# Botcha: Detecting Malicious Non-Human Traffic in the Wild

Sunny Dhamnani[a], Ritwik Sinha[b], Vishwa Vinay[a], Lilly Kumari[d] and Margarita Savova[e]

[a]Adobe Research, India
[b]Adobe Research, United States
[d]University of Washington, Seattle, United States
[e]Adobe Systems, United States

**Abstract**

Malicious bots make up about a quarter of all traffic on the web and degrade the performance of personalization and recommendation algorithms that operate on e-commerce websites. Positive-Unlabeled learning (PU learning) provides the ability to train a binary classifier using only positive (P) and unlabeled (U) instances. The unlabeled data comprises both positive and negative classes. It is possible to find labels for strict subsets of non-malicious actors, e.g., the assumption that only humans purchase during web sessions, or clear CAPTCHAs. However, finding signals of malicious behavior is almost impossible due to the ever-evolving and adversarial nature of bots. Such a set-up naturally lends itself to PU learning. Unfortunately, standard PU learning approaches assume that the labeled set of positives are a random sample of all positives, this is unlikely to hold in practice. In this work, we propose two modifications to PU learning that make it more robust to violations of the *selected-completely-at-random* assumption, leading to a system that can filter out malicious bots. In one public and one proprietary dataset, we show that proposed approaches are better at identifying humans in web data than standard PU learning methods.

**Keywords**

Positive unlabeled learning, biased sampling, non-human agents, unlabeled data, malicious bot, web traffic

## 1. Introduction

Non-Human Traffic or traffic generated by robots (or bots) is estimated to constitute close to half of all web traffic [1]. Some bots have a legitimate purpose (e.g. web crawlers) while others try to intrude the systems with malicious intent. It is estimated that half of all bot traffic has a malicious intent [1]. Good bots identify themselves but malicious bots have an incentive to spoof their user agents and behave like humans. Malicious bots may be designed to generate fake reviews, scrape price or content, crack credentials, infiltrate payment systems, defraud advertisers, or spam online forums. Recommendation and personalization systems are particularly vulnerable to bot activity [2].

*OHARS'20: Workshop on Online Misinformation- and Harm-Aware Recommender Systems, September 25, 2020, Virtual Event*

EMAIL: dhamnani.sunny@gmail.com (S. Dhamnani); risinha@adobe.com (R. Sinha); vinay@adobe.com (V. Vinay); lkumari@uw.edu (L. Kumari); mgalabov@adobe.com (M. Savova)

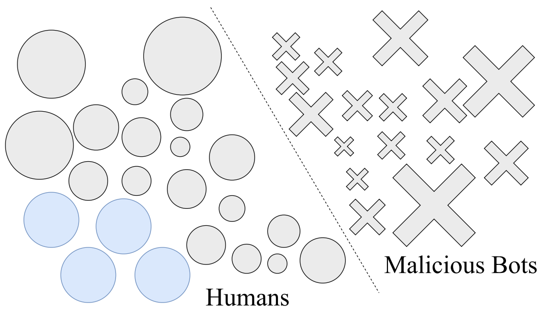

**Figure 1:** Problem set-up: circles denote humans and crosses denote bots. Only the blue circles are known to be humans, the true label for all grey instances is unknown. The fact that larger circles are more likely to be blue denotes that the observation's attributes determine their likelihood of being selected (labeled). Under SCAR (Selected Completely at Random) violation, the classification goal is to identify the dashed dividing line between the two classes.

The major challenge in building machine learning (ML) models to detect bad bots is getting labeled data. In this context, ML methods that aim to learn from positive and unlabeled data (PU learning) provide promise [3]. PU learning learns from data where only a subset of one class is labeled. We explore an application of PU learning to malicious non-human traffic detection on the web. Considering humans as the positive class, we can identify positive instances by assuming that only humans purchase on e-commerce websites, clear CAPTCHAs, or visit from validated IP addresses.

Current PU learning frameworks assume that the labeled subset of the positive class is *Selected Completely at Random (SCAR)* from the positive class, where the labeling mechanism does not depend on the attributes of the instance [3]. That is, the labeled subset of humans is not influenced by the features of the observations. Unfortunately, such an assumption is hard to justify in practice. For example, it is reasonable to expect that not all human visitors to an e-commerce website are equally likely to make a purchase. This requires us to revisit the PU framework to handle problems where the random sampling assumption is violated. Figure 1 describes the problem we are addressing.

In this work, we address the question of classifying a web session as originating from a human surfer or a robot, using PU learning. Our contribution includes two novel models to handle biased sampling within the positive class, one of which is a scalable version of the proposals in [4]. In our experiments, positive-unlabeled scenarios are artificially created in a publicly available intrusion detection dataset. We notice that the proposed approaches perform better than existing PU learning models [3, 5]. In a proprietary e-commerce dataset, our methods work well in distinguishing humans from bots. We call our framework "Botcha". Given the limited need for labeled data, it is readily applied in the wild. Filtering out all bot traffic allows recommendation and personalization systems to learn from unbiased data from real human activity.

## 2. Related Work

Malicious non-human activity on the web has been observed in the context of fake reviews, information theft, the spread of misinformation, spam on social networks, and click fraud in advertising [6, 7]. Given the diverse and often adversarial nature of web fraud, it is imperative to find new strategies to detect bots. In such dynamic circumstances, data-driven strategies hold promise.

While there has been some work to build recommendation systems that are robust to adversarial attacks [2, 8], in this work we aim to filter out all bot traffic to provide unbiased data to the recommendation and personalization systems to learn from. To classify a visitor as a bot or human, the standard machine learning strategy requires representative instances from both classes and building a supervised learning model that can differentiate between them. Due to limited labeled data for bot detection, alternative data-efficient strategies have also been investigated. Semi-supervised learning has been applied to the bot detection problem [9]. Unfortunately, while it is reasonable to expect that we have a reliable subset of known humans, bots on the web are adversarial, ever-evolving and hard to sample from. This renders semi-supervised learning limited in scope.

PU Learning requires only a subset of one of the two classes to be labeled. Hence, PU learning is appealing in the bot detection problem where we can assume that a subset of humans is labeled. Early work in [10] and [3] has shown how PU learning can achieve the effectiveness of standard supervised learning. We believe that the PU learning framework is natural for use in a variety of fraud detection applications on the web.

Empirical success in a variety of scenarios has led to a recent focus on the class of PU learning algorithms [11]. Unfortunately, most prior work in this area assumes that the labeled points are randomly sampled from the positive class. This assumption is referred to as Selected Completely at Random [3, 4]. That is to say, the positively labeled instances in the dataset are a random unbiased sample of the universe of positive instances, and are not a function of the attributes of the data point. To allow the building of PU learning-based models in scenarios where this is an unrealistic assumption, we build on prior work by Bekker et al. (2019) [4]. However, it has two primary challenges. First, the model strategy presented in [4] requires the analyst to decide on a set of features to compute the propensity score. Second, the proposal requires optimization using an Expectation-Maximization (EM) algorithm. Unfortunately, the EM Algorithm is known to be slow to converge [12]. Given that we would like to apply this to a scenario with tens of millions of data points and hundreds of features, this presents certain challenges in the direct application of this to our work. To test our proposed algorithms, we first conduct a series of simulation experiments on standard supervised learning datasets representing different fraud-like setups. We artificially hide the true labels which we then hope to recover via the learning algorithm, thereby showing the viability of our methods.

## 3. Models

We first describe the notations and then briefly review PU learning work in [3] (Section 3.2). In section 3.3 and 3.4, we describe the proposed approaches, which are the main contributions of the paper.

## 3.1. Notation & Prerequisites

To distinguish humans from bots we need to learn a classifier that generates the probabilities $p(y|\boldsymbol{x})$. Here $y \in \{0, 1\}$ denotes if the observation was generated by a human ($y = 1$) or bot and $\boldsymbol{x}$ is the corresponding feature vector. The dataset for PU learning are instances $(\boldsymbol{x}, y, s)$ from a space $\mathcal{X} \times \mathcal{Y} \times \mathcal{S}$, where $\mathcal{X}$ and $\mathcal{Y}$ denote the feature and label space respectively. The binary variable $s$ represents if the instance is labeled. Since only positive instances (humans) are labeled, $p(y = 1|s = 1) = 1$. Marginalizing $p(s = 1|\boldsymbol{x})$ over $y$, we get:

$$p(s = 1|\boldsymbol{x}) = p(s = 1|y = 1, \boldsymbol{x}) \times p(y = 1|\boldsymbol{x}) + p(s = 1|y = 0, \boldsymbol{x}) \times p(y = 0|\boldsymbol{x})$$

Now, $p(s = 1|y = 0, \boldsymbol{x}) = 0$ since only the positive instances are labeled. This leads to

$$p(y = 1|\boldsymbol{x}) = \frac{p(s = 1|\boldsymbol{x})}{p(s = 1|y = 1, \boldsymbol{x})} \tag{1}$$

Equation (1) forms the basis of all models which we describe next.

## 3.2. Vanilla Model (EAM)

The work by Elkan and Noto is based on the SCAR assumption. The approach assumes that the labeled positive instances were chosen uniformly at random from the universe of positive instances [3]. Formally this means $p(s = 1|y = 1, \boldsymbol{x}) = p(s = 1|y = 1)$, i.e., the sampling process is independent of $\boldsymbol{x}$. We can rewrite equation (1) as

$$p(y = 1|\boldsymbol{x}) = \frac{p(s = 1|\boldsymbol{x})}{c}\text{, with } c = \frac{1}{n} \sum_{\langle \boldsymbol{x}, y=1 \rangle} P(s = 1|\boldsymbol{x}). \tag{2}$$

The constant $c$ represents the fraction of labeled positive points and $n$ is the size of the labeled set. Note that the numerator can be obtained by training a classifier that separates the labeled ($s = 1$) points from the unlabeled ($s = 0$). Similarly, $c$ can be estimated using this trained classifier and a validation set. Averaging predicted scores of known positives in the validation set gives an estimate for $c$. We refer to this model as *Elkan's Assumption Model* (EAM) in our experiments and it forms the baseline for our methods. For a detailed discussion, we refer the readers to [3].

## 3.3. Modified Assumption Model (MAM)

The SCAR assumption above enables the building of PU Learning models for a range of scenarios. However, we believe that this is an unrealistic assumption and argue that explicitly accounting for selection bias for the known positives allows us to build models that are more aligned to the data.

We propose *Modified Assumption Model* (MAM), geared towards practical cases where labeling is performed via a stratified procedure. Instead of using the SCAR assumption, we make a more lenient assumption that known positives come from two sub-groups, where for one the sampling depends on $\boldsymbol{x}$ and the other is independent of $\boldsymbol{x}$.

We introduce a new binary variable $b \in \{0, 1\}$ that indicates which of the two sub-groups a given labeled instance ($s = 1$) comes from. So, $b = 0$ indicates that value of $s$ is independent of $\boldsymbol{x}$, whereas $b = 1$ implies that value of $s$ is dependent on $\boldsymbol{x}$. Marginalizing over b, we get:

$$
\begin{aligned}
p(s = 1|y = 1, \boldsymbol{x}) = {} & p(s = 1|y = 1, b = 1, \boldsymbol{x}) \times p(b = 1|y = 1, \boldsymbol{x}) \\
& + p(s = 1|y = 1, b = 0, \boldsymbol{x}) \times p(b = 0|y = 1, \boldsymbol{x})
\end{aligned}
$$

Since $s$ is independent of $\boldsymbol{x}$ when $b = 0$, so $p(s = 1|y = 1, b = 0, \boldsymbol{x}) = c$ and given that $p(b = 0|y = 1, \boldsymbol{x}) = 1 - p(b = 1|y = 1, \boldsymbol{x})$, we can re-write above equation as

$$
p(y = 1|\boldsymbol{x}) = \frac{p(s = 1|\boldsymbol{x})}{c + p(b = 1|y = 1, \boldsymbol{x}) \times (1-c)}, \text{ with } c = \frac{1}{n} \sum_{\langle \boldsymbol{x}, y=1, b=0 \rangle} P(s = 1|\boldsymbol{x}) \tag{3}
$$

Similar to EAM the numerator can be obtained by training a classifier that separates the labeled ($s = 1$) points from the unlabeled ($s = 0$). The denominator model can be trained using $b = 1$ and $b = 0$ sets (note that points in these sets are labeled and positive, i.e., $s = 1$ and $y = 1$). The constant $c$ can be estimated by averaging the scores predicted by the numerator model for instances with $b = 0$ in the validation set. If $p(b = 1|y = 1, \boldsymbol{x}) = 0$ for all data points, i.e., sampling is independent of $\boldsymbol{x}$, we recover EAM from MAM. Our MAM proposal closely relates to the proposals made in [4], however, the algorithm in [4] does not scale to large scale datasets.

### 3.4. Relaxed Assumption Model (RAM)

The most general model, referred as *Relaxed Assumption Model* (RAM), does not make any assumption about $s$ being independent of $\boldsymbol{x}$. Instead, we attempt to model this process explicitly, i.e., we build a model for $p(s = 1|y = 1, \boldsymbol{x})$ - the denominator in equation (1). We first acquire a set of positive unlabeled instances of $y = 1$ with $s = 0$ and then utilize standard binary classification methods to distinguish $s = 0$ from $s = 1$ amongst the positive instances.

We propose the use of a nearest-neighbor based method that finds points in the dataset that are *close* to the known positives but are not in the sampled set ($s = 1$). Since any point outside the sampled set is $s = 0$, the nearest neighbor to a ($y = 1, s = 1$) point not in this set is implicitly taken to be ($y = 1, s = 0$). Note that this assumption may not always be true. As in the other models, we aim to find techniques that are robust even when the modeling assumption may be wrong. It is important to note that we do not alter the numerator in equation (1) and hence training classifier for numerator remains identical to EAM.

## 4. Experiments and Results

### 4.1. Simulated Experiments on a Public Dataset

In our first set of experiments, we artificially create PU learning datasets by hiding the ground truth labels of a labeled dataset during training. We then evaluate the trained model on a labeled test set. The simulations primarily involve controlling the subset of positive data points that are labeled for training, and all the other instances are unlabeled. The simulated datasets having varying degrees of "randomness", one of the extremes is a completely random subset of positive

**Table 1**
Test-set performance on public dataset. The best algorithm in each column is colored blue and second best is light blue. RAM and MAM perform significantly better when SCAR assumption is violated (low randomness). EAM only provides a marginal improvement over RAM when the known positives are a random subset from positive class.

Increasing randomness →

| | Method | Mixing $m = 0$ | | Mixing $m = 30$ | | Mixing $m = 70$ | | Mixing $m = 100$ | |
|---|---|---|---|---|---|---|---|---|---|
| | | AUC | Pr@Recall99 | AUC | Pr@Recall99 | AUC | Pr@Recall99 | AUC | Pr@Recall99 |
| **topper** $t = 0.90$ | Biased SVM [5] | 0.705 | 0.524 | 0.705 | 0.576 | 0.688 | 0.535 | 0.689 | 0.560 |
| | EAM [3] | 0.757 | 0.719 | 0.760 | 0.751 | **0.776** | **0.751** | **0.792** | **0.697** |
| | MAM (proposed) | 0.811 | 0.724 | 0.761 | 0.736 | 0.778 | 0.737 | 0.701 | 0.636 |
| | RAM (proposed) | **0.897** | **0.724** | **0.837** | **0.756** | 0.770 | 0.743 | 0.765 | 0.669 |
| **topper** $t = 0.925$ | Biased SVM [5] | 0.624 | 0.517 | 0.691 | 0.512 | 0.666 | 0.519 | 0.669 | 0.513 |
| | EAM [3] | 0.761 | 0.730 | 0.761 | 0.751 | **0.774** | **0.747** | **0.791** | **0.701** |
| | MAM (proposed) | 0.831 | 0.737 | 0.792 | 0.752 | 0.743 | 0.717 | 0.721 | 0.682 |
| | RAM (proposed) | **0.906** | **0.773** | **0.812** | **0.767** | 0.764 | 0.745 | 0.748 | 0.700 |

samples (satisfying SCAR perfectly). The other is extreme is a carefully crafted subset of positive samples where SCAR assumption is violated.

**Public Dataset:** We use the KDDCUP'99 dataset (NSL-KDD Dataset), a widely adopted labeled dataset for network intrusion detection. The train and test datasets have a total of 148, 517 records with 43 features each. To get around known problems with the dataset [13], we merge the given train and test records, which we then re-split into the train, validation and test sets in an 80:10:10 proportion. Overall, the dataset contains 71, 463 intrusive sessions (all intrusions are bot-generated) while the rest are legitimate sessions.

**Data Simulations:** The process of creating artificial datasets involves hiding the labels for all negative points and a proportion of the positive points. We sample a labeled subset of positive data points to create a known subset of positives. We first build a supervised classifier to score each data point. The classification task here is to distinguish intrusive vs legitimate sessions and the score is the predicted class probability. We use this score to introduce sampling bias in creating the known subset of positives. Using a Random Forest classifier, we achieve an AUC (area under ROC curve) value of 0.9921 on the training data and 0.9911 on the test data. We then curate different PU learning datasets by performing sampling over the scored data points by controlling two parameters as described below.

*1. Topper*: This parameter is used to introduce sampling bias by selecting only those positive points whose prediction score (using the supervised model) is higher than the $t^{\text{th}}$ quantile of all positive labeled points. This selection of the top fraction of positives introduces a sampling bias since we are only selecting points with a high score. The idea is to capture spread within the positive class, and one meaningful scale is to use the estimated probability that a point is positive, given its features. Note that sampling is only done for positive class, the labels for all negative points are hidden.

*2. Mixing*: This parameter controls 'randomness' for the known subset of positives. After creating a sample of known positives based on the topper parameter, at value $m$ we swap $m\%$ of the selected points with points from the positive set, the swapping is done with replacement. As we move from $m = 0$ to $m = 100$ we decrease the sampling bias in the set and correspondingly increase the randomness. A mixing of 100% means SCAR is completely satisfied.

The subset obtained at a particular value of $t$ and $m$ is the known labeled subset of positives, and the remaining points (all negatives and the unsampled positives) are treated as unlabeled. With distinct values of $t$ and $m$ we obtain different simulated datasets. At a particular value of the topper parameter ($t$), with $m = 100$, we get a completely random sample of positive class (satisfying SCAR), on the other end with $m = 0$ we get an extremely biased sample, containing only high scoring points. When $m < 100$ the sampling is not completely random and depends on the score of the supervised model that uses all the features $\boldsymbol{x}$. Consequently, the sampling variable $s$ is not independent of $\boldsymbol{x}$ and the dataset does not align with the assumption of Elkan and Noto. We show that in cases of biased sampling, the proposed methods outperform the baseline approaches that rely on SCAR assumption.

**Results on simulated datasets:** We train MAM and RAM and compare against the baselines - EAM [3] and biased SVM [5] - on simulated datasets with varying degrees of randomness. For uniformity, we use Random Forest as the base classifier for all three methods EAM, MAM and RAM. Biased SVM uses a SVM formulation [5].

The performance metrics are AUC (area under ROC curve) and Precision@Recall99, precision when 99% of known positives in the validation set are classified correctly. Unlike the standard '0.5' threshold for classification, we set the classification threshold such that 99% of the legitimate sessions (positives) are correctly classified as legitimate. This is particularly important since in real systems we do not wish to interrupt legitimate users with any scrutiny. And so, Precision@Recall99 is an important metric to consider.

The results for the simulated experiments are shown in Table 1. When sampling is extreme, *towards the left with smaller mixing parameter*, RAM and MAM perform significantly better than the EAM along the two evaluation metrics. With more randomness (increasing mixing), EAM beats other methods but our proposed RAM still has competitive performance. Biased SVM performs poorly throughout. This shows that in extremely biased situations the proposed models MAM and RAM provide significant improvements by explicitly accounting for the sampling bias. On the other hand, EAM provides slight improvement at high mixing (random sample) since it is tailored specifically for scenarios when the SCAR assumption holds.

## 4.2. Application to Real E-Commerce Data

This section describes the application of RAM to a proprietary dataset from the traffic logs of an e-commerce website.

**Data Description:** The data contains a record for every page request, here referred to as a 'hit'. We consider one week and collapse these records into 'sessions' for each user. A session combines a series of hits made by a user. The session ends with 30 minutes of inactivity. Overall we identify 3.6 million unique visitors from 6 million sessions, and more than 100 million hits. The task is to label a session as arising from a human or a bot. The sessions from legitimate bots are filtered out using user-agent strings. The feature representation of all sessions utilizes a standard set of technology (e.g. browser and device types), behavioral (e.g. time between hits) and session related (e.g. timezone and time-of-day). Since this is an e-commerce website, we also have information as to whether a particular session resulted in a purchase. This information is leveraged to build our partial set of positives. The details are presented next.

**Known subset of positives:** Out of the 6 million sessions, $36k$ (0.6%) sessions are *purchase*

**Table 2**
Test-set observations for E-commerce dataset.

| Class of sessions | No. of sessions | No. predicted human | % predicted human |
|---|---|---|---|
| Positive | 74k | 73k | ~ 99% |
| Negative | 24k | 608 | ~ 2.5% |
| Unlabeled | 1.08M | 890k | ~ 82% |
| Total | 1.18M | 965k | ~ 82% |

sessions and 360$k$ (6%) sessions belong to an identified purchaser. We label this 6% of sessions as positive (Human class). The dataset is then split into the train, test, and validation sets in an 80:10:10 ratio for modeling purposes.

**Partially labeled test dataset:** To validate our approach, we split the test data into 3 groups of points and observe the distribution of prediction scores across these classes. This split is based on heuristics which we describe next. *Positive data points*: The subset of sessions that had a user corresponding to a purchase session in the training dataset. *Negative data points*: The subset of sessions which have been originated from AWS/Azure servers are tagged as negative. The assumption being that browsing sessions originating from these cloud environments are unlikely to be initiated by humans. The set of AWS/Azure IPs are publicly available [14, 15]. *Unlabeled data points*: The set of sessions which are neither tagged as positive nor negative.

It is important to note that all points during model training had the label of 'Positive' or 'Unlabeled'. The 'Negative' label is only used for validation. Also notice that the set of known positives is neither complete (not all Humans purchase), nor is it an unbiased sample (different users have varying propensities for purchases).

Using the validation set, we identify a threshold that captures 99% of positive labels. The output score of the RAM model is converted into a boolean *is-human* label by using this threshold. Table 2 shows the break-up of the traffic in the dataset and how RAM classifies points from each of these classes. As seen in the table, we misclassify only a few negatively labeled sessions (<3%) and in total, we have close to 82% human traffic as reported by this model. We expect high human traffic since the website has strict login requirements for accessing their content. Additionally, we observe a stark separation in the prediction scores for positive and negative classes. Most positive samples had a score close to 1, while negatives were scored close to 0.

## 5. Conclusions

In this paper, we have addressed the problem of detecting non-human traffic using positive and unlabeled data. Providing recommendation and personalization systems unbiased data to learn from, leads to a better experience for the end-customer. We specifically accounted for the *selected completely at random* assumption in standard PU Learning methods and conducted simulation studies for validation. We also evaluated our most general model, RAM, on a large real word e-commerce dataset. Given the scale of fraud due to bots, such bot detection systems have a clear utility. The methods described in this paper show promising results in addressing the endemic bot problem.

## References

[1] D. Networks, 2020: Bad Bot Report | IT Security's Most In-Depth Analysis on Bad Bots, https://bit.ly/2Azqx3d, 2020. Accessed: 2020-05-15.

[2] S. Zhang, Y. Ouyang, J. Ford, F. Makedon, Analysis of a low-dimensional linear model under recommendation attacks, in: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, 2006, pp. 517–524.

[3] C. Elkan, K. Noto, Learning classifiers from only positive and unlabeled data, in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008, pp. 213–220.

[4] J. Bekker, P. Robberechts, J. Davis, Beyond the selected completely at random assumption for learning from positive and unlabeled data, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2019, pp. 71–85.

[5] B. Liu, Y. Dai, X. Li, W. S. Lee, P. S. Yu, Building text classifiers using positive and unlabeled examples, in: Data Mining, 2003. ICDM 2003. Third IEEE International Conference on, IEEE, 2003, pp. 179–186.

[6] H. Gao, M. Tang, Y. Liu, P. Zhang, X. Liu, Research on the security of microsoft's two-layer captcha, IEEE Transactions on Information Forensics and Security 12 (2017) 1671–1685.

[7] O. Stitelman, C. Perlich, B. Dalessandro, R. Hook, T. Raeder, F. Provost, Using co-visitation networks for detecting large scale online display advertising exchange fraud, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13, 2013, pp. 1240–1248.

[8] X. He, Z. He, X. Du, T.-S. Chua, Adversarial personalized ranking for recommendation, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 355–364.

[9] Y. Li, O. Martinez, X. Chen, Y. Li, J. E. Hopcroft, In a world that counts: Clustering and detecting fake social engagement at scale, in: Proceedings of the 25th International Conference on World Wide Web, WWW '16, 2016, pp. 111–120.

[10] W. S. Lee, B. Liu, Learning with positive and unlabeled examples using weighted logistic regression, in: In Proceedings of the 20th International Conference on Machine Learning, ICML'03, 2003, pp. 448–455.

[11] J. Bekker, J. Davis, Learning from positive and unlabeled data: A survey, Machine Learning 109 (2020) 719–760.

[12] F.-X. Jollois, M. Nadif, Speed-up for the expectation-maximization algorithm for clustering categorical data, Journal of Global Optimization 37 (2007) 513–525.

[13] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA), 2009, pp. 1–6.

[14] AWS, AWS IP address ranges, https://amzn.to/2z2Ql7h, 2020. Accessed: 2020-04-27.

[15] Microsoft, Microsoft azure datacenter IP ranges, https://bit.ly/36aMDon, 2017. Accessed: 2020-04-27.