Website Navigation Behavior Analysis for Bot Detection

Rabih Haidar
Office of Information Technology
American University of Beirut
Beirut, Lebanon
Email: rh131@aub.edu.lb

Shady Elbassuoni
Department of Computer Science
American University of Beirut
Beirut, Lebanon
Email: se58@aub.edu.lb

***Abstract*—Detecting bots is an important goal for most website admins. In this paper, we propose a novel machine learning bot detection approach based on local website navigation behavior. While machine learning has been used before for bot detection, most existing approaches rely on general hypotheses based on statistical analysis over multiple websites and are thus easy to counter. In our work, we build a website-specific hypothesis or classifier based on the actual navigation data of the website. The advantages of our approach is that it can be generally used to detect any type of bots and is difficult to counter unless website-specific bots are designed as well. Our classifier uses a Two-Class Boosted Decision Tree classification model and can be periodically re-trained to learn new hypotheses as bots evolve. We tested our approach on two real-world websites and achieved an accuracy of around 83%, outperforming the state-of-the-art machine-learning-based bot detection techniques by almost 14%. We also show that our approach can successfully distinguish between various classes of bots and we show how it can be deployed as a real-world application by any website to automatically detect bots as they navigate the website.**

I. Introduction

Internet robots, or bots for short, are software applications that trigger automated scripts over the Internet. They perform repetitive tasks that come at a much higher rate than humans would possibly perform, usually for malicious purposes. Examples of such malicious bots include website scrapers which download entire contents of websites without permission from the website owners, spam bots that collect personal information from websites such as email addresses and phone numbers or spread links and advertisements. Other examples include trading bots that acquire best bargains on trading websites or online shopping retailers, promotion bots which promote content online such as videos or posts, and even denial of service (DoS) bots which aim to bring down an entire website. All these aforementioned bot attacks have a negative impact on many businesses today. For example, a massive DoS attack has just recently caused an outage of many popular websites including Twitter, Spotify, and Airbnb [1].

Today, more than half of the Web traffic can be attributed to bots, according to the latest bot traffic report by the security firm Imperva [2]. This has been also confirmed in our datasets, where over 87% of web sessions, recorded during a period of one week for two major websites, were generated by bots. These bots range from malicious bots associated with viruses to web crawlers such as search engine bots. Such web crawlers are generally deemed useful as they enable websites to be indexed by major search engines. However, they also consume resources on the websites they visit, often visit websites without their approval, and might crawl data that the website owners do not wish to be crawled. Hence, website admins should be able to decide what gets crawled and by who.

Automatically detecting bots, whether malicious or web crawlers, is thus an important goal for most website owners and admins. Various techniques have already been proposed and are utilized to provide websites with defense mechanisms against bot attacks. These methods are either based on IP address and domain look-up [2] or human verification-tests such as CAPTCHA [8]. In the former, a database of IP addresses and domains from which bot attacks are generated is used, which is typically constructed by manually or semi-automatically inspecting website navigation logs to identify malicious behavior indicative of bots. However, such look-up databases must be constantly maintained as bots tend to use different IP addresses in order to bypass such method of detection [1]. On the other hand, human-verification tests involve asking users to verify certain information such as text embedded in an image or an audio transcription to prove they are not robots. This imposes a burden on genuine users who might have to constantly pass these verifications tests [6]. This is driving many major websites such as Google away from such approaches [3]. In addition, as new advances in information extraction and machine learning techniques are achieved, bots are getting more effective at passing these common human-verification tests with high accuracy [3], [8].

In this paper, we propose to use a machine learning approach to detect bots. Machine learning has been previously used for the task of bot detection. However, most of these approaches rely on generic features such as the average number of hits per website, DNS traffic, start time of the crawling session, the average navigation time, and the fre-

[1]https://techcrunch.com/2016/10/21/many-sites-including-twitter-and-spotify-suffering-outage/

[2]https://www.incapsula.com/blog/bot-traffic-report-2016.html

[3]http://www.bbc.co.uk/news/technology-39231307/

978-1-5090-5004-8/17 $31.00 © 2017 IEEE
DOI 10.1109/DSAA.2017.13

quency of IP attempts [1], [7], [10]. While these approaches produce very general hypotheses that can be used by multiple websites, they can be easily bypassed by bots as they evolve to avoid detection, which is a well-known challenge that faces spam detection approaches in general. On the other hand, our approach is website specific in the sense that it learns a different hypothesis for each website based on the website's web navigation behavior. In particular, we build a classifier for each website that is based on features extracted from the website's web sessions such as the pages hit, the order of navigation, the count of hits per each page, in addition to aggregate features such as the average number of page hits, average navigation time and so on. The advantage of using such a website-specific approach is that it takes into consideration the unique nature of each website and how it is navigated by genuine human users and is thus much more difficult to counter unless bot designers build specific bots for each website as well. To be able to do this, bot developers will need to have access to the web log of the website, which should not be publicly available, in order to mimic humans as they navigate the website. Our approach can be also seamlessly integrated with other bot detection methods and is thus complementary to existing techniques. For instance, our approach can be used to automatically maintain a database of IP addresses from which bot attacks are generated. It can also be used to flag suspicious website navigators who are in turn asked to pass a verification test to prove they are not bots.

We evaluated our approach on two real-world websites, namely wheelers[4] and whereleb [5], and achieved an accuracy of around 83% in bot detection. We also compared our approach to state-of-the-art machine learning approaches[1], [4], [5], [7], [11] which rely on a general hypothesis based on aggregate features extracted from web sessions such as average navigation time, average page requests, etc. Our approach outperformed the compared-to family of approaches by over 14% in terms of bot detection accuracy. We also evaluated the effectiveness of our approach in differentiating between web crawlers and other types of bots and we were able to correctly detect malicious bots with a recall of over 83% and web crawlers with a recall of over 91%.

The rest of the paper is organized as follows. In Section II, we give an overview of related approaches for bot detection, focusing on machine learning ones. Section III describes our website-specific approach for bot detection. In Section IV, we present our experimental results on two real-world websites. Finally , we conclude and provide future directions in Section V.

II. Related Work

There are many techniques for bot detection currently deployed. Perhaps one of the most known methods adopted worldwide is the active checking against the published domains or IP addresses of bots. Nevertheless, public bot-lists are never up-to-date, since robots change their identity, on-purpose, without prior announcement [1]. Therefore, many research efforts have been directed to detect bot attacks by other means, mainly through analysis of web session behavior. For example, an approach was proposed to detect bots by detecting whether the session involved any mouse movements or keyboard typing [12] by deploying java scripts with each page response. This approach clearly has its limitations since a number of users disable java scripts, and hence websites admins might be inclined not to use java scripts. The authors suggested to accommodate this by adding another layer of detection where they embed invisible links and check whether they are navigated during a session which is an indication of a bot activity. This trick can be easily countered by simply designing a bot that does not follow any invisible links. A different approach relies on virtual machines to capture three distinguishing features for bots, namely automatic start-up, command channel establishment and information dispersion or harvesting [17].

A family of approaches relied on machine learning as in the case of our approach [1], [5], [7], [11]. However, they mainly focused on extracting dominant features that distinguish humans from robots and assigning weights to those features. The features are then abstracted to become valid across all websites. Examples of such features are the frequency of hits from a single IP source or domain, percentage of page requests, average time between two consecutive page visits, average crawling time, average session timing, average of image and multimedia hits, expected crawling time, as well as other statistical features[1], [7], [11], [14]. Among this family of approaches, the best performing one utilized Bayesian network algorithms [7] and achieved an accuracy ranging from 80% to 86% across different websites. We compare their approach to ours and show that our approach, which is based on website-specific features rather than generic cross-website features, clearly outperforms their approach in terms of detection accuracy. This is not surprising as we believe bots have drastically evolved since the time of their experiments.

Another Bayesian approach was also proposed for detecting bots based on the similarity of their DNS traffic to that of known bots [15]. However, they assumed that bots in the same botnet have similar DNS traffic, through which they can be distinguished from other hosts. They also assumed that at least one bot in any botnet is known. A semi-supervised learning approach was proposed where the main focus was on handling the unbalanced data in the training set [16], which is an issue we also address via sampling in our approach.

Finally, there are many other techniques proposed for bot detection in specific domains such as game environments [18], [19], [20], scholarly information environments [21] and networks [22], [23]. However, our focus in this paper is on the detection of website bots.

III. Approach

Our system architecture is depicted in Figure 1. Our approach works as follows. A website log file consisting of

[4]http://wheelers.me

[5]http://whereleb.com

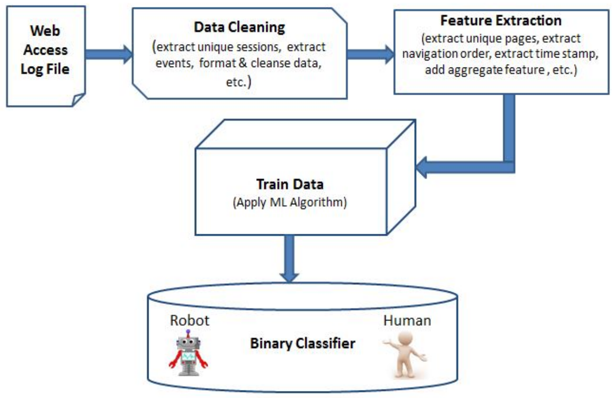
Web Access Log File
Data Cleaning
(extract unique sessions, extract events, format & cleanse data, etc.)
Feature Extraction
(extract unique pages, extract navigation order, extract time stamp, add aggregate feature, etc.)
Train Data
(Apply ML Algorithm)
Robot
Binary Classifier
Human

Fig. 1. System Architecture

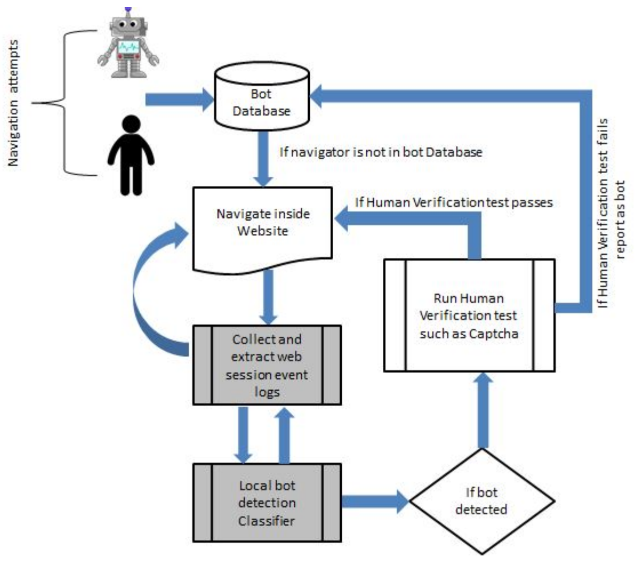
Navigation attempts
Bot Database
If navigator is not in bot Database
Navigate inside Website
If Human Verification test passes
Run Human Verification test such as Captcha
If Human Verification test fails report as bot
Collect and extract web session event logs
Local bot detection Classifier
If bot detected

Fig. 2. Bot Detection Algorithm

event logs for sessions is first fed to a data-cleaning process, which is described in Section III-A. After the file has been pre-processed, a set of website-specific as well as aggregate features are extracted and are used to train a binary classifier that predicts for a new session, whether it belongs to a human or a bot. Our feature extraction procedure is described in Section III-B and our classification approach is described in Section III-C.

Once a classifier has been trained, it can be effectively used to detect bots as follows. Given a navigation request to the website, a new web session is initiated. The web session is then constantly passed through the same data cleaning and feature extraction processes that were used for training the classifier. The web session is also constantly assessed by the learned classifier, which outputs whether the session belongs to a human or bot. In the latter case where a session was labeled as bot, the navigator is asked to pass a verification test to verify that she is a human, and if the verification test fails, the IP address and domain of the navigator is reported as belonging to a bot. In case a bot was not detected by our classifier, the navigator can continue to navigate the website, and with each navigation step, the session again passes through the detection algorithm that we just described. Figure 2 describes our bot detection algorithm.

A. *Log File and Data Cleaning*

A website log file typically consists of primitive information about web sessions such as the session id, the client IP address and domain, the navigated pages, the navigation timestamps and other navigation information such as multimedia hit events. We first extract the unique sessions and present each session as a series of page hits. Moreover, we label each session as either *bot* or *human* as follows. We use a database of IP addresses and domains of known bots, and for each session whose IP address or domain is in that database, we label its corresponding session as bot. While this will provide us with accurate labels for the positive class (i.e., bots), we cannot simply assume that all other web sessions which do not match an existing entry in the database belong to humans. In fact, our whole work is motivated by the fact that we cannot rely on such databases to fully detect bots. To be able to accurately label sessions as humans, we provide the system admin of the website with a tool to quickly inspect sessions that will be potentially labeled as human and judge whether they indeed belong to humans or should be labeled as bots. This way, we will be able to label every session as human or bot as accurately as possible. We acknowledge than even with this, we might still have some bot sessions that are labeled as humans, however those will be a small fraction and can be considered noise that should not affect a strong hypothesis that generalizes well by avoiding overfitting to the training data.

B. *Feature Extraction*

Once web sessions have been labeled, we proceed to extract two sets of features for each session. The first, which we refer to as *website-specific* or local features consists of four subsets. The first subset contains a binary feature for every page in the website, where the value of the feature is set 1 if it has been hit during that web session and 0 otherwise. These features have been used in previous work as well, however in the context of analyzing web content [4]. The second subset of features we extract consists of a numeric feature for every web page as well, which represents the navigation order for each page in the session. For instance, if page number 6 happened to be the second navigated page during the navigation path of the session, then the numeric feature representing the order of navigation for this page will be set to 2. In case a page was not hit at all during a navigation session, its corresponding order of navigation feature will be set to 0. On the other hand, If a page is hit more than one time during one session, then its first navigation order will only be considered.

TABLE I
AN EXAMPLE DATASET CONSISTING OF 5 SESSIONS NAVIGATING A WEBSITE OF 3 PAGES

Session	P1H	P2H	P3H	P1O	P2O	P3O	P1C	P2C	P3C	P1T	P2T	P3T	Agg1	Agg2	Agg3	Agg4	Label
1	0	0	1	0	0	1	0	0	2	0	0	0.14	0.67	0.14	0	4	Y
2	1	1	1	2	3	1	6	1	5	0	0.11	0.05	4	0.08	0.08	0	N
3	1	1	0	2	1	0	1	2	0	0	0.17	0	1	0.17	0	0	N
4	1	0	1	2	0	1	4	0	2	0	0.14	0.14	2	0.14	0	0	Y
5	1	0	1	1	0	2	2	0	4	0	0	0.12	2	0.12	0	2	Y

The third subset of website-specific features consists of yet another numeric feature for each web page, which represents the hit count for that page during the session. For example, if the page number 117 was hit 7 times throughout a given session, then a value of 7 is assigned to the corresponding feature. Again, if a particular page was never hit during a particular session, its hit count feature will be set to 0 in that case. Finally, the fourth and last subset of website-specific features is composed of a feature for each web page that represents the time spent on that web page. To compute the time spent in one web page, we subtract the timestamp at which that page was hit from the timestamp at which the consecutive page in the session was hit.

In addition to website-specific features, we also extract a set of features aggregated over the whole website for each web session. The aggregate features consist of the average page hits per session, the total multimedia hits, the average time needed to navigate from one page to another, along with the standard deviation of each.

Table I displays a sample dataset of 5 sessions for an example website consisting of 3 pages along with the features extracted for each session. For instance, session 3 represents a human that navigated the website hitting only pages 1 and 2 (P1H=1, P2H=1, P3H=0 where H stands for hit). Moreover, page 2 was navigated first (P2O=1, P1O=2 where O stands for order). During the navigation, the user hit page 1 once and page 2 twice (P1C=1, P2C=2 where C stands for count). The navigation time from page 1 to 2 was 0.17 msec (P2T=0.17 where T stands for timestamp). The Attribute *Agg1* represents the average hit per page for the session (total of 3 hits /total of 3 pages = 1). The attribute *Agg2* is the average navigation time between one page and another. The attribute *Agg3* is the standard deviation for attribute *Agg2*. *Agg4* is the number of multimedia hits, i.e. the count of picture and video hits. The features *P1H* up to *P3T* are the website-specific features, whereas the remaining four represent the set of aggregate features.

C. Classification

Our training data is comprised of instances of web sessions, where each instance is represented using the set of features extracted as described in the previous section, and which are labeled as either human or bot based on the criteria described in Section III-A. As is the standard in machine-learning approaches, we divide this dataset of web sessions into two sets, where the first consists of 60% of the data instances and we refer to as the training set, and the second contains the remaining 40% of the data and we refer to it as the test set.

The training set was used to train a number of classifiers, each with different hyperparameters. To set the hyperparameters of the trained models and to select the best model, we used *10-fold cross-validation* over the training set. The test set on the other hand was used to estimate the out-of-sample performance of the final selected classifier and to compare it to state-of-the-art approaches. As explained in the beginning of this section, the trained classifier can then be used to detect whether a new web session is originated by a human or a bot. In the latter case, where the session is detected as bot, the navigator is asked to pass a human verification-test and is either allowed to continue navigating the website or its IP address and domain are reported as belonging to a bot.

To ensure that our classifier is always up-to-date and can effectively detect bots as they evolve, the training phase of the classifier should be periodically carried out based on the most recent events in the website log file. In the next section, we describe our experiments on two real-world websites and show the effectiveness of our approach in detecting bots as compared to state-of-the-art approaches and other baselines. We also describe this re-learning phase and give guidelines on how it should be carried out based on actual data.

IV. EXPERIMENTS

A. Datasets

Our data on which we run our experiments is extracted from the log files of two popular websites, namely wheelers and whereleb. The first is an online vehicle dealer and consists of a total of 123 unique web pages. The second is a recommender website for businesses such as pubs and restaurants in a major city in the middle east and consists of 117 pages. For each website, a log file consisting of web sessions for a duration of one week was obtained and cleaned as explained in Section III-A. Moreover, each session in each dataset was labeled as either human or bot using the method described in the same section. Finally, the website-specific and aggregate features for each web session were extracted using the feature extraction procedure outlined in Section III-B. In particular, for wheelers, 496 total features were extracted, and for whereleb, 472 total features were extracted. This resulted in two datasets, the first consisting of around 100,000 instances, where each instance correspond to a web session for the wheelers website, and the second consisting of around 65,000 instances for the whereleb website.Table II provides a summary of the two datasets. Although the websites used for our experiments are of average

TABLE II
DATASETS

Website	#pages	#sessions	#features	#human	#bot
wheelers	123	101,234	496	14,153	87,081
wherleb	117	65,914	472	7,915	57,999

size, our regime is expected to perform well for websites with larger number of pages since the navigation traffic usually increases for large websites, which will allow more data instances to be extracted and used for training. In addition, regularization and feature reduction can also be used to deal with any blowup in the number of features that might occur in the case of very large websites. In fact, our approach can even perfectly work for websites with dynamically increasing pages as our proposed hypothesis can be re-tuned and retrained periodically.

B. Training and Validation

As mentioned in the previous section, each dataset was divided into two sets, a training set consisting of 60% of the instances in the dataset and a test set containing the remaining 40% of the instances. The training set was used to train a number of classification models, including but not limited to Support Vector Machines (SVM), Neural Networks (NN), Random Forests (RF), Naive Bayes (NB) and Boosted Decision Trees (BDT). All the models were trained using the Microsoft Azure cloud-based machine-learning framework [6]. The training set was also used to set the hyperparameters of the different learning models considered and for model selection using a 10-fold cross-validation. Since our datasets were quiet imbalanced as can be seen from Table II, i.e., the number of bot sessions was much larger than those of humans, we applied both under-sampling and over-sampling on our datasets [13]. Both are widely-used techniques to overcome the issue of data imbalance in machine learning, and we did not observe any difference in performance between both sampling methods. Hence, we present the results using under-sampling in our experiments.

Tables III and IV show the *cross-validation* results for the best performing Machine Learning Algorithms. Based on cross-validation, all classification models performed very close in terms of most of the standard classification metrics, with the Boosted Decision Trees (BDT) model performing slightly better than all other models in terms of accuracy, F1-Measure as well as the area under the curve (AUC) for both websites. Thus, we will only provide results for the BDT classifier henceforth. The hyperparameters of the BDT classifier were set as follows: maximum number of leaves per tree = 20, minimum number of samples per leaf node = 10, learning rate = 0.2, and number of trees = 100. The training time for all the trained classifiers was in the order of seconds.

[6]https://studio.azureml.net/

TABLE III
CROSS-VALIDATION RESULTS FOR WHEELERS

Model	Accuracy	Prec.	Recall	F1-Measure	AUC
SVM	0.790	0.853	0.702	0.770	0.852
NN	0.793	0.854	0.708	0.774	0.876
NB	0.799	0.843	0.737	0.780	0.882
RF	0.800	0.839	0.739	0.770	0.880
BDT	0.801	0.840	0.740	0.780	0.884

TABLE IV
CROSS-VALIDATION RESULTS FOR WHERELEB

Model	Accuracy	Prec.	Recall	F1-Measure	AUC
SVM	0.789	0.843	0.701	0.760	0.841
NN	0.791	0.843	0.708	0.773	0.862
NB	0.796	0.831	0.737	0.788	0.879
RF	0.797	0.828	0.739	0.756	0.878
BDT	0.800	0.838	0.765	0.780	0.881

TABLE V
FEATURE SELECTION RESULTS USING THE BDT CLASSIFIER

Website	Accuracy	Prec.	Recall	F1-Measure	AUC
wheelers	0.829	0.910	0.745	0.816	0.916
whereleb	0.827	0.900	0.739	0.808	0.899

C. Feature Selection

In this section, we perform multiple experiments to identify the best set of features for the task of bot detection using the Boosted Decision Trees classifier. Recall that our features consist of four sets of website-specific features plus a set of aggregate features. To avoid overfitting, we just try out different combinations of sets of website-specific features as a whole with different aggregate features.

The highest cross-validation accuracy as shown in Table V was obtained when sets 1,2 and 3 of the website-specific features were used along with the aggregate features *Agg1*, *Agg2* and *Agg3*, consistently for both websites. Recall that sets 1,2 and 3 of the website-specific features represent the hit pages, order of navigation and hit count of every page, respectively. On the other hand, *Agg1* represents the average hit per page for the session, *Agg2* is the average navigation time between one page and another, and *Agg3* is the standard deviation for *Agg2*. Note that the fourth aggregate feature *Agg4*, which represents the number of multimedia hits, dropped the accuracy whenever we attempted to add it to any combination of features in our datasets. Similarly, set 4 of the website-specific features, which represents the navigation timestamp from one page to another, was also negatively impacting the cross-validation accuracy whenever we considered it with any combination of features in our dataset. This can be attributed to the fact that many bots currently seem to navigate websites in a slower manner to imitate humans and thus avoid being detecting as bots.

Finally, we also experimented with a classifier that relied on the navigation order of pages as the only set of features and obtained an accuracy of around 79%. This indeed highlights the importance of local or website-specific features such as the navigation order in effectively detecting bots.

D. Comparison to State-of-the-art

In this section, we compare our approach with the best set of features that we obtained using feature selection to a representative of a family of machine-learning approaches that use statistical features aggregated over a whole website to detect bots [1], [4], [5], [7], [11]. The representative classifier was trained in a very similar fashion to ours. That is, for each instance or web session in each of our two datasets, the corresponding features used by any of the compared-to approaches were extracted. These features included the percentage of page requests, the average navigation time and its standard deviation, at what time the navigation occurred, the percentage of requests for zip and multimedia files, the percentage of image hits and the maximum click rate. We then trained a set of classifiers that uses all of these features using our training set, and used cross-validation to set the hyperparameters of the learned models and to select the best model. Moreover, we performed feature selection to select the best set of features among those used by the compared-to approaches, again based on cross-validation using the training set. The best model for the competitor was the Naive Bayes with a regularization weight of 1.

Finally, the *test set* was used to predict the out-of-sample performance for our approach versus the representative classifier of the compared-to approaches. Table VI shows the test results for our approach versus this competitor classifier. As can be seen from the table, the competitor classifier obtained an accuracy of around 73% ,whereas our approach had an accuracy of 83%, resulting in 14% increase in accuracy. Our approach also outperformed the competitor in all other metrics with the exception of *precision*. This means that our approach results in slightly higher false positives (i.e., humans that were detected as bots) than the competitor. This is intuitive since our approach tries to detect bots that mimic humans and has higher recall and F-measure, which traditionally comes at the expense of precision [9]. Moreover, detecting humans as bots is not as drastic as detecting bots as humans, since in the former case, the humans will pass the verification test and will thus be allowed to continue to navigate the website. On the other hand, detecting even just one bot as a human can have drastic results on the website in some cases.

E. Malicious Bots versus Search Engine Bots

We also evaluated the effectiveness of our website-specific approach in differentiating between various types of bots. The significance of this task is that sometimes website owners are willing to accommodate certain types of bots, say search engine ones, which mainly crawl the website in order to index it for web search. To this end, we built a three-way classifier that detects three types of navigators, namely humans, search engines bots, and malicious bots, instead of the binary classifier we used in all the previous experiments that just predicted whether a navigator is human or bot. In order to segregate search engine bots from malicious bots in our original dataset, we checked the IP address of each bot against popular search-engine yellow pages such google, bing, yahoo, etc [7]. Any bot session that was not listed in the yellow pages was thus labeled as a malicious bot. We then trained a three-way classifier exactly the same way we did for the binary classifier. Table VII shows the prediction results when a test file consisting of 1,000 sessions navigating wheelers was tested using our three-way classifier. As can be seen, our model can effectively differentiate between both types of bots with a small margin of error.

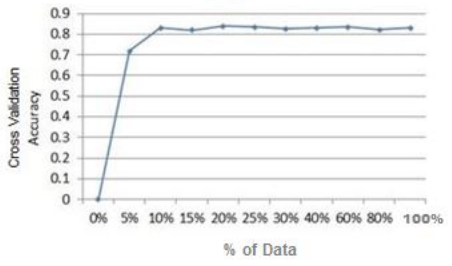

Fig. 3. Learning Curve for wheelers

F. Learning Phase

One of the crucial questions we need to address is how much data is needed for a specific website to learn a robust classifier for bot detection? The more the website is popular and is frequently visited, the shorter the learning phase becomes. In the case of wheelers, we were able to collect around 100,000 instances within one week, and around 65,000 instances for whereleb. We followed the standard method of plotting the learning curve in order to check at which point the cross-validation accuracy starts to converge. That is, we plotted the cross-validation accuracy using the Boosted Decision Trees classifier against different sizes of training sets (i.e., varying the training set size from 5% to 100% of the whole dataset). In the wheelers website case as can be seen in Figure 3, it turned out that 10% of the 100,000 instances were enough to learn a robust classifier that generates well. Similar conclusion was obtained when inspecting the whereleb learning curve (See Figure 4) where around 15% of the data (65,000 instances) was sufficient for training in that case. In general, around 10,000 web sessions were sufficient for both websites to learn robust classifiers that can be used to detect bots. This is inline with popular rules of thumb which state that the size of the training data should be in the order of 10 times the number of features (which were around 500 in our case for both websites).

G. Error Analysis

In this section, we try to address the following question: how many pages does a bot have to navigate within a web session before it is detected by our classifier? To answer this question, we divided our test data into a series of folds. Fold 1 had all the test sessions where a maximum of 10 pages were navigated. Fold 2 contained the test sessions where a maximum of 20

[7]http://www.iplists.com

TABLE VI
TEST RESULTS FOR OUR APPROACH VERSUS THE COMPETITOR OVER BOTH WEBSITES

		wheelers						whereleb				
Approach		Accuracy	Prec.	Recall	F1-Measure	AUC		Accuracy	Prec.	Recall	F1-Measure	AUC
Our		0.831	0.920	0.765	0.815	0.921		0.826	0.890	0.739	0.801	0.881
Competitor		0.734	0.937	0.514	0.660	0.768		0.731	0.916	0.502	0.601	0.760

TABLE VII
RESULTS OF THE THREE-WAY CLASSIFIER ON WHEELERS

Class	Sessions	Correct Pred.	Wrong Pred.
Malicious Bots	131	109	22
Search Engine Bots	679	620	59
Human	190	161	29

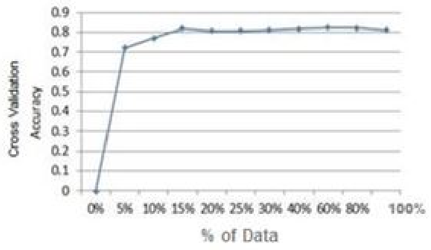

Fig. 4. Learning Curve for whereleb

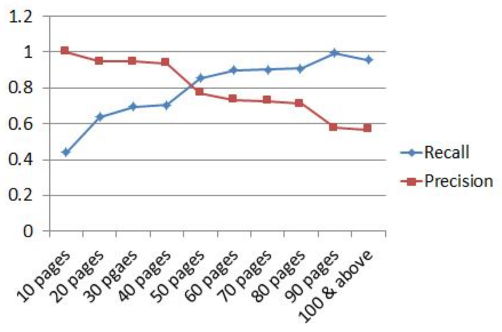

Fig. 5. Precison / Recall vs Navigation Length for our classifier on wheelers

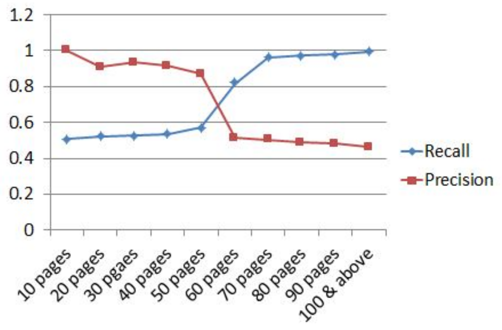

Fig. 6. Precison / Recall vs Navigation Length for the competitor on wheelers

pages were navigated, and so on. The last fold was the set of sessions where 100 pages and above were navigated. Figures 5 shows the recall and precision curves versus the session length (i.e., number of pages navigated) for wheelers. As can be seen from the figure, for short sessions, the precision is typically very high while the recall is low. This is indicative that it is difficult to detect bots with just few navigation steps. However, as the bot navigates the website more, it is more likely to be detected by our approach. This can be verified by the fact that as the session length increases, the recall increases as well. This was also evident in the case of whereleb as can be seen in Figure 7.

Figures 6 and 8 display the same curves but this time utilizing the competitor approach. As can be seen, our classifier has a significantly better precision-recall tradeoff than the competitor, regardless of the session length. This indicates that even for very meticulous bots that might not navigate a website long enough to be detected, we are still able to detect them more frequently than our competitor. Moreover, the cut-off point where precision starts trailing recall, and hence more bots will be detected, is earlier for our approach compared to the competitor.

In another experiment, we performed feature ranking using Weka [8] to rank the features for our best performing classifier with respect to the feature ability in differentiating between humans and bots. Table VIII shows the top 10 features for wheelers where the top three features correspond to page number 17 of the website. After inspecting the website to uncover the identity of this page, it turned out to be one of multiple pages dedicated for news and advertisements of vehicle offers. Similarly, Table IX shows the top 10 ranked features for whereleb, where page 6 is the landing or index page. This is inline with our initial hypothesis that each website has a unique human navigation behavior that is highly dependent on the structure and the semantics of the website. It will be almost impossible for any bot to be able to discover how frequently a page within a website is visited and in which order it is usually navigated by actual users unless it has access to the website's log file, which is almost impossible under normal circumstances.

Finally, we sampled some wrongly classified instances and tried to manually investigate why they were misclassified. Some of the incorrectly classified bots were missed by our classifier because they navigated a very small number of pages (one or two pages only), which made it impossible for our classifier to distinguish them from actual users who also

[8]Machine Learning tool, https://weka.wikispaces.com/

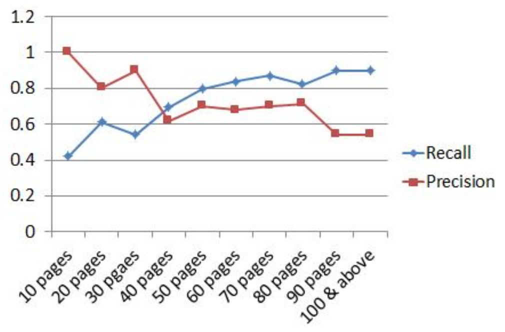

Fig. 7. Precison / Recall vs Navigation Length for our classifier on whereleb

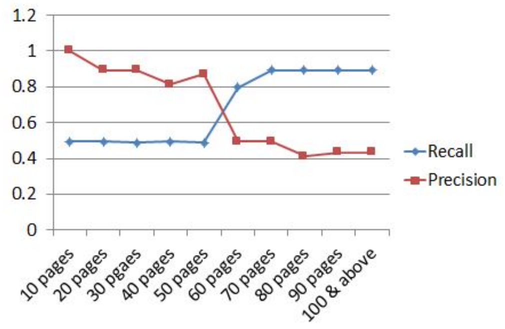

Fig. 8. Precison / Recall vs Navigation Length for the competitor on whereleb

briefly navigated the website. On the other hand, some of the incorrectly classified human sessions were possibly detected as bots by our classifier due to their tendency to navigate the pages of the website in an adhoc manner.

V. Conclusion and Future Work

In this paper, we proposed a machine-learning approach to detect bots based on local web session navigation behavior. Our approach is website specific and makes use of the unique nature of each website and how it is navigated by humans and is thus difficult to counter unless website-specific bots are designed as well. The proposed bot detection approach utilizes a binary classifier that is built using web session information that are typically present or can be captured upon request in the log files of websites. We outlined data cleaning and feature extraction methods that can be used to transform a website's log file into a training dataset that can be used to train our bot detection classifier. We also outlined how our proposed approach can be deployed in a real-world scenario as a mechanism to automatically detect bots as they navigate websites. We evaluated our approach on two real-world websites and showed its effectiveness in detecting bots compared to a family of state-of-the-art approaches combined together.

We believe that a truly comprehensive bot prevention tool shell offers more than one layer of protection for websites, employing a range of methods and technologies such as statistical and probabilistic methods, artificial intelligence, agent validation and various network approaches. In this paper, we are proposing to consider adding on top of that a local bot detection capability. In future work, we plan to test our approach on more websites, particularly larger ones. The most challenging aspect in doing so would be to acquire the access logs for such websites. We also plan to investigate other data mining techniques to improve the accuracy of our bot detection approach such as process mining.

TABLE VIII
Top 10 feature for wheelers

Rank	Feature	Feature Interpretation
1st	P17C	Hit count of page 17
2nd	P17O	Order of navigation of page 17
3rd	P17H	Whether page 17 was hit or not
4th	PH7	Whether page 7 was hit or not
5th	PO7	Order of navigation of page 7
6th	Agg3	Average time stamp
7th	PH9	Whether page 9 was hit or not
8th	PC9	Hit count of page 9
9th	PO9	Order of navigation of page 9
10th	PO1	Order of navigation of page 1

TABLE IX
Top 10 feature for whereleb

Rank	Feature	Feature Interpretation
1st	P6O	Order of navigation of page 6
2nd	P6C	Hit count of page 6
3rd	P6H	Whether page 6 was hit or not
4th	PH19	Whether page 19 was hit or not
5th	PO19	Order of navigation of page 19
6th	PC19	Hit count of page 19
7th	PO2	Order of navigation of page 2
8th	PC2	Hit count of page 2
9th	PO1	Order of navigation of page 1
10th	PH2	Whether page 2 was hit or not

VI. Acknowledgment

We would like to thank the owners and system administrators of wheelers and whereleb who were cooperative at all time and provided the web logs to conduct our experiments.

References

[1] Stassopoulou, Athena, and Marios D. Dikaiakos. ”Web robot detection: A probabilistic reasoning approach.” Computer Networks 53.3 (2009): 265-278.
[2] Kugisaki, Yuji, et al. ”Bot detection based on traffic analysis.” Intelligent Pervasive Computing, 2007. IPC. The 2007 International Conference on. IEEE, 2007.
[3] Yu, Fang, Yinglian Xie, and Qifa Ke. ”Sbotminer: large scale search bot detection.” Proceedings of the third ACM international conference on Web search and data mining. ACM, 2010.
[4] Wang, Chao, Jie Lu, and Guangquan Zhang. ”Mining key information of web pages: A method and its application.” Expert Systems with Applications 33.2 (2007): 425-433.
[5] Pao, H.K., Y.J. Lee, and C.Y. Huang. ”Statistical learning methods for information security: fundamentals and case studies.” Applied Stochastic Models in Business and Industry 31.2 (2015): 97-113.
[6] McKenna, Sean F. ”Detection and classification of Web robots with honeypots”. Diss. Monterey, California: Naval Postgraduate School, 2016.
[7] A Balla, A Stassopoulou and M Dikaiakos.”Real-time Web Crawler Detection”.2011 18th International Conference on Telecommunications.
[8] Amant, Robert St, and David L. Roberts. ”Natural Interaction for Bot Detection.” IEEE Internet Computing 20.4 (2016): 69-73.

[9] Morstatter, Fred, et al. ”A new approach to bot detection: striking the balance between precision and recall.” (2016): 1-8.

[10] Shyry, S. Prayla. ”Efficient Identification of Bots by K-Means Clustering.” Proceedings of the International Conference on Soft Computing Systems. Springer India, 2016.

[11] Fetterly, Dennis, Mark Manasse, and Marc Najork. ”Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages.” Proceedings of the 7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004. ACM, 2004.

[12] Park, KyoungSoo, et al. ”Securing Web Service by Automatic Robot Detection.” USENIX Annual Technical Conference, General Track. 2006.

[13] Estabrooks, Andrew, Taeho Jo, and Nathalie Japkowicz. ”A multiple resampling method for learning from imbalanced data sets.” Computational intelligence 20.1 (2004): 18-36.

[14] Bomhardt, Christian, Wolfgang Gaul, and Lars Schmidt-Thieme. ”Web robot detection-preprocessing web logfiles for robot detection.” New developments in classification and data analysis. Springer Berlin Heidelberg, 2005. 113-124.

[15] Villamarin-Salomon, Ricardo, and Jose Carlos Brustoloni. ”Bayesian bot detection based on DNS traffic similarity.” Proceedings of the 2009 ACM symposium on Applied Computing. ACM, 2009.

[16] Kang, Hongwen, et al. ”Large-scale bot detection for search engines.” Proceedings of the 19th international conference on World wide web. ACM, 2010.

[17] Liu, Lei, et al. ”Bottracer: Execution-based bot-like malware detection.” International Conference on Information Security. Springer Berlin Heidelberg, 2008.

[18] Kang, Ah Reum, et al. ”Online game bot detection based on party-play log analysis.” Computers & Mathematics with Applications 65.9 (2013): 1384-1395.

[19] Yampolskiy, Roman V., and Venu Govindaraju. ”Embedded noninteractive continuous bot detection.” Computers in Entertainment (CIE) 5.4 (2008): 7.

[20] Mitterhofer, Stefan, et al. ”Server-side bot detection in massive multiplayer online games.” IEEE Security and Privacy 7.3 (2009): 29-36.

[21] Huntington, Paul, David Nicholas, and Hamid R. Jamali. ”Web robot detection in the scholarly information environment.” Journal of Information Science 34.5 (2008): 726-741.

[22] Hsu, Ching-Hsiang, Chun-Ying Huang, and Kuan-Ta Chen. ”Fast-flux bot detection in real time.” International Workshop on Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2010.

[23] Al-Hammadi, Yousof, Uwe Aickelin, and Julie Greensmith. ”DCA for bot detection.” 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). IEEE, 2008.