2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th Intl. Conference on Data Science and Systems

Online Web Bot Detection Using a Sequential Classification Approach

Alberto Cabri
DIBRIS
University of Genoa
Genoa, Italy
alberto.cabri@dibris.unige.it

Grażyna Suchacka
Institute of Mathematics and Informatics, University of Opole
Opole, Poland
gsuchacka@uni.opole.pl

Stefano Rovetta
DIBRIS
University of Genoa
Genoa, Italy
stefano.rovetta@unige.it

Francesco Masulli
DIBRIS
University of Genoa
Genoa, Italy
francesco.masulli@unige.it

***Abstract*—A significant problem nowadays is detection of Web traffic generated by automatic software agents (Web bots). Some studies have dealt with this task by proposing various approaches to Web traffic classification in order to distinguish the traffic stemming from human users' visits from that generated by bots. Most of previous works addressed the problem of *offline* bot recognition, based on available information on user sessions completed on a Web server. Very few approaches, however, have been proposed to recognize bots *online*, before the session completes. This paper proposes a novel approach to binary classification of a multivariate data stream incoming on a Web server, in order to recognize ongoing user sessions as generated by bots or humans. The present approach uses deep neural networks combined with Wald's Sequential Probability Ratio Test to express the relationship between subsequent HTTP requests in an ongoing session and to assess the likelihood of each session being generated by a bot or human before it ends. Experimental results showed the ability of the proposed approach to detect Web bots *online* with high performance scores and a small number of false negatives, as evidenced by the Recall index, minimizing the impact on human visitors. Another valuable indicator is the speed of decision: the present method allows very quick classification of nearly all sessions, leaving only very few of them undecided.**

I. Introduction

Contemporary Web-based services rely heavily on tasks performed online by automatic software agents, called Internet robots or Web bots. Recent reports about HTTP workload on Web servers confirmed that the majority of traffic may be attributed to robots and this trend is constantly growing [6], [29]. More than half of all robots are "bad bots" with malicious goals, such as impersonators, scrapers, spamming and hacking tools.

In practice, only a small fraction of all bots may be easily identified based on requests' user agent fields or by examining whether the "robots.txt" file was requested in session. Many bots, especially the malicious ones, hide their identities by impersonating legitimate Web clients via their user agent fields and thus access a website without being identified.

Robots browse the Web in a clearly different way than humans do. Differences in Web traffic patterns for bots and humans have been widely investigated, typically based on Web server access logs [4], [5], [13], [21]. These differences have motivated research towards bot recognition approaches based on navigational patterns and statistical properties of Web traffic [6], [9], [14]. Most research in this field has been devoted to tell bots and humans apart with the use of classification techniques, such as Bayesian classifiers [18], [23] decision trees [12], [24], support vector machines [8], association rule mining [10], or ensemble methods [17]. Some studies aimed at comparing the efficiency of various classification algorithms [3], [16], [19]. Furthermore, unsupervised classification techniques revealed a high potential for differentiating between bots and humans [20], [28].

Our preliminary results on unsupervised classification of Web sessions [15] confirmed intrinsic "behavioral" differences between human and bots, indicating that machine learning techniques may be efficient in their differentiation even in an unsupervised setting.

All the aforementioned studies considered the problem of bot detection *offline*, meaning that a classification decision is taken given the description of the whole session, completed at the server side. Solutions to that problem make it possible to gain an insight into properties of bot traffic and assess its impact on server performance and security. They are also helpful in preparing log data for the purpose of genuine visitors' behavioral modeling. However, they do not allow one to identify bots as they interact with the website, which is necessary to enforce proper bot treatment, e.g., to prevent them from accessing the sensitive site contents and minimize detrimental results of their activities.

To the best of our knowledge, only two studies addressed the problem of real-time Web bot detection [1], [6]. Balla et al. [1] proposed application of decision trees to classify ongoing sessions based on their aggregated features, updated for each page request. Minimum five clicks were needed to take a decision. Doran and Gokhale [6] proposed a method based on a first-order discrete time Markov chain model capturing differences in resource request patterns between bots and humans. The efficiency of their method depends on the minimum number of requests that must be observed before taking a classification decision.

The main contributions of this paper can be summarized as follows. We propose a novel approach for identifying bots that are visiting a Web site before the end of the session, i.e., in real time or on line. The approach does not require computing aggregated session features and does not model statistical dependences between subsequent requests in a session. The

978-1-5386-6614-2/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2018.00252

problem of online bot recognition is formulated as a binary classification task of sequentially sampled multivariate data, and solved by cascading a Multi-Layer Perceptron (MLP) neural network and a sequential decision module based on Wald's Sequential Probability Ratio Test (SPRT). The effectiveness of the proposed approach has been verified using a dataset from a real e-commerce Web server.

II. Formulation of the Problem

The HTTP protocol is stateless and does not define sessions [2] [7]. According to common practices [3] [6] [17] [18] [19], a Web session can be defined as a stream of HTTP requests that share the following properties: (1) they are generated from the same IP address; (2) they are associated to the same user agent string; and (3) the gap between two subsequent requests is at most thirty minutes. The length of a session may vary from two to several thousands entries.

The problem of online classification of Web sessions can be stated as an instance of early classification of multivariate data streams. It is reasonable to assume that, within the duration of one session, the navigational style and patterns do not change much, so the corresponding data stream can be assumed to be produced by a stationary source. On the other hand, there is a statistical dependence between subsequent requests. However, due to the asynchronous generation by user agents, the order of requests is noisy and the temporal correlation is difficult to model.

It is then possible to formalize the online Web bot detection problem as the task of identifying whether a sequence of HTTP requests for a given session can be labeled as performed by a bot or human before the end of the sequence. This is a binary classification task with sequentially sampled inputs.

III. The Proposed Online Bot Detection Method

A. A Two-Stage Approach

In order to develop the classifier and evaluate its efficacy, a set of sessions reconstructed from HTTP data recorded in Web server access logs is used. In this work, classification of variable-length streams of requests was addressed in two stages, as follows.

The first stage is a classifier that outputs a score, approximating the likelihood that the individual request was issued by a bot. In this phase each request is evaluated as if it were issued independently of any session. This classifier was implemented with a multi-layer perceptron (MLP) neural network (a neural network with dense connections, ReLU hidden units, and softmax output).

As previously noted, the temporal dependence of requests is difficult to model. Therefore, this stage does not take into account the sequential patterns of requests, but issues outputs based on the "naive assumption" of independence of the requests.

The second stage for the final decision making is then implemented by applying Wald's Sequential Probability Ratio Test [27] to the estimated likelihoods.

Any time a new request is received, a three-state classification decision is taken for the whole current session:

- class **zero**, if the session is due to a human visitor,
- class **one**, if the session is generated by a bot,
- **None**, if the output stage is not confident about the decision.

In the first two cases the decision is accepted. In the third case, classification of the session continues with the next request.

If no decision has been made before the session ends, the session itself is tagged as **undecided**. Since the goal of this work is online recognition *before* the session ends, to assess the performance of the method an undecided session is considered as a classification error, so in the testing phase it is counted as a false positive or a false negative, according to its ground truth.

B. Description of Request Features Used

Classification is based on a set of descriptive requests features extracted or determined from log entries. Each request is characterized by the following features:

- Numerical features:
  - inter-arrival time (seconds): it is zero for the first request in a session or reports the interval between the arrival times of current request and the preceding one,
  - size (kilobytes): volume of data transferred in the HTTP response.
- Categorical features:
  - HTTP method (GET, HEAD, POST, etc.), indicates the desired action to be performed for a given resource,
  - HTTP response status code (200, 404, etc.), is issued by a server in response to a client's request made to the server.
- Boolean features:
  - empty_referrer, it is *True* if the referrer field was empty,
  - is_page, it is *True* if the request was for a page description file or *False* for an embedded file,
  - is_graphics, it is *True* if an image file was requested,
  - is_style, it is *True* if the request was for a style description file,
  - is_datafile, it is *True* if a specific data file (e.g., a compressed document) was requested,
  - is_script, it is *True* if the request was for a script file.

Numerical features are standardized by subtracting the mean and scaling to unit variance. Categorical features are encoded in the one-hot mapping. Boolean values are encoded as zero or one for *False* or *True*, respectively.

As a result, the original ten-feature set describing each request was transformed into twenty five input variables, which can be given as an input to the neural network.

C. Labeling

The sessions assembled from HTTP data as described in Section II require appropriate labeling to provide targets for the training of a supervised classifier. In order to provide a reliable ground truth, this task was performed off-line, i.e., by considering all requests in each session and obtaining overall statistical indicators, and by including external knowledge.

In more detail, ground truth labels were manually assigned to each session with the support of Udger online database, available with a local parser license [25]. This database contains a taxonomy of known user agents and known IP addresses, commonly recognized by experts as representing bots or Internet browsers. In addition to the already mentioned source, a free online database, User-agents [26], has been used to reinforce the label attribution when in doubt.

According to the procedure followed [22], a session was labeled as bot-generated when at least one of the following conditions was met:

- the corresponding user agent had been assigned to the class "crawler", "validator", "e-mail client", "library", "multimedia player", or "offline browser" in the Udger database or had been reported as robot in the User-agents database,
- the corresponding IP address had been assigned to the class "crawler", "known attack source - mail", "fake crawler", "known attack source - http", or "known attack source - ssh" in the Udger database,
- the user agent contained a word suggesting a robot,
- the session had at least one of the features untypical for human sessions: the image/page ratio equal to zero, all page requests with empty referrers, all response codes equal to 4xx, or all requests of type HEAD,
- the file "robots.txt" was requested in the session.

Conversely, a session was labeled as human-generated if the corresponding user agent had been assigned to the class "browser" or "mobile browser" in the Udger database.

D. MLP Neural Network

The Multi-Layer Perceptron has been used to learn the sessions' model from the feature set described in the former section. The geometry of the neural network has been defined after preliminary experiments by picking variable-size random subsets of the original dataset to obtain good values of the selected performance measures while avoiding overfitting. The chosen geometry, in addition to the input and output layers, includes two identical hidden layers with fifty hidden units each (25-50-50-2).

The processing pipeline for neural network training includes therefore:

- standardization of the session features;
- k-fold cross-validation with 10 randomly sampled splits;
- hidden activation function: rectified linear unit, $f(x) = \max(0,x)$;
- output activation function: logistic sigmoid (softmax);
- cost function: cross-entropy, to approximate likelihoods;
- training with ADAM optimizer [11], based on stochastic gradient descent, with constant learning rate $= 10^{-3}$ and stopping after 1000 iterations maximum.

The output layer provides the probability estimates of each individual request for the two classes.

This setup was programmed in Python using the *MLPClassifier* module of the *Scikit-Learn* [30] library.

E. Sequential Probability Ratio Test

The probability estimated by the neural network form the information basis for early decision making. It is evident that, within a session, all requests are somehow related because they have been performed by either the same individual or the same bot and the exploration of a website is constrained by the already visited pages. However, for the purpose of the proposed approach each request is considered as independent from any other. This is a simplifying assumption that allows a simpler implementation, reduces the number of parameters to optimize, and at the same time it is a worst-case one, which prevents over-optimistic evaluations.

Online bot detection can be considered a sequential classification task in which observations (requests in a session) are sampled one at a time, and the hypothesis *the session is performed by a bot* is verified for each incoming observation.

As previously stated, the classifier can output three different values and the assumption of independence among the requests in a sessions qualifies the Wald's SPRT criterion as a good discriminator [27] based on posterior probabilities of available requests.

Let k be the number of observations at step k, let $x_1,...,x_k$ be the sequence of observations and let $p_1(k)$ and $p_0(k)$ be the posterior probabilities of class 1 (bot) and class 0 (human), respectively.

The basic sequential probability ratio is:

$$\frac{p_1(k)}{p_0(k)} = \frac{f_1(x_1)f_1(x_2)...f_1(x_k)}{f_0(x_1)f_0(x_2)...f_0(x_k)} = \prod_{i=1}^{k} \frac{f_1(x_i)}{f_0(x_i)}, \tag{1}$$

where $f_1(x_i)$ and $f_0(x_i)$ are the class likelihoods of observation x_i at step i. In this work, the expression in (1) was logarithmically transformed to use log-likelihood:

$$\log p_1(k) - \log p_0(k) = \sum_{i=1}^{k} \log\left(f_1(x_i) - \log f_0(x_i)\right). \tag{2}$$

The rationale is that a sum of differences is more numerically stable than a product of ratios when very small values are present.

Let C_1, C_0 ($C_1 > C_0$) be two thresholds related to the balance between errors of type I (false positive) and II (false negative). Then, according to the SPRT criterion:

$$decision = \begin{cases} 1 & \text{if} \quad \log p_1(k) - \log p_0(k) \geq C_1 \\ 0 & \text{if} \quad \log p_1(k) - \log p_0(k) \leq C_0 \\ \text{None} & \text{if} \quad C_0 \leq \log p_1(k) - \log p_0(k) \leq C_1 \end{cases} \tag{3}$$

The thresholds C_1 and C_0 have the dimensions of cumulative log-likelihoods. Their values control the balance between false positives and false negatives and influence the trade-off between early decision and classification with high confidence. In our experiments they were assigned by grid search, which led to selecting $C_1 = 4.6$ and $C_0 = -5.5$.

IV. Experiments

A. Experimental Setup

The final session dataset contained 13 395 sessions: 6 195 bots and 7 200 humans, therefore the classes are roughly balanced. Sessions that remained unlabeled (two sessions only) were not used in this experimental study.

The data source, containing 1 397 838 HTTP requests, was obtained from a real online retailer, whose identity cannot be revealed due to a non-disclosure agreement. Data was collected from April 1^{st} to April 30^{th}, 2014.

B. Evaluation criteria

The goal of this study is detection of Web bots, that are assigned to class 1, whereas humans are in class 0. The misclassified sessions are added to either false positives (true zeros classified as ones) or false negatives (true ones classified as zeros) according to their relevant ground truth.

Another important statement regards the undecided sessions, which are considered errors because the aim of the classifier is taking a decision before the end of the session: each undecided session is assigned to the opposite class, according to the ground truth. Experimental analysis highlights that most of undecided sessions are composed of a small number of observations as shown in Fig. 3.

Measures selected to evaluate classifier performance are accuracy, precision, recall and F1. The performance scores are presented in a cumulative form considering the number of requests that were sufficient to take a classification decision (Fig. 1). For each step and all the preceding ones the incremental number of true and false positives and negatives were computed and used to determine the corresponding performance scores.

C. Results

As reported in Fig. 1, the proposed classifier is able to detect bots in real time to a high extent, achieving F1 score in the range of 0.96 to over 0.98, depending on the classification step. Precision, which measures the performance with respect to erroneously classifying humans as bots, is much higher than recall, which is always greater than 0.94. This indicates that human visitors are not penalized by the algorithm.

Figure 2 demonstrates a very high efficiency of the algorithm at the initial classification steps over the greatest amount of sessions, which is the main objective of our early bot detection approach. More than 85% of the recognized sessions can be classified at the second request and nearly 100% at the fifth one. The greatest part of bots may be detected at the very beginning of their sessions, even at the first request arrival, leaving only 0.71% of the total sessions as undecided.

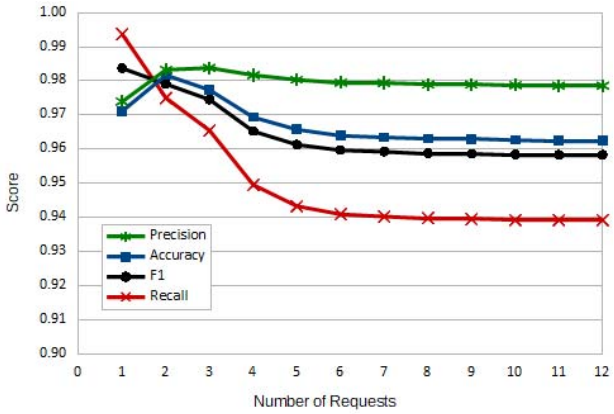

Fig. 1. Evolution of classification indexes as a function of the number of steps. Undecided sessions are counted as errors.

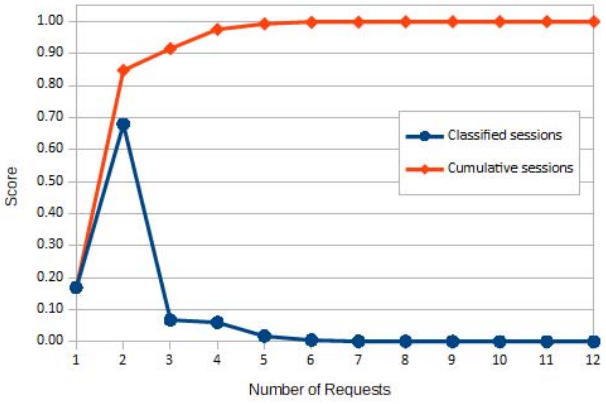

Fig. 2. Fraction of sessions classified at each step and cumulatively.

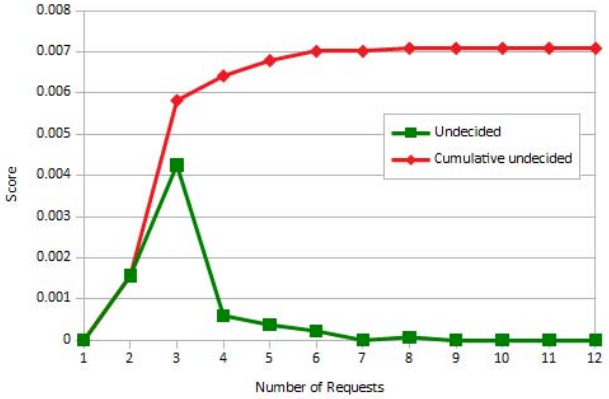

Fig. 3. Fraction of sessions undecided at each step and cumulatively.

All the performance scores flatten out at about step 8, where most sessions have already been classified.

V. Discussion and Conclusion

The experimental results indicate that the approach to online bot detection presented in this paper is effective. Some of its advantages are a simple implementation, which helps reducing the overload on a server, increases scalability, reduces the statistical model uncertainty by keeping the number of parameters low, therefore encouraging good generalisation; and a favourable observed error distribution with respect to the balance between false positives and false negatives, which is important in online applications to avoid disrupting the user's experience.

With regard to the measure of performance, it should be noted that the design decision to penalize all undecided sessions as errors is a natural consequence of the design goals. An alternative design choice could have been to defer the decision to an off-line phase. Although this choice is described in the literature and is less penalizing with respect to the performance indicators, it is incompatible with online operation.

The worst-case assumption of counting an undecided session as an error could also be replaced by deciding according to the prior probability, in this case by attributing all undecided sessions to class **zero** (human), or to the class with the most undecided sessions (in this case, **one** or bot).

Acknowledgment

This paper is based upon work from COST Action IC1406 High-Performance Modeling and Simulation for Big Data Applications (cHiPSet), supported by COST STSM grants.

References

[1] A. Balla, A. Stassopoulou, and M. D. Dikaiakos, "Real-time Web crawler detection," in *Proc. of the ICT'11*, 2011, pp. 428–432.

[2] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol - HTTP/1.0," IETF RFC 1945, May 1996.

[3] C. Bomhardt, W. Gaul, and L. Schmidt-Thieme, "Web robot detection - preprocessing Web logfiles for robot detection," in *New Developments in Classification and Data Analysis*, pp. 113–124, 2005.

[4] M. D. Dikaiakos, A. Stassopoulou, and L. Papageorgiou, "An investigation of Web crawler behavior: Characterization and metrics," *Comput. Commun.* 28(8), pp. 880–897, May 2005.

[5] D. Doran and S. S. Gokhale, "Web robot detection techniques: Overview and limitations," *Data Min. Knowl. Discov.* 22, pp. 183–210, 2011.

[6] D. Doran and S. S. Gokhale, "An integrated method for real time and offline Web robot detection," *Expert Systems* 33(6), pp. 592–606, 2016.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1," IETF RFC 2616, June 1999.

[8] T. Gržinić, L. Mršić, and J. Šaban, "Lino - an intelligent system for detecting malicious Web-robots," in *Intelligent Information and Database Systems*, 2015, pp. 559–568.

[9] W. Guo, S. Ju, and Y. Gu, "Web robot detection techniques based on statistics of their requested URL resources," in *Proc. of CSCWD'05*, vol. 1, May 2005, pp. 302–306.

[10] G. Jacob, E. Kirda, C. Kruegel, and G. Vigna, "PUBCRAWL: Protecting users and businesses from CRAWLers," in *Proc. of USENIX Security'12 Symposium*, ser. Security'12, 2012, pp. 25–25.

[11] Diederik P. Kingma, Jimmy Ba, "Adam: A Method for Stochastic Optimization," eprint arXiv:1412.6980, 2014.

[12] S. Kwon, M. Oh, D. Kim, J. Lee, Y.-G. Kim, and S. Cha, "Web robot detection based on monotonous behavior," in *Proc. of ISI'12*, vol. 4, 2012, p. 43–8.

[13] J. Lee, S. Cha, D. Lee, and H. Lee, "Classification of Web robots: An empirical study based on over one billion requests," *Comput. Secur.* 28(8), pp. 795–802, 2009.

[14] W.-Z Lu and S.-Z. Yu, "Web robot detection based on Hidden Markov Model," in *Proc. of ICCCAS'06*, vol. 3, 2006, pp. 1806–1810.

[15] S. Rovetta, A. Cabri, F. Masulli, G. Suchacka, "Bot or not? A case study on bot recognition from Web session logs," in *Proc. of WIRN'17*, 2018 (in press).

[16] C. H. Saputra, E. Adi, and S. Revina, "Comparison of classification algorithms to tell bots and humans apart," *JNIT* 4, pp. 23–32, 2013.

[17] D. S. Sisodia, S. Verma, and O. P. Vyas, "Agglomerative approach for identification and elimination of Web robots from Web server logs to extract knowledge about actual visitors," *JDAIP* 3, pp. 1–10, 2015.

[18] A. Stassopoulou and M. D. Dikaiakos, "Web robot detection: a probabilistic reasoning approach," *Comput. Netw.* 53(3), pp. 265–278, 2009.

[19] D. Stevanovic, A. An, and N. Vlajic, "Feature evaluation for Web crawler detection with data mining techniques," *Expert Syst. Appl.* 39(1), pp. 8707–8717, 2012.

[20] D. Stevanovic, N. Vlajic, and A. An, "Detection of malicious and non-malicious website visitors using unsupervised neural network learning," *Appl. Soft Comput.* 13(1), pp. 698–708, 2013.

[21] G. Suchacka, "Analysis of aggregated bot and human traffic on e-commerce site," in *Proc. of FedCSIS'14*, 2014, pp. 1123–1130.

[22] G. Suchacka and I. Motyka, "Efficiency analysis of resource request patterns in classification of Web robots and humans," in *Proc. of ECMS'18*, 2018, pp. 475–481.

[23] G. Suchacka and M. Sobków, "Detection of Internet robots using a Bayesian approach," in *Proc. of IEEE CYBCONF*, 2015, pp. 365–370.

[24] P.-N. Tan and V. Kumar, "Discovery of Web robot sessions based on their navigational patterns," *Data Min. Knowl. Discov.* 6(1), pp. 9–35, 2002.

[25] Udger, https://udger.com (visited: September 4, 2017).

[26] User-agents, http://www.user-agents.org (visited: September 4, 2017).

[27] A. Wald, "Sequential tests of statistical hypotheses," *Ann. Math. Statist.* 16(2), pp. 117–186, 1945.

[28] M. Zabihimayvan, R. Sadeghi, H. N. Rude, and D. Doran, "A soft computing approach for benign and malicious Web robot detection," *Expert Syst. Appl.* 87, pp. 129–140, 2017.

[29] I. Zeifman, "Bot traffic report 2016," Technical report, [Online]. Available: https://www.incapsula.com/blog/bot-traffic-report-2016.html.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, 12:2825–2830, 2011.