AMERICAN UNIVERSITY OF BEIRUT

Web Session Navigation Behavior For Bot Detection

by
Rabih Abdulsalam Haidar

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science
to the Department of Computer Science
of the Faculty of Arts And Sciences
at the American University of Beirut

Beirut, Lebanon
April 2017

AMERICAN UNIVERSITY OF BEIRUT

Web Session Navigation Behavior For Bot Detection

by

Rabih Abdulsalam Haidar

Approved by:

Dr. Shady Elbassuoni, Assistant Professor — Advisor
Computer Science

Dr. Wassim El Hajj, Associate Professor — Member of Committee
Computer Science

Dr. Haidar Safa, Professor — Member of Committee
Computer Science

Date of thesis defense: April 18, 2017

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Haidar Rabih AbdulSalam
Last First Middle

◉ Master's Thesis ○ Master's Project ○ Doctoral Dissertation

☑ I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☐ I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One ___ year from the date of submission of my thesis, dissertation or project.**
Two ___ years from the date of submission of my thesis , dissertation or project.
Three ___ years from the date of submission of my thesis , dissertation or project.

Signature 20/4/2017
Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

Acknowledgements

We would like to thank the owners, developers and system administrators of wheelers and whereleb websites, who provided us with the log files we based our experiments on. They were not hesitant to address all our requirements regarding the meta data and format changes during the research. Special thanks to Professor Shady Elbassuoni and the committee members who provided all advisory and guide required throughout the journey of this thesis.

An Abstract of the Thesis of

Rabih Abdulsalam Haidar for Master of Computer Science
Major: Computer Science

Title: Web Session Navigation Behavior For Bot Detection

Web robots are everywhere in today's web technology. These bots range from robots associated with viruses known as malicious to spiders also known as search engine bots. The latter, attempt to crawl the Internet harvesting various information from websites for different purposes whereas no one can claim control over how and when this rich information is going to be used. While artificial intelligence keeps improving, robots become very smart too. Bots are likely to increase in quality and quantity as the world wide web develops and evolves. This is becoming a real threat to today's businesses and social life. What we're more likely to see in the future are smarter bots which can do anything at anytime. This naturally urges contemporary researchers and experts in cyber security to invest in every possible direction to try protect the web environment. Detecting bots ,whether malicious or search engine bots, is an important goal for most website admins. In this thesis, we propose a novel machine learning bot detection approach based on web session 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 bot attacks 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 state-of-the-art machine-learning bot detection approaches by almost 14%. In summary, we are after a solution where each website can learn, generate and tune its own defensive mechanism and can co-exist with other defensive mechanisms or even stand alone by itself. All the currently deployed solutions for bot detection

are fading and require replacement with time. Even the best currently innovative bot prevention methods such as CAPTCHA is expected to become useless as bots become more adaptive to it. On the contrary, our detection method is expected to survive and achieve higher accuracy in detection with time. This is due to its specific tailored hypothesis and responsive nature which makes it more accurate every time it learns about human navigation reaction towards its website pages. It is true that human behavior can be imitated by bots but it is going to be tough this time for a bot to learn and imitate the way humans behave and react to a particular website.

Contents

List of Figures

List of Tables

Chapter 1

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 such as email addresses and phone numbers from websites 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 negative impact on many businesses today. For example, a massive DoS attack has just recently caused an outage to many popular websites including Twitter, Spotify, and Airbnb [1]. Thus, bot detection is of great interest in the social media industry.

Automatically detecting bots is thus an important task for most websites. Various techniques and methods have already been proposed and are utilized to provide websites with defense mechanisms against malicious bot attacks. These methods are either based on IP address and domain look-up [2] or human verification-tests [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 tables or databases must be constantly maintained as bots tend to use different IP addresses or domains in order to by-pass such method of detection [1]. On the other hand, human-verification tests involve asking website 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 do these verifications tests [6]. Moreover, as new advances in information extraction and

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

machine learning techniques are achieved, bots can be taught to pass these common human-verification tests with high accuracy [3, 8].

In this thesis, we propose to use a machine learning approach to detect bots. Machine learning has been previously deployed for the task of bot detection. However, most of these approaches rely on generic features such as the average number of hits per page, the average navigation time, frequency of IP attempts, etc [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 web session navigation behavior. In particular, we build a classifier for each website that is based on website-specific features extracted from web sessions such as the actual hit pages, 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 is much more difficult to counter unless bot designers build specific bots for each website as well. Moreover, our approach can be 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 or domains 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 wheelers[2] and whereleb [3], and achieved an accuracy of around 83% in bot detection. We also compared our approach to a state-of-the-art machine learning approach [7] which relies 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 approach by over 14% in terms of bot detection accuracy. Although the accuracy achieved is less than the accuracy obtained with other non-machine learning approaches such as CAPTCHA, but our approach is expected to become more stable or even improved with time as the hypothesis tunes itself as in our model proposal. CAPTCHA is not fool-proof in stopping bots. It can be easily circumvented by character or image recognition and even by deploying CAPTCHA solving to cheap laborers. In general, all current approaches are proven to become less effective with time.

Our proposed hypothesis is literally made out of specific navigation characteristics that can well reveal the navigator identity. We believe that bot designers cannot easily imitate our selection of locally private features. This particularly provides better chances for generalization and can contribute towards a long-

[2]http://wheelers.me

[3]http://whereleb.com

term solution. Our model suggests that each website shall learn to seek its own hypothesis to defend itself. In order for bot designers to beat this, they have to design special bots that can cope differently with every single website. We will try in this thesis to study the learning phase requirements. We will address the questions of how much enough data instances and time are needed for a specific website of a specific size in order to build a robust private hypothesis.

The search engines eager attempts and continual trials to bypass bot detection methods provoke a challenge to website owners to continuously seek more solutions to allow them have more control over the robots or spider crawls. It is also important to note that any web user can also configure his source navigator to disguise in a robot identity, the fact that raises the need even more for website owners to proactively detect such behavior and take the needed precautions accordingly. Actually, in our studny, most malicious bots were spotted declaring themselves as search engine bots in their header protocol to avoid being detected. The need to limit robot sessions is crucial when it comes to small and medium size websites. These websites are usually deployed with reasonably humble hardware specs thus facing daily challenges to maintain acceptable page performance for their actual clients [1].

There are bad types of search engine bots which are usually sent to perform illegal type of crawls. These bad search engine bot ignore META tags and robots.txt file, which defines the crawling policing to be respected, follows URLs and revisit the site in a much higher frequency than expected thus causing it to crash sometimes. Anyways, banning or limiting search engine bot crawls is becoming more controversial nowadays and website admins start to seriously consider treating them as malicious bots. It is also important to note in this context that deep and dark websites are highly sensitive against spiders as their main mission is to remain isolated from the world wide web. This implies that even detecting good search engine bots could be an area of interest as well.

Primary research indicates that there have been very limited machine learning approaches that attempted to study session navigation behavior to tackle the robot detection-evolving problem. The currently existing and most effective implementations for bot detection are becoming source of annoyance to the actual web users. Modern methods rely on asking users every time they try to access a page to verify images manually as an attempt to ensure that the navigator is a human and accordingly prevent bots to surf their websites.

Search engines and fraud bots are becoming smarter and more adaptive to those defensive techniques applied at the web server tier [6]. For example, modern search engines are nowadays designing sophisticated robots that can crawl slowly into the web so that websites that are equipped with crawl-time detection intelligence algorithms are misled. Other bots are smart enough to detect and identify images when websites impose it on new sessions. Although bots that generate traffic can be easily spotted, a wide range of low rate robots are still difficult to identify and are sometimes associated with malicious attacks [3]. For

this reason, a solution which is based neither on IP address of the source nor on the frequency or speed of attacks is anticipated in this context.

The rest of the thesis is organized as follows. In the related work section , we give an overview of most relevant related work. We describe our website-specific machine learning approach in Section 3. In Section 4, we evaluate our approach on two real-websites and compare it to related approaches. Finally in Section 5, we conclude and provide future directions.

Chapter 2

Literature Review

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 capturing three distinguishing features for bots, namely automatic start-up, command channel establishment and information dispersion or harvesting [17].

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 technique was proposed where the main focus was on handling the imbalance in training data [16], an issue we also address via sampling in our approach.

Therefore, only few researches that seek to understand crawling behavior depend on the web logs of websites. They focus on bots that either originate from IP addresses belonging to search engine domains or those who declare their identity via HTTP-protocol headers. These researches statistically analyze access logs capturing the HTTP traffic and showed that the accumulated activity of sessions belonging to humans contributed to little percentage of the total HTTP

requests navigating the sites. They also proved that the behavior of bots have significant differences from humans. Similarly, they investigated the timely distribution of hits requests originating from search engines. They captured hundreds of thousands of HTTP requests where actual human crawling traffic contributed sometimes to less than 2% of the overall traffic. In general, it is quite inaccurate to judge on bots from the web log HTTP header declaration [1].

Botnet detection has received much attention in network and cyber security where researches tried hard to spot particular types of botnets based on the prior knowledge of the bot behavior. This is obviously not efficient as new bots are born everyday. In the same hand, there has been no systematic approach to identify various types of bots. There are studies that examined the anomalies in logs for detecting aggressive search bots. The counter argument is that not all search bots are aggressive, especially the recent ones [3].

Many of the solutions that employ data mining to tackle the problem of bot detection focus mainly on statistical methods such as counting the frequency of hits from a single IP source or domain. Others tend to learn statistically the navigation time behavior over the websites. Their main objective was to monitor specific features and collect statistical figures that can be abstracted for a global hypothesis that would remain true and applicable across all websites. What we did in this thesis is that we broke those constraints and freed ourselves from the obsession of having one single all-time true hypothesis. Despite the fact that session features are likely to change and highly vary from one website to another, our aim is to tackle the problem by targeting a simpler yet locally tailored hypothesis.

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. These 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]. The best-known statistical and probabilistic analysis of features relied on the classification of sessions known to have utilized the Bayesian network algorithms[7]. Their prediction model achieved an accuracy ranging from 80% to 86% on different websites. Unfortunately, the experiment relied heavily on features that were useful at that time but not anymore. Bots at that time used to crawl very fast and avoid image clicking. They also used to change IP address less frequently. The crawling time also used to be triggered during special hours of the day. Most of the mentioned features are not necessarily valid anymore since as we said earlier bots nowadays are much smarter and can disguise utilizing advanced technology. We compare their approach in addition to other approaches that attempted the same idea 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.

Our main mission in this thesis is to figure out a tailored hypothesis that is based on private feature values for each specific website. Given that these features are available in all websites, yet their feature values can vary dramatically from one web page to another generating a local hypothesis that requires a tailored bot to beat.

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].

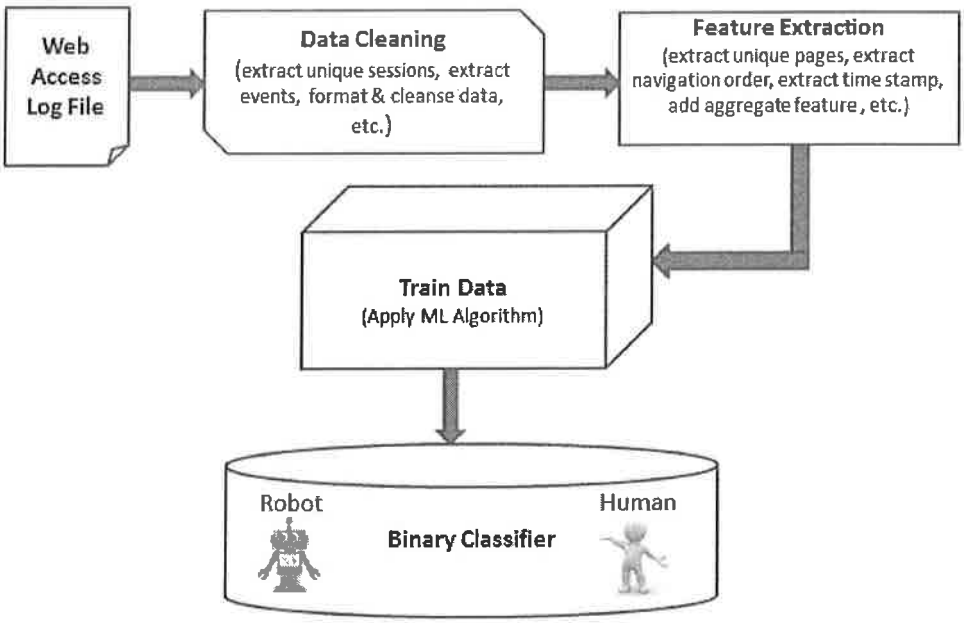


Figure 3.1: System Architecture

Chapter 3

Approach

Our system architecture is depicted in Figure 3.1. Our approach works as follows. A website log file consisting of event logs for sessions is first fed to a data-cleaning process, which is described in Subsection 3.0.1. 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 Subsection 3.0.2 and our training procedure is described in Subsection 3.0.3.

Once a classifier has been trained, it can be effectively used to detect bots

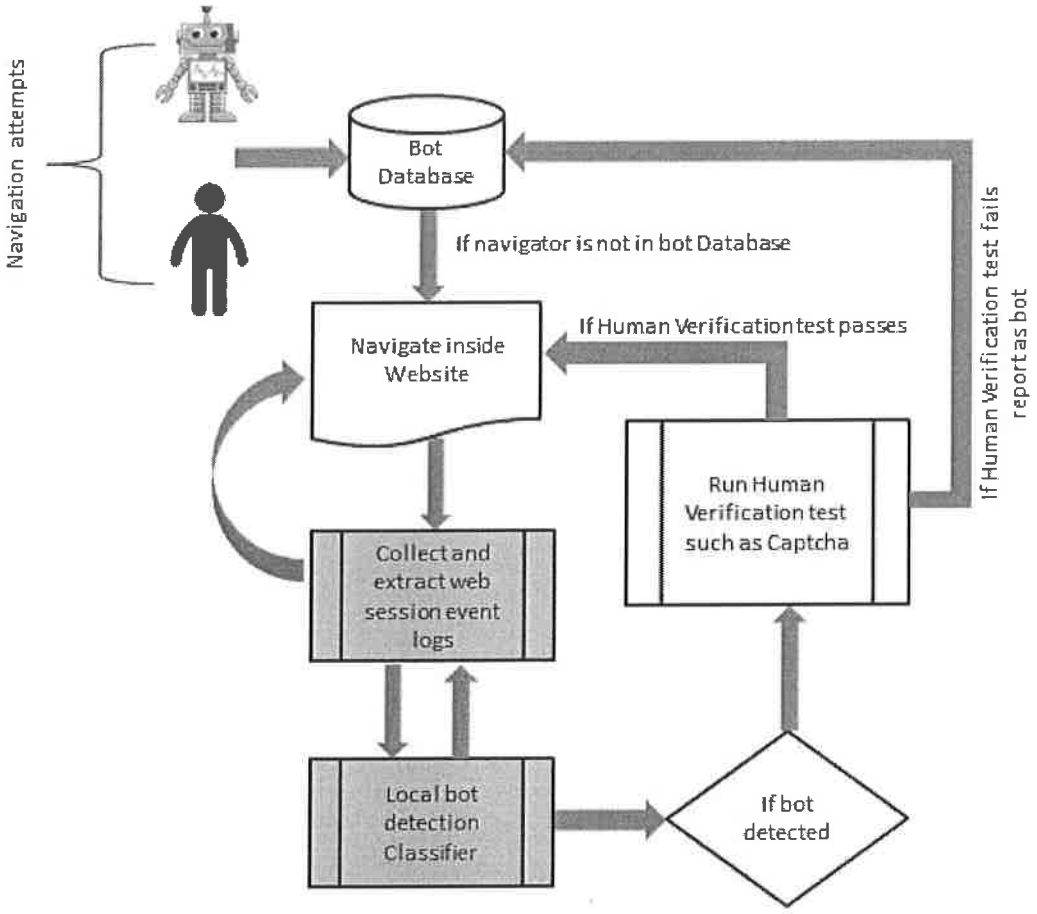


Figure 3.2: Execution Phase

as follows. Given a navigation request to the website, a new web session is initiated. The web session is then passed through the same data cleaning and feature extraction processes that were used for training the classifier. The web session and its web features are then passed through the learnt 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 it 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 3.2 describes the execution phase of our proposed model to detect bots.

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

Table 3.1: An example dataset consisting of 5 sessions navigating a website of 3 pages

3.0.1 Log File and Data Cleaning

Our classification model is based on web session navigation behavior which can be extracted from a website log file. A website log file typically consists of primitive information about web events related to sessions such as the session id, the client IP address and domain, the navigated pages, the navigation timestamps as well as many other navigation information such as multimedia hit events.

Given a website log file as described above, we first extract the unique sessions and present each session as a series of web page hits. Moreover, each session is labeled as either *bot* or *human* based as follows. We use a database of IP addresses and domains of known bots, and for each session whose IP address or domain matches an existing one 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 who 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.

3.0.2 Feature Extraction

Given a cleaned website log file as describe, we proceed to extract two sets of features for each web session. The first, which we refer to as website-specific features consists of four subsets. The first subset of website-specific features 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. Note that such features were extracted from web logs before but used for extracting and analyzing web content[4]. The second subset consists of a numeric feature for every web page as well that represents the order navigation 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 was the only one considered.

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 sessions. The aggregate features consists 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 3.1 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, it also happened that 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 page 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* in the table are the set of what we call web-specific features. The remaining four feature represent the set of aggregate features.

3.0.3 Classification

Our training data is comprised of instances of web sessions, where each instance is represented as the set of features extracted as described in the previous subsection and are labeled as either human or bot in the cleaning phase described in Subsection 3.0.1. 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 hyper-parameters and also to validate the trained classifiers and choose the best-

performing classifier among them using a 10-fold cross-validation. The test set on the other hand will be used to estimate the out-of-sample performance of the selected classifier. 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 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 experimental data on our two websites.

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

Table 4.1: Datasets

Chapter 4

Experiments

4.0.1 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 web log consisting of web sessions for a duration of one week was obtained and cleaned as explained in Subsection 3.0.1. Moreover, each session in each dataset was labeled as either human or bot using the method described in Subsection 3.0.1 as well. Finally, the website-specific and aggregate features for each web session were extracted using the feature extraction procedure outlined in Subsection 3.0.2. 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 4.1 provides a summary of the two datasets.

4.0.2 Cross-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 Vec-

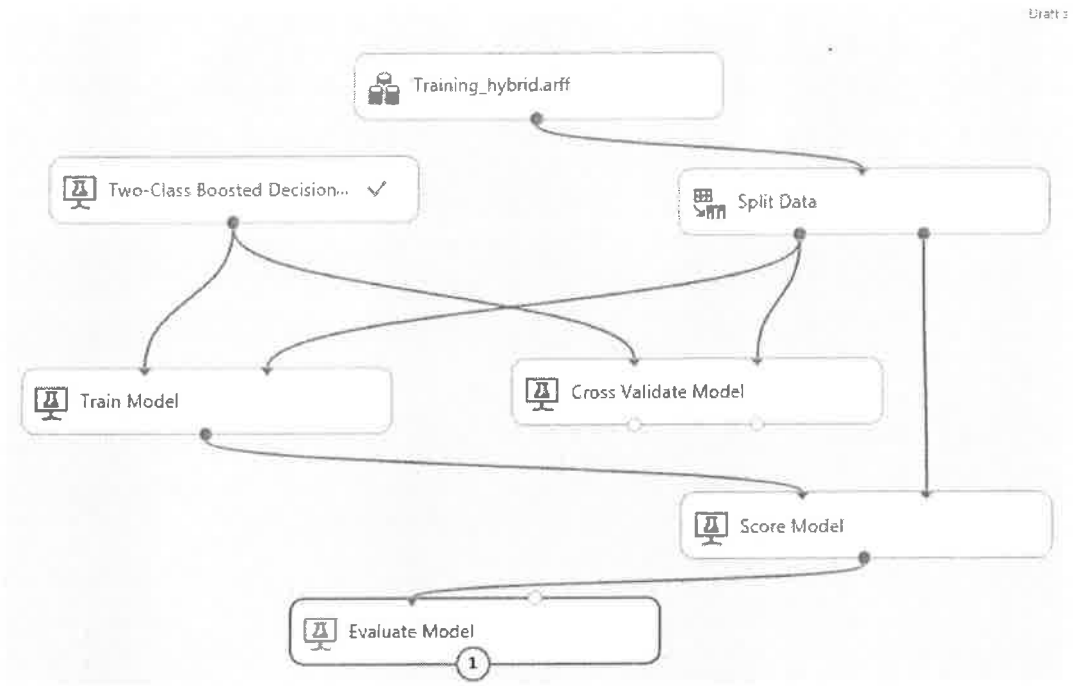


Figure 4.1: Experiment

tor Machine (SVM), Neural Networks (NN), Random Forest (RF), Naive Bayes (NB) and Boosted Decision Trees (BDT). All the models were trained using the Microsoft Azure cloud-based machine-learning framework [1]. The training set was also used to set the hyper-parameters of the different learning models considered and for model selection using 10-fold cross-validation. Since our datasets were quiet imbalanced as can be seen from Table 4.1, that is 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 under-sampling and over-sampling are widely-used techniques to overcome the issue of data imbalance in machine learning. We did not observe any difference in performance between both sampling methods and hence we present the results using under-sampling for our experiments. Figure 4.1 simulates the experiment steps using Azure.

Tables 4.2 and 4.3 show the *cross-validation* results for all the trained classifiers (with best hyper-parameters for each corresponding classification model) for the wheelers and whereleb websites respectively. As can be observed from the tables, all classification models performed very close in terms of most of the standard classification metrics, with the Boosted Decision Tree model performing slightly better than all other models in terms of accuracy, F1-Measure as well as the accuracy under curve for both websites. Thus, we will only provide results

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

for the Boosted Decision Tree classifier henceforth.

Model	Accuracy	Precision	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 4.2: Cross-validation results for wheelers

Model	Accuracy	Precision	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 4.3: Cross-validation results for whereleb

4.0.3 Feature Selection

In this subsection, we perform multiple experiments to identify the best set of features for the task of bot detection using the Boosted Decision Tree classifier. Recall that our features consists of four sets of website-specific features plus a set of aggregate features. To avoid over-fitting, 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 4.4 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 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 dropping the cross-validation accuracy whenever we considered it with any combination of features in our dataset.

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

Table 4.4: Cross-validation results for best features using Boosted Decision Tree Classifier

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

Table 4.5: Test results for the competitor's approach vs ours over both websites

4.0.4 Comparison to State-of-the-art

In this subsection, we compare our best-performing approach to a state-of-the-art approach for bot detection. Particularly, we compare our approach with the best set of features that we obtained from the Feature Selection experiments above to the machine-learning approach that uses statistical features aggregated over a whole web session to detect bots [7]. The competitor's approach uses a classifier based on the following set of features: percentage of page requests, average time navigation, standard navigation for time, percentage of requests for zip and multimedia files,percentage of image requests and maximum click rate. The competitor classifier was trained in a very similar fashion to ours. Namely, for each instance or web session in each of our two datasets, the corresponding features used by the competitor classifier were extracted and then a classifier was trained and cross-validated using the training set. Similar to our approach, feature selection was also applied to obtain the best of set of features for the competitor approach. The *test set* was then used to predict the out-of-sample performance for both approaches. Table 4.5 shows the test results for our approach using the Boosted Decision Tree classifier with the best set of features versus the competitor approach which used Naive bayes classifier using also the best set of features. As can be seen from the table, their approach demonstrated an accuracy around 73% at its best ,whereas our approach had an accuracy of 83% resulting in 14% of increase in accuracy. Our approach also outperformed the competitor approach 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 expected since our approach tries to detect bots that mimic humans and has higher recall and F-measure, which traditionally comes on 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.

4.0.5 Malicious Bots versus Search Engine Bots

Since our log files contain a large number of bot sessions, we also check whether our model has the potential to identify the type of bot it detects. The significance of this task is that sometimes website owners are willing to accommodate some 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. That is, instead of the binary classifier we used before 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 [2]. Any bot session that was not listed in the yellow pages was thus labeled as a malicious bot. We then trained as three-way classifier exactly as we did for the binary classifier. Table 4.6 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.

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

Table 4.6: Results for the three-way classifier on the wheelers

4.0.6 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 website, we were able to collect around 100,000 instances and around 65,000 instances for whereleb, within one week . We followed the standard method of plotting the learning curves 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 Tree 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 4.2, it turned out that 10% of the 100,000 instances were enough to learn a robust classifier that generates well. Similarly, Figure 4.2 shows the learning curve for whereleb and it can be observed that around 15% of the data (65,000 instances)

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

is sufficient for training in that case. Thus, 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 in the order of 10 times the number of features (which were around 500 in our case for both websites).

There is no evidence of real bias or variance which suggests that not much actions to be taken. Anyways, there is no chance to solicit more features especially that E_{in} (In-sample error) starts to track the E_{out} (out-of-sample error) at 10% of the original data set.

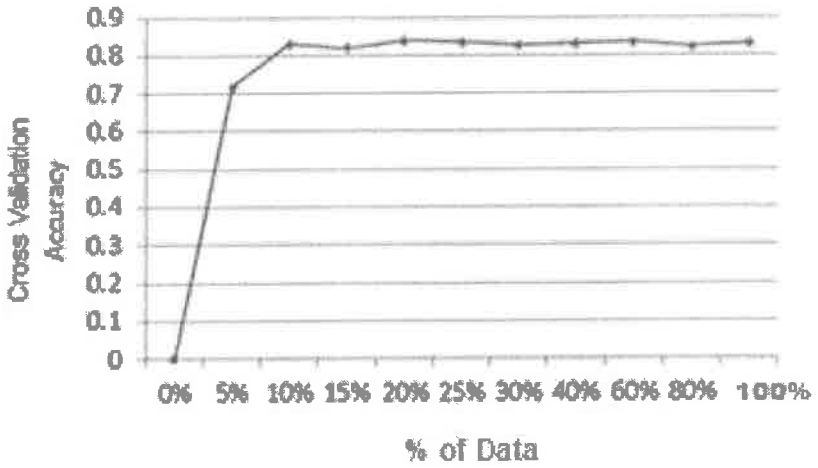
In our proposal and During The learning phase, the website will have to collect its own data and processes it to be able to produce its own hypothesis where the pages themselves, the navigation order, the count of hits per page, the average time-stamp are the features of the data set. The module is now in place and ready to identity the incoming sessions.

4.0.7 Error Analysis

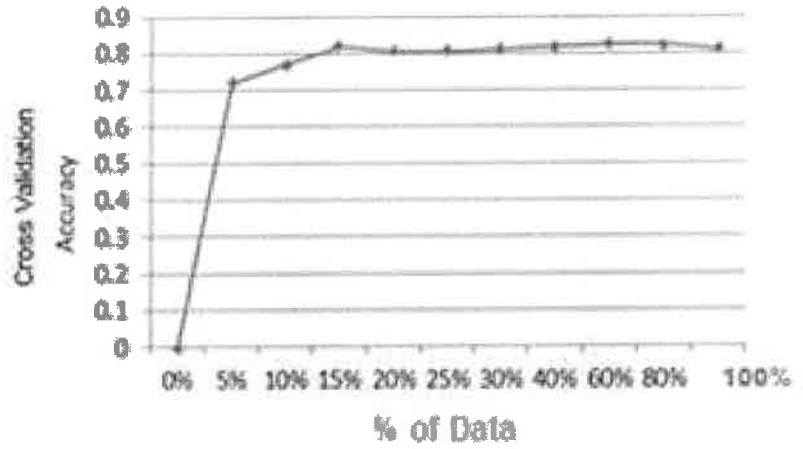
In this subsection, we perform some error analysis to understand how our classifier behaves. Particularly, we do feature ranking using Weka's [3] attribute selection ranker to rank the features for our best performing classifier with respect to the feature ability in differentiating between humans and bots. Tables 4.7 and 4.8 show the top 10 features for both websites. The top three features of wheelers correspond to page number 17 of the website. After going back to the website to uncover the identity of this page, we were not surprised to find out that it is one of multiple pages dedicated for news and advertisement of vehicle offers. Page 1 of wheelers as well as Page 2 of whereleb happen to be the landing pages of both websites, respectively. It is intuitive that these pages are significant in our hypothesis as most bots and humans usually access websites from the home URL. These findings made us more confident that website-specific features are dominant in terms of their importance in forming a robust bot detection hypothesis. It is 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 access log file, which is almost impossible under normal circumstances.

We also performed another analysis 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 sessions where a maximum of 10 pages were navigated. Fold 2 contained the test sessions where a maximum of 20 pages were navigated, and so on. The last fold was the set of sessions where 100 pages and above were navigated. Figure 4.3 show the recall and precision curves versus the session

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



[wheelers]



[whereleb]

Figure 4.2: Learning Curves

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 4.7: Top 10 feature for wheelers

length (i.e., number of pages navigated) for both websites. As can be seen from the two figures, 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 is consistent across the two websites.

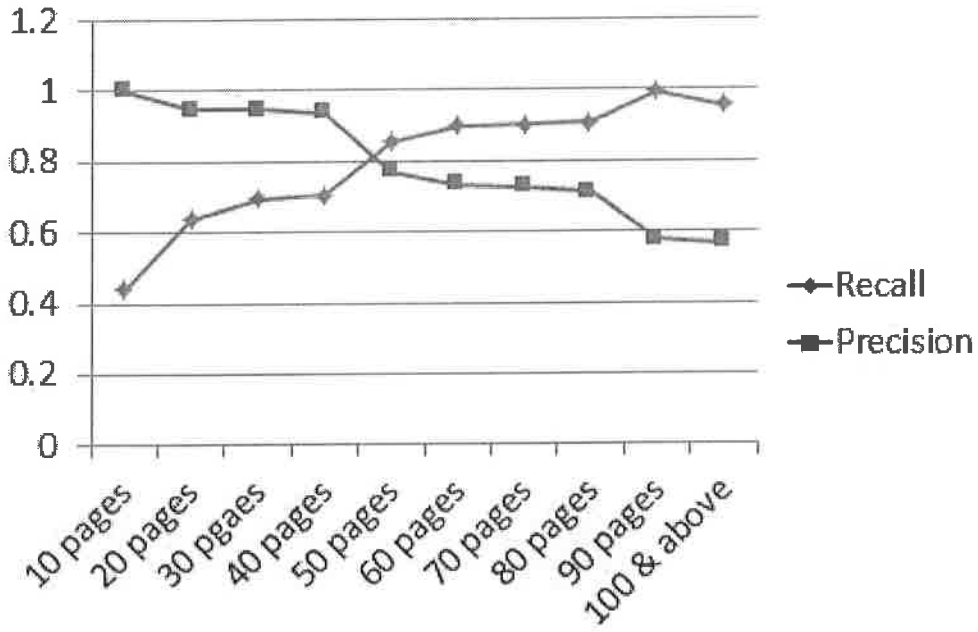
Figure 4.4 display the same curves but this time utilizing the competitor's approach. As can be seen, our classifier has a significantly better precision-recall tradeoff than the competitor, regardless of the session length. This indicates that even for very meticulous bots that might not navigate a website long enough to be detected, we still are 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.

Finally, we sampled some wrongly classified instances and tried to manually investigate what went wrong. We realized that a bunch of an incorrectly classified bots were possibly missed by our classifier because they accessed a very minimal number of pages (one two or two pages only) throughout their navigation, which made it almost impossible for our classifier to distinguish them from actual users who also tend to have similar behavior. We also noticed that robots rarely tend to hit one page and leave, on contrary they try to navigate as much pages as possible particularly pages with links and contact information. It is quite interesting to know the longest traversed paths were correctly classified as bots. Human sessions rarely navigate many pages. A massive number of correctly classified human sessions were spotted hitting a maximum of 5 pages before the sessions ends. Some of the wrongly classified human sessions were possibly detected as bots by our classifier due to their tendency to navigate the pages in awkward fashion in terms of logical page orders. At the end, it is really difficult to manually judge

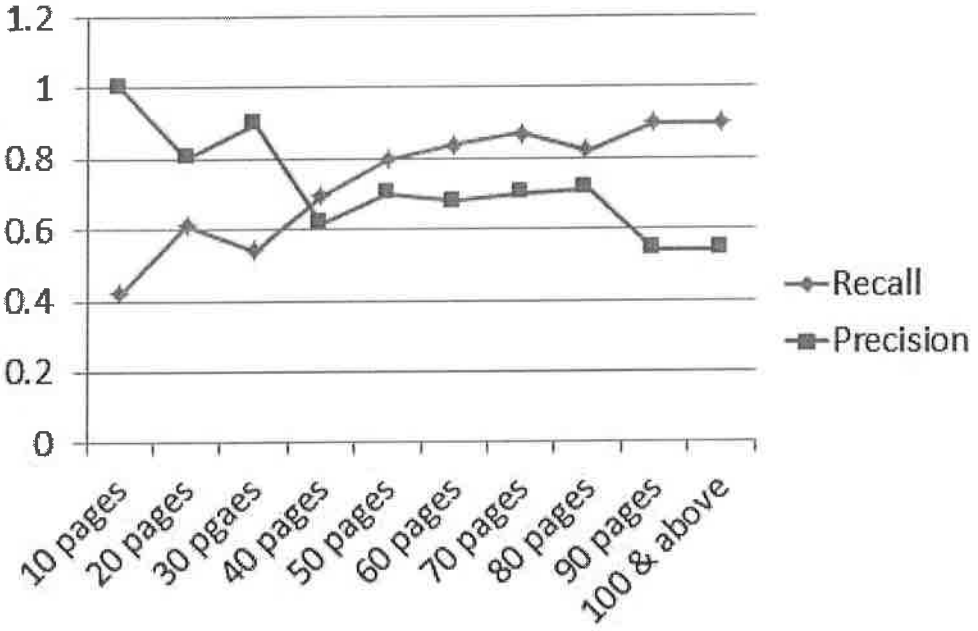
Rank	Feature	Feature Interpretation
1st	P6O	Order of navigation of page 6
2nd	P6C	Hit count of page 6
3rd	P6H	Whether page 6 was hit or not
4th	PH19	Whether page 19 was hit or not
5th	PO19	Order of navigation of page 19
6th	PC19	Hit count of page 19
7th	PO2	Order of navigation of page 2
8th	PC2	Hit count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

Table 4.8: Top 10 feature for whereleb

and understand the wrongly classified sessions knowing that we have a wide range of features to explore. Perhaps, we need to conduct some advanced statistical experiments to be able to obtain better visibility.

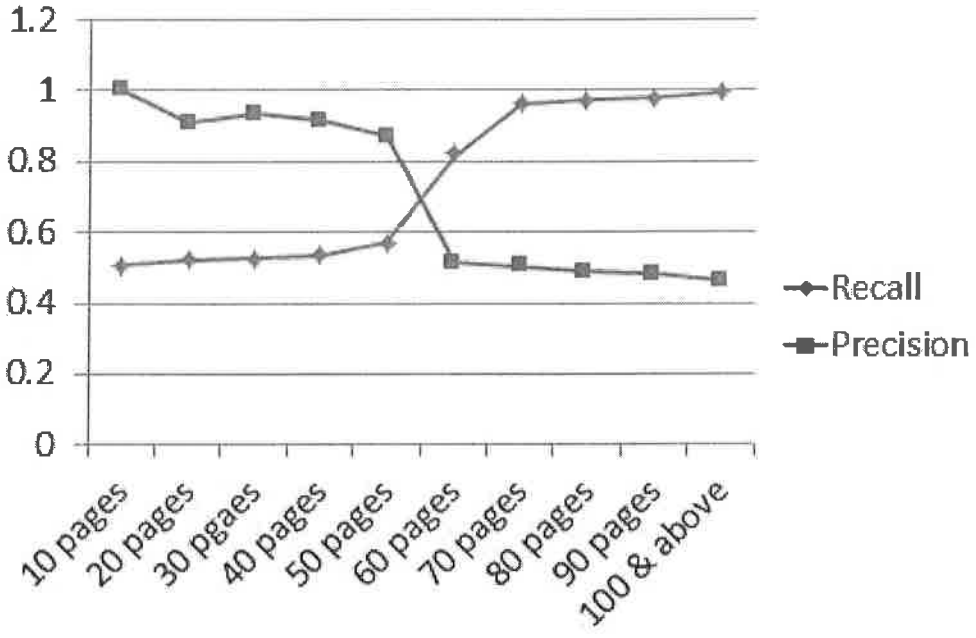


[wheelers]

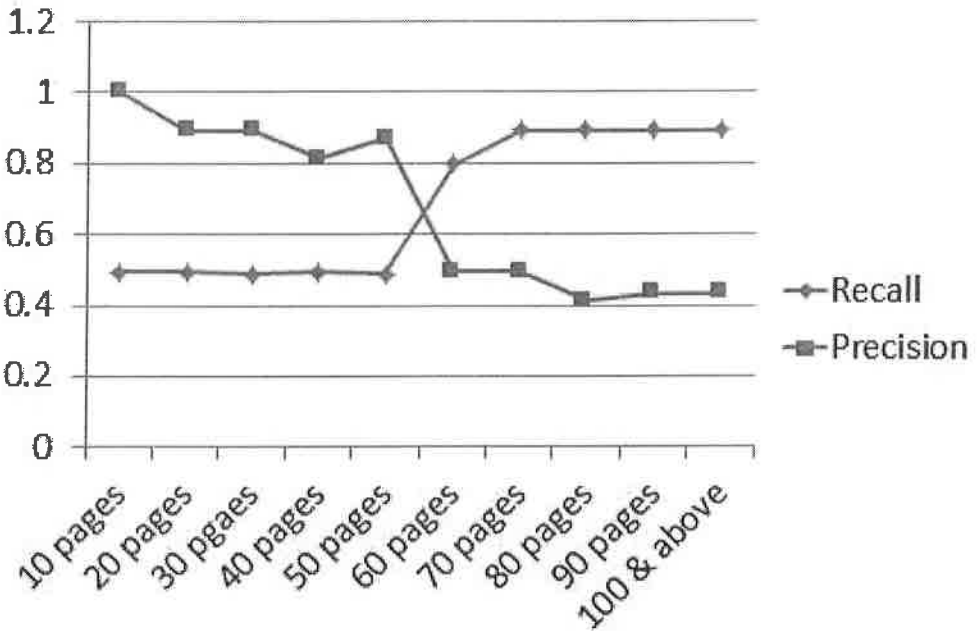


[whereleb]

Figure 4.3: Precison / Recall VS Navigation Length for our classifier



[wheelers]



[whereleb]

Figure 4.4: Precison / Recall VS Navigation Length for the competitor's classifier

4.1 Results And Analysis

We will present in the section the results of experiments V, VI and VII. Experiment V was a result of the promising results achieved by experiments I, II and III, which in their turns demonstrated individual accuracy between 79 % and 81 %.Experiment VI is a reproduced work of the traditional statistical machine learning approaches. We were trying to check its accuracy results against modern bots and at the same time trying to benefit from its data set features. Experiment VII is the final experiment that demonstrated best accuracy results out of combining experiment V with other statistical features from experiment VI which proven to be beneficial.The algorithm that demonstrated best results was the Two-class Boosted Decision Tree algorithm with maximum number of leaves per tree set to 20 and learning rate set to 0.2.

For each of the experiments we will present the 10-fold cross validation accuracy obtained by applying the supervised learning and then we show the testing results when the model is executed for the data dedicated for testing.

4.1.1 Experiment V -Results

Cross validation accuracy=0.8112

Testing Results: AUC[4]= 0.884, See tables 4.9 , 4.10 and Figure 4.5

Table 4.9: Testing Results for Exp.V

TP	FN	FP	TN
3952	1392	730	4567

Table 4.10: Testing Results for Exp.V

Accuracy	Precision	Recall	F1-score
0.801	0.844	0.740	0.788

4.1.2 Experiment VI-Results

Cross validation accuracy=0.7389

Testing Results: AUC=0.768, See tables 4.11 and 4.12 and Figure 4.6.

4.1.3 Experiment VII -Results

Cross validation accuracy= 0.8281

Testing Results: AUC=0.913, See tables 4.13,4.14,4.15 and Figure 4.7.

[4]Accuracy under curve

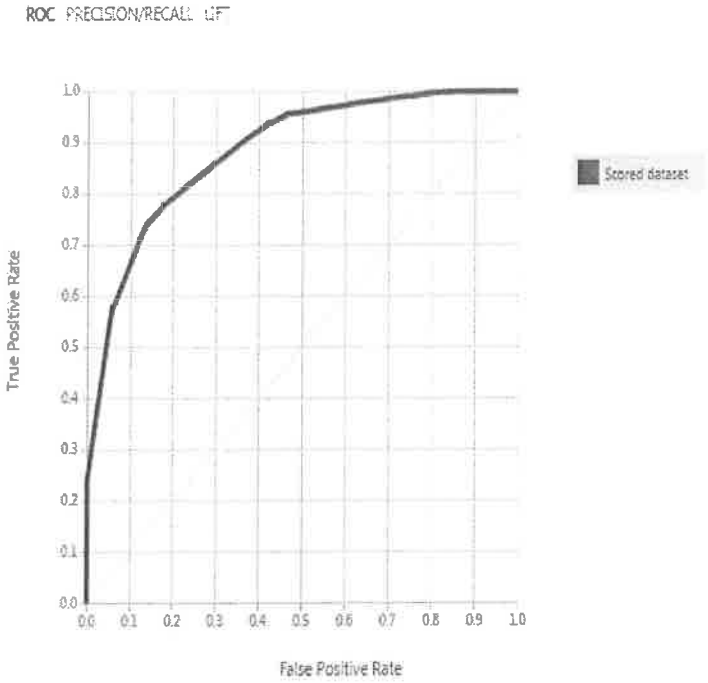


Figure 4.5: Accuracy Under Curve for Exp. V

Table 4.11: Testing Results for Exp.VI

TP	FN	FP	TN
2760	2614	156	5101

Table 4.12: Testing Results for Exp.VI

Accuracy	Precision	Recall	F1-score
0.739	0.947	0.514	0.666

Table 4.13: Testing Results for Exp.VII,Y=Bot

TP-Rate	FP-Rate	Prec.	Recall	F-Meas.	ROC	Class
0.917	0.266	0.779	0.917	0.842	0.909	Y
0.734	0.083	0.897	0.734	0.807	0.909	N
0.827	0.175	0.837	0.827	0.825	0.909	Avg

Table 4.14: Testing Results for Exp.VII

TP	FN	FP	TN
3896	1428	446	5020

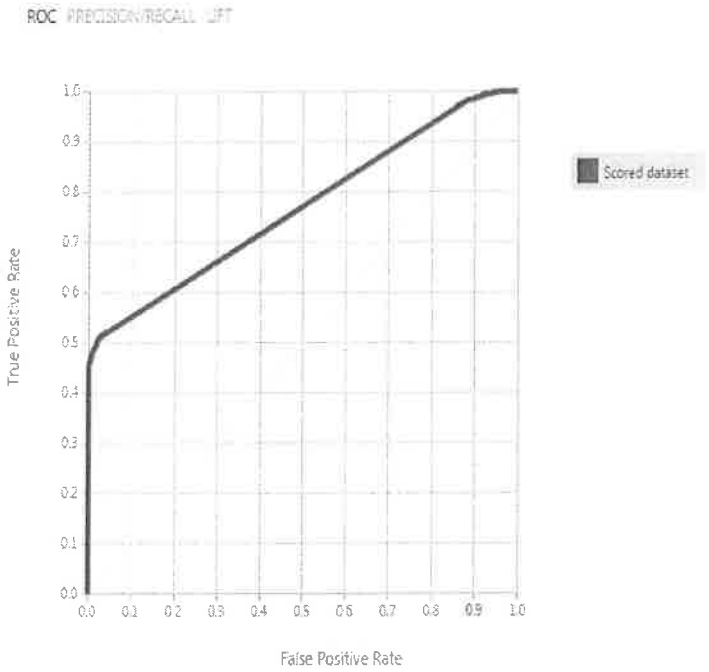


Figure 4.6: Accuracy Under Curve for Exp.VI

Table 4.15: Testing Results for Exp.VII

Accuracy	Precision	Recall	F1-score
0.826	0.897	0.732	0.806

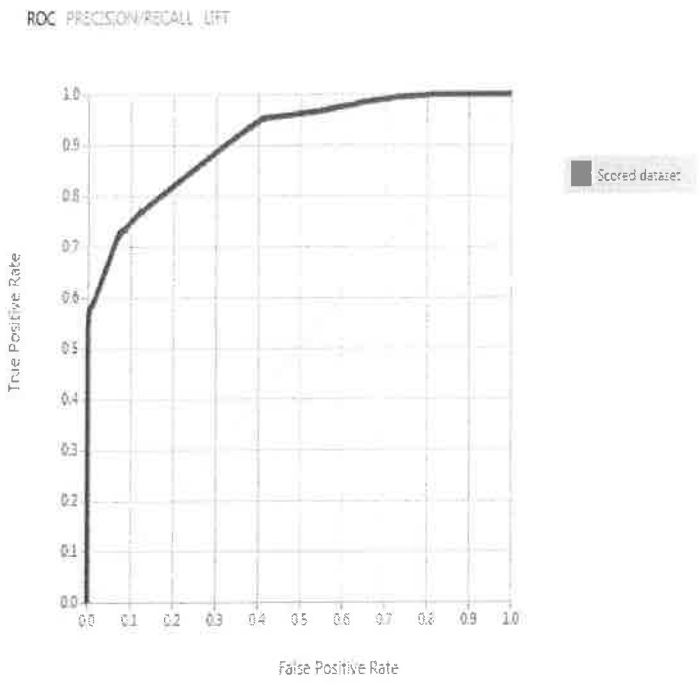


Figure 4.7: Accuracy Under Curve for Exp. VII

Chapter 5

Conclusion and Future Work

In this thesis, we proposed a machine-learning approach to detect bots based on web session navigation behavior. Our approach is website specific 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 a bot detection classifier. We tested our approach on two real-world websites and showed its effectiveness in detecting bots compared to a state-of-the-art approach.

We believe that a truly comprehensive bot prevention tool shell offer more than a layer of protection for the website. Employing a range of methods and technologies such as statistical and probabilistic methods, artificial intelligence, agent validation and various network approaches. In this thesis, 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. We also plan to investigate other data mining techniques to improve the accuracy of our bot detection approach such as process mining.

Appendix A

Abbreviations

BOT	Malicious or Search Engine Robot
ML	Machine Learning
SVM	Support Vector Machine Learning Algorithm
NN	Neural Networks Machine Learning Algorithm
NB	Naive Base Machine Learning Algorithm
RF	Random Forest Machine Learning Algorithm
BDT	Boosted Decision Tree Machine Learning Algorithm
AUC	Accuracy Under Curve
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
DoS	Denial Of Service

Bibliography

[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.