.J.: Re-evaluating the wisdom of crowds in assessing web security. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 299–314. Springer, Heidelberg (2012)
10. Cova, M., Kruegel, C., Vigna, G.: Detection and analysis of drive-by-download attacks and malicious javascript code. In: Proceedings of the 19th International Conference on World Wide Web, pp. 281–290. ACM (2010)
11. Cox, D.R., Oakes, D.: Analysis of Survival Data. Champman and Hall, CRC (1984)
12. Curtsinger, C., Livshits, B., Zorn, B.G., Seifert, C.: Zozzle: fast and precise in-browser javascript malware detection. In: USENIX Security Symposium, pp. 33–48 (2011)
13. Daigle, L.: Whois protocol specification
14. Daigle, L.: Rfc 3912: Whois protocol specification, September 2014. http://www.tools.ietf.org/html/rfc3912
15. Feinstein, B., Peck, D.: Caffeine monkey: automated collection, detection and analysis of malicious javascript. In: Proceedings of the Black Hat Security Conference, **2007** (2007)
16. Felegyhazi, M., Kreibich, C., Paxson, V.: On the potential of proactive domain blacklisting. In: Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More, LEET 2010, p. 6. USENIX Association, Berkeley, CA, USA (2010)
17. Fukunaga, K.: Introduction to Statistical Pattern Recognition, 2nd edn. Academic Press, San Diego (1990)
18. Hammerla, N., Kirkham, R., Andras, P., Plötz, T.: On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution. In: Proceeding of International Symposium on Wearable Computers (ISWC) (2013)
19. Kittler, J., Hatef, M., Duin, R., Matas, J.: On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. **20**(3), 226–239 (1998)
20. Likarish, P., Jung, E., Jo, I.: Obfuscated malicious javascript detection using classification techniques. In: 4th International Conference on Malicious and Unwanted Software (MALWARE), pp. 47–54 (2009)
21. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 1245–1254. ACM, New York, NY, USA (2009)
22. McNemar, Q.: Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika **12**, 153–157 (1947)
23. Menardi, G., Torelli, N.: Training and assessing classification rules with imbalanced data. Data Min. Knowl. Disc. **28**(1), 92–122 (2014)
24. Moore, T., Clayton, R.C.: Evaluating the wisdom of crowds in assessing phishing websites. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 16–30. Springer, Heidelberg (2008)
25. Plötz, T., Hammerla, N.Y., Olivier, P.: Feature learning for activity recognition in ubiquitous computing. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 1729–1734 (2011)
26. Prakash, P., Kumar, M., Kompella, R., Gupta, M.: Phishnet: predictive blacklisting to detect phishing attacks. In: 2010 Proceedings IEEE INFOCOM, pp. 1–5, March 2010

27. Rieck, K., Krueger, T., Dewald, A.: Cujo: efficient detection and prevention of drive-by-download attacks. In: Proceedings of the 26th Annual Computer Security Applications Conference, pp. 31–39. ACM (2010)
28. Ruvolo, J.: WOT statistics, December 2014. https://www.mywot.com/en/community/statistics
29. Seifert, C., Welch, I., Komisarczuk, P.: Identification of malicious web pages with static heuristics. In: Telecommunication Networks and Applications Conference (ATNAC), pp. 91–96 (2008)
30. Seifert, C., Welch, I., Komisarczuk, P., Aval, C., Popovsky, B.: Identification of malicious web pages through analysis of underlying dns and web server relationships. In: 33rd IEEE Conference on Local Computer Networks (LCN), pp. 935–941 (2008)
31. Truong, H.T.T., Lagerspetz, E., Nurmi, P., Oliner, A.J., Tarkoma, S., Asokan, N., Bhattacharya, S.: The company you keep: mobile malware infection rates and inexpensive risk indicators. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 39–50 (2014)