Learning Explainable Network Request Signatures for Robot Detection

Rajat Agarwal, Sharad Chitlangia, Anand Muralidhar, Adithya Niranjan
Abheesht Sharma, Koustav Sadhukhan and Suraj Sheth
Amazon Ads
{agrajat,chitshar,anandmur,aditmn,abheesht,koustav,sursheth}@amazon.com

ABSTRACT

Rapid growth of deep learning models in recent years for robot and fraud detection has led to significant improvement in precision and recall but has also created a challenge for explainability and trust in the model decisions. In this paper, we propose a scalable multi-tiered framework that generates explainable network request level signatures for crawler bots on a large e-commerce advertising program. Depending on the bot traffic distribution, the framework uses a combination of volumetric aggregation, decision trees and predictive deep learning models based on weak labels to generate precise and explainable bot signatures, achieving 87.9% coverage over a black-box crawler detection system comprising of multiple deep learning models and heuristic techniques. We further demonstrate that the learnt signatures are more robust in time when compared to traditional IP level bot denylists and reduce false negatives for the black-box crawler bot detection system. Explainable network signatures also enable manual inspection and help with attributing traffic to bot toolkits, which not only improves trust in the black-box system decisions but also provides insights into the evolving bot landscape.

ACM Reference Format:
Rajat Agarwal, Sharad Chitlangia, Anand Muralidhar, Adithya Niranjan, Abheesht Sharma, Koustav Sadhukhan and Suraj Sheth. . Learning Explainable Network Request Signatures for Robot Detection. In *3rd Workshop on AI-Enabled Cybersecurity Analytics - KDD 2023.* ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

Robotic traffic exists on the web for a wide range of reasons spanning across crawlers and scrapers, price grabbers, automation tools, DDoS attacks, ad fraud, etc. With advances in machine learning, robot and fraud detection models have evolved to use a large variety of input features, and modeling techniques have transitioned from classical tree-based models to sophisticated neural networks. While neural network based techniques lead to higher precision and recall due to their ability to model rich feature interactions, they make it more difficult to explain and analyze model decisions. Considering the high monetary impact of robotic traffic in online advertising, and absence of ground truth bot labels at scale, explainability is essential for increased trust in black-box model predictions and understanding evolving robotic behaviors.

We explore the problem of crawler bot detection on a large-scale e-commerce advertising program. Crawler traffic in advertising refers to robotic traffic that leads to large number of ad impressions but almost no clicks - leading to anomalously low click to impression ratio, also known as click-through rate (CTR). The bot detection system for any ad program comprises of multiple sophisticated machine learning models and techniques that classify an ad impression request as robotic or human. The goal of this paper is to build techniques that generate precise and human explainable rules (signatures) for bot classification and also maximize coverage of these rules over impressions marked as robotic by the black-box. We learn these explainable rules based on fields available in the network logs of the ad request because bots are known to exhibit unique signatures based on these network attributes. The rules can then be investigated manually for spoofing or can be attributed to known bot traffic generation toolkits. High coverage of these rules on bot traffic flagged by the black-box system not only increases human confidence in decisions of black-box machine learning models, but also allows us to build mitigation techniques for novel bots and toolkits.

One of the key challenges in generating rules from network logs is the high cardinality of the individual fields in the dataset, which can take millions of unique values. High cardinality features make learning rules computationally expensive and overly complex due to the combinatorial search space, even while using greedy search approximations like in a decision tree. In this paper, we present a scalable framework for generating explainable rules derived from high cardinality input fields in a binary classification setting for robot detection. The framework approaches rule generation in a multi-tier fashion depending on the traffic volume across features, and uses a combination of heuristics, decision tree and predictive modeling with deep neural networks. We benchmark the precision and coverage of the explainable rules generated by the framework on the bot traffic detected by black-box system.

The paper is structured as follows: Section 2 formalizes the problem statement, Section 3 describes the related work and Section 4 describes the multi-tier modeling framework for rule generation. This is followed by a description of experiments and results in Section 5. We highlight key learnings and insights from the generated explainable bot signatures in Section 6 and conclude in Section 7.

2 PROBLEM FORMULATION

We approach the problem of learning rules to explain binary classification decisions of a black-box crawler bot detection system consisting of multiple machine learning models and heuristics. Formally, let X be a set of all ad impression requests, and Z be an indicator variable representing if the ad impression was clicked on. The black-box crawler bot detection system generates a binary prediction P for all impressions in X, classifying them as robotic

AI-Enabled Cybersecurity Analytics - KDD 2023, August 7, 2023, Long Beach, CA

(0) or human (1). The goal of the rule generation module is to use fields in the network logs which represent every impression in X by high cardinality features $F_1, F_2, \ldots, F_n$, to generate high precision human understandable robotic rules that also maximize coverage over impressions marked as robotic by the black-box system. We note that the formulation does not make any assumptions on the feature set F being used by any of the models or heuristics in the black-box.

3 RELATED WORK

Robot detection. Traditional bot detection approaches use curated static lists of IPs and UserAgents (UA) based on industry-wide intelligence and historical bot patterns. Attempts to identify more granular entities like UA-IP combinations, UA regex based rule mining have been made, but such approaches suffer from low recall and precision because IPs can consist of mixed human and bot traffic, ISPs regularly rotate addresses and these fields can be easily spoofed or get stale. Compared to broad identifiers like IP, UA, etc., attributes extracted from network requests are very difficult to manipulate and spoof. Although there has been relatively less work on this due to lack of data, approaches like [6] extracted sub-strings of HTTP headers as signatures of fraudulent traffic. Similarly, [2] clustered HTTP headers based on a normalized form of string distances, with the cluster thresholds being manually tuned. More recent works approach bot detection as a real-time system that is capable of swiftly adapting to evolving bot and human traffic patterns by relying on weakly supervised deep learning based models [9], improving the overall precision and recall over traditional list and rule based systems but trading-off performance with explainability. Building on [9], [3] has shown that pre-training embeddings of network request attributes can enable better bot detection.

Explainable AI. Approaches to explain model decisions involve either computing feature importance [7, 10, 12] for individual examples or the overall model or distilling the model predictions into a set of human understandable rules. Using feature importance to explain model decisions in deployed settings has multiple challenges. First, the feature importance scores vary significantly based on the technique used to compute the scores - commonly known as the disagreement problem [5]. Second, for models with large number of input features, a large subset of features can get high importance scores, or in case of high cardinality features where embedding based inputs are preferred, high importance scores may be assigned to only specific embedding dimensions - leading to poor explainability. Finally, in deployed bot detection systems where potentially multiple models are used to detect different types of bots (example - IP based vs user-id based), one would have to use multiple feature importance techniques for different models, making it difficult to assign a single importance score to a feature that is shared across models. In contrast, rule mining based explainability techniques treat the underlying predictive model or a set of models as a black-box and work based on a fixed set of feature inputs and final black-box output predictions [1, 8, 13]. Rules despite being more explainable, can often have less coverage and efficacy as compared to predictive models - hence, there has been work towards exploring the use of externally available relevance information that can aid the creation of rules to explain neural network style predictive models [11].

4 MODELING FRAMEWORK

In this section, we describe the multi-tiered approach to learn explainable signatures (also referred as rules) on high cardinality inputs. To be able to classify a rule as a crawler, we estimate its click-through rate (CTR) and mark rules with anomalously low CTRs as crawlers. The framework utilizes the fact that feature value tuples $(F_1, F_2, ..., F_n)$ follow a long-tail frequency distribution when inputs are high cardinality. At the head of the distribution, where the traffic volume is large for each feature tuple, click-through rate (CTR) can be directly estimated with high confidence just by volumetric aggregation. For the tail, we present a scalable decision tree learning model that learns a series of rules to aggregate across multiple tail feature tuples to derive a high confidence CTR estimate. For the remaining tail feature tuples which could not be aggregated by the decision tree rules due to sparse traffic volumes and tree scalability constraints, we build a separate deep learning model to estimate the robotic probability with help of other auxiliary features and weak labels. We now provide detailed descriptions for each of the three modules.

4.1 Volumetric Aggregation

The simplest approach to generate rules is to enumerate all feature tuples $(F_1, F_2, ..., F_n)$ alongside their impression volumes and CTRs. Feature tuples that have more than a minimum volume of impressions and CTR less than a specified threshold are outputted as robotic rules. The CTR threshold is determined manually by trading off False Positive Rate (FPR) with coverage over the black-box model. This technique works well for feature tuples at the head of the traffic distribution where large volumes of traffic is observed per feature tuple and CTR estimates are highly confident, which is ensured by the minimum impression volume condition. Hence, the generated rules contain one equality clause for all features, which are combined using an AND condition. An example is given below:

$$F_1 = f_1 \text{ AND } F_2 = f_2 \text{ AND } F_3 = f_3 ... \text{ AND } F_n = f_n \quad (1)$$

4.2 Decision Tree based Rule Generation

While obvious bots with fixed features can be detected by volumetric aggregation, more sophisticated bots can rotate some or all of the feature values, requiring sub-tuple level aggregation to get reliable CTR estimates. Since the features have very high cardinality, the number of possible sub-tuple combinations is combinatorially large for enumeration. Hence, we resort to a greedy approximation to learn sub-tuple level rules by training a standard decision tree based classifier on all impressions X with labels as Z (is-Click). Each impression is represented by a concatenation of the one-hot encoding of the individual features F_i, and cross-entropy based information gain is used as the split criterion. CTR is then estimated at the leaf nodes, and paths to leaf nodes with CTR below a specific threshold are outputted as robotic rules. Hence the rules can now include both equality and non-equality clauses for individual features and may not contain all features. Example path to leaf node:

$$F_1 = f_1 \text{ AND } F_2 \neq f_2 \quad (2)$$

4.2.1 Scaling Decision Tree Learning. One of the key challenges in the above decision tree formulation is to scale the training of the decision tree on billion-scale ad impression data and large cardinality inputs. We use two key insights to achieve this. First, individual features despite being high cardinality, also follow a long tail frequency distribution. To be able to estimate CTR confidently at the leaf node, similar to volumetric aggregation, we lower bound the number of impressions allowed in the leaf node. Hence, given the tail distribution, majority of the values in the high cardinality feature will never be considered as candidate splits and can be dropped from the input one-hot representation by mapping them to a dummy value represented by the <*unk*> token. While, this significantly reduces the length of the feature vector input to the decision tree, we still have billions of rows for individual impressions. Here, we use the second insight, that multiple impression events have the same network features - allowing us to reduce the data to only two rows (one per label) for every unique feature tuple by adding a sample weight column representing the impression count of the $(F_1, F_2, ..., F_n, Z)$ combination post the one-hot indexing. The number of impressions at the leaf node is then computed as the sum of sample weights of rows classified to the particular leaf.

4.3 Predictive Model using Deep Learning

Since a minimum impression threshold was applied at the complete feature tuple level in volumetric aggregation and at sub-tuple level in the decision tree, there would still be remaining tail feature tuples which would not have been considered by either of the techniques. For such tuples, since CTR could not be estimated directly with high confidence, we build a predictive model to classify them as robotic or human. As building a predictive model on sparse features alone will overfit, we make use of auxiliary features about the ad request and weak labeling techniques to predict the robot probability of individual impressions. For rule generation, we aggregate the impression level robotic probabilities at unique network feature tuple level by computing the mean. The mean probabilities are then thresholded based on FPR, outputting a robotic list of tuples similar to the format used in volumetric aggregation. We now describe the training setup for the predictive model.

Features. Since the network features alone would lead to overfitting due to the high cardinality, we augment the ad request with a variety of features including: (1) Frequency and velocity counters for users' page requests over various time periods ranging from few seconds to hours; (2) User-entity counters indicating distinct users from an IP address to help disambiguate IPs with multiple users and normalize their impression volumes; (3) Time of ad impression; (4) Logged-in status of the user; (5) Embedding-based inputs for the users' identifier and network attributes with the embedding matrices being learnt end-to-end with the training objective. To handle the long tailed distributions, an out of vocabulary embedding is included in all embedding look-up tables.

Labels. Since accurate ground truth labels are unavailable at scale, we resort to weak supervision for labeling - where we identify high-hurdle activities that are more likely to be performed by humans than a bot. Specifically, we label ad requests that led to a purchase as human(1). To further increase human label density, we also label ad requests from customers with high RFM score as human. A high RFM score denotes that customers have purchased recently, frequently and with a high monetary value. All other requests are labeled as robotic(0).

Model. The embeddings based input are concatenated with other tabular features and passed to a three-layered feedforward neural network, with 1024 neurons each and ReLU activation in the hidden layers. The output of the final layer is passed through a sigmoid activation unit to get the robotic probability for an impression. To avoid overfitting, we apply L2 regularization with the regularization constant as 0.001 to all the embedding layers, and the second and third feedforward layer.

Training. Since our dataset is highly skewed towards the "robot" label (0), we assign a sample weight $w_i = \frac{C}{N_i}$ to the i^{th} sample, where C is a constant and N_i denotes the number of impressions in the $(hour_i, day_i, logged\_in_i, label_i)$ bucket. We use weighted binary cross entropy loss optimized using Adam [4] with learning rate $5e-5$ for training our model.

5 EXPERIMENTS

5.1 Dataset

We learn the signatures on 1 day of impression data generated from the ad program. Following the scaling technique mentioned in Section 4.2.1, the concatenated one-hot input feature vector length in the decision tree reduces by 6 orders of magnitude and the number of input rows reduce by 3 orders of magnitude.

5.2 Metrics

We define the following metrics to evaluate the bot detection and explainability efficacy for the proposed techniques:

(1) **Invalidation Rate (IVR)** This is defined as the fraction of total impressions marked as robotic by the algorithm.
(2) **False Positive Rate (FPR)** This is defined as the fraction of human impressions invalidated by the algorithm. Since we do not have ground truth labels, FPR is approximated by using purchasing users as a proxy for the distribution of human labels.
(3) **Black-Box Coverage** This indicates the fraction of impressions flagged as robotic by the black-box system that overlap with the rules generated by the proposed technique.

5.3 Results

Since crawler bots typically do not use customer accounts, we restrict our evaluation to non-loggedin traffic in this paper. The metrics for the individual techniques and the combined framework are reported in Table 1. We also report the exclusive invalidation rate for each technique, which refers to the invalid requests that were not detected by the other two techniques in the framework. We note that all reported IVR and FPR metrics in Table 1 are relative to the black-box baseline.

5.4 Discussion

Results in Table 1 show that the proposed framework was able to generate explainable rules that cover 87.91% of the bot traffic

Table 1: Performance Metrics relative to Black-Box

Algorithm	Relative IVR	Relative FPR	Relative Exclusive IVR	Black-Box Coverage
Volumetric Aggregation	47.37%	6.40%	1.48%	46.20%
Decision Tree	82.15%	8.14%	20.28%	81.06%
Deep Predictive Model	66.90%	8.14%	2.90%	65.58%
All Three Algorithms	89.79%	18.61%	N/A	87.91%

detected by the black-box system while incurring only 18.61% of the black-box FPR - indicating the highly precise nature of these rules. Volumetric aggregation has the lowest black-box coverage since it does not operate on the tail of the traffic distribution. It also has the lowest exclusive IVR because the obvious bots detected by volumetric aggregation can also be detected by the decision tree. Any exclusive IVR for volumetric aggregation is because of the lower impression volume threshold used in volumetric aggregation when compared to the decision tree because of scaling limitations. Decision tree is able to generate high precision rules with extremely high coverage due to the sub-tuple level aggregation capabilities. Exclusive IVR of the predictive deep learning model indicates that it detected bots at the tail of the distribution where even the sub-tuple level aggregation in the decision tree was sparse. While the expectation was that the predictive model should also have been able to detect majority of the bots flagged by the decision tree and volumetric aggregation, its lower invalidation rate suggests that more work is needed to refine the weak labeling strategy and training technique, which we plan to explore in future work. The proposed framework also achieves additional 1.88% relative exclusive IVR over the black-box indicating that the rules also help enhance overall bot detection by covering some of the false negatives of the black-box.

6 EXPLAINABILITY ANALYSIS

6.1 Temporal Durability of Network Signatures

The key reason for selecting network attributes to generate explainable signatures is their temporal durability - that is, bots typically have a unique network attribute signature while they rotate or spoof fields like IP Addresses and UserAgents. Table 2 demonstrates the temporal durability of the rules generated by the proposed techniques on a given day, over a span of 1 week. We compare this with durability of robotic IPs generated via the deep predictive model - as IP was a feature in the model, robotic IPs can be generated by aggregating the robotic probabilities at IP level instead of network features.

We observe that the IP-based list significantly deteriorates in invalidation rate, indicating bots rotate their IP address to evade detection. The invalidation rate for volumetric aggregation and decision tree based rules increases or remains flat, as these are high confidence rules which continue to detect bots even as the IP rotates.

Table 2: Temporal IVR relative to D^{th} day IVR of Signatures For Demonstrating Durability of Network Features

Algorithm	Signature	D+1	D+2	D+3	D+4	D+5	D+6
Deep Predictive Model	IP Address	-32.1%	-42.3%	-44.9%	-46.5%	-47.9%	-48.7%
Volumetric Aggregation	Network Features	-4.1%	+17.4%	+18.6%	+14.9%	+13.0%	+12.9%
Decision Tree	Network Features	-5.3%	-1.8%	-1.0%	-1.1%	-0.6%	-1.3%
Deep Predictive Model	Network Features	-25.7%	-8.5%	-7.7%	-10.1%	-12.5%	-12.8%

The invalidation rate for network rules from the predictive model drops slightly because they cover sparse feature tuples suggesting that daily model retraining and inference is required to maintain high explainability on sparse signatures.

6.2 Bot Toolkit Identification

With explainable signatures enabling manual inspection, we are able to attribute network signatures to different types of bot toolkits, providing insights into how robotic traffic was generated. We categorize various bot toolkits identified in an initial manual inspection as follows:

(1) **Automated Browsers** Web scraping and crawling tasks commonly use browser instances running in headless mode, which means the browser runs without a visible browser window at much lower compute resources. Few frameworks also provide a programmatic interface to control and interact with headless browsers.
(2) **HTTP Libraries** Most modern programming languages like Python and Java have libraries that enable users to make HTTP requests with options for setting request headers, handling redirects, and handling response data. This makes them a popular choice among developers to build various crawling frameworks, which in turn are used by bot operators. We found that a subset of these bots could also be characterized as malicious as they actively try to hide themselves using various techniques such as User-Agent spoofing.
(3) **Crawling as a Service** Bot operators crawl e-commerce sites to collect product catalog, reviews and price data to generate analytics and intelligence for their customers. Analyzing the network attributes of such bot requests revealed usage of customized Python frameworks.

7 CONCLUSION AND FUTURE WORK

We presented a scalable multi-tiered framework to learn explainable rules for a black-box crawler bot detection system, that works across different parts of the traffic distribution. We show that the explainable rules generated are not only highly precise and have high coverage but also provide a temporally durable mechanism to identify bots while reducing the overall false negatives. Having high coverage explainable network signatures also help us to classify and understand the bot landscape, thereby increasing trust in the black-box machine learning systems. In future work, we plan to experiment with improved weak supervision and self-supervision techniques to enhance the performance of the predictive model and expand the decision tree framework to consume features beyond network attributes to generate more precise and higher recall signatures.

REFERENCES

[1] Boz, O. Extracting decision trees from trained neural networks. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2002), KDD '02, Association for Computing Machinery, p. 456–461.

[2] Cai, T., and Zou, F. Detecting http botnet with clustering network traffic. In *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing* (2012), pp. 1–7.

[3] Chitlangia, S., Muralidhar, A., and Agarwal, R. Self supervised pre-training for large scale tabular data. In *NeurIPS 2022 Workshop on Table Representation Learning* (2022).

[4] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014).

[5] Krishna, S., Han, T., Gu, A., Pombra, J., Jabbari, S., Wu, S., and Lakkaraju, H. The disagreement problem in explainable machine learning: A practitioner's perspective. *CoRR abs/2202.01602* (2022).

[6] Li, K., Liu, C., and Cui, X. Poster: A lightweight unknown http botnets detecting and characterizing system. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2014), CCS '14, Association for Computing Machinery, p. 1454–1456.

[7] Lundberg, S. M., and Lee, S. A unified approach to interpreting model predictions. *CoRR abs/1705.07874* (2017).

[8] Luo, G. Automatically explaining machine learning prediction results: A demonstration on type 2 diabetes risk prediction. *Health Information Science and Systems 4* (03 2016).

[9] Muralidhar, A., Chitlangia, S., Agarwal, R., and Ahmed, M. Real-time detection of robotic traffic in online advertising. In *IAAI 2023* (2023).

[10] Ribeiro, M. T., Singh, S., and Guestrin, C. "why should I trust you?": Explaining the predictions of any classifier. *CoRR abs/1602.04938* (2016).

[11] Srinivasan, A., King, R. D., and Bain, M. E. An empirical study of the use of relevance information in inductive logic programming. *J. Mach. Learn. Res. 4*, null (dec 2003), 369–383.

[12] Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. *CoRR abs/1703.01365* (2017).

[13] Zhou, Z.-H., Jiang, Y., and Chen, S.-F. Extracting symbolic rules from trained neural network ensembles. *AI Commun. 16*, 1 (jan 2003), 3–15.