Predictive Response Optimization: Using Reinforcement Learning to Fight Online Social Network Abuse

Garrett Wilson
Meta Platforms, Inc.

Geoffrey Goh
Meta Platforms, Inc.

Yan Jiang
Meta Platforms, Inc.

Ajay Gupta
Meta Platforms, Inc.

Jiaxuan Wang
Meta Platforms, Inc.

David Freeman
Meta Platforms, Inc.

Francesco Dinuzzo
Meta Platforms, Inc.

Abstract

Detecting phishing, spam, fake accounts, data scraping, and other malicious activity in online social networks (OSNs) is a problem that has been studied for well over a decade, with a number of important results. Nearly all existing works on abuse detection have as their goal producing the best possible binary classifier; i.e., one that labels unseen examples as "benign" or "malicious" with high precision and recall. However, no prior published work considers what comes next: what does the service actually *do* after it detects abuse?

In this paper, we argue that *detection* as described in previous work *is not the goal* of those who are fighting OSN abuse. Rather, we believe the goal to be *selecting actions* (e.g., ban the user, block the request, show a CAPTCHA, or "collect more evidence") that *optimize a tradeoff* between harm caused by abuse and impact on benign users. With this framing, we see that enlarging the set of possible actions allows us to move the Pareto frontier in a way that is unattainable by simply tuning the threshold of a binary classifier.

To demonstrate the potential of our approach, we present *Predictive Response Optimization (PRO),* a system based on reinforcement learning that utilizes available contextual information to predict future abuse and user-experience metrics conditioned on each possible action, and select actions that optimize a multi-dimensional tradeoff between abuse/harm and impact on user experience.

We deployed versions of PRO targeted at stopping automated activity on Instagram and Facebook. In both cases our experiments showed that PRO outperforms a baseline classification system, reducing abuse volume by 59% and 4.5% (respectively) with no negative impact to users. We also present several case studies that demonstrate how PRO can quickly and automatically adapt to changes in business constraints, system behavior, and/or adversarial tactics.

1 Introduction

Online Social Networks (OSNs) connect billions of people around the world, allowing them to share content, engage in discussions, learn about events and issues, find employment, buy and sell goods, and undertake any number of other useful activities. However, the very utility and popularity of OSNs attracts malicious actors who want to take advantage of the OSN's users for their own (usually monetary) gain. As a result, spam [11], phishing [36], fake engagement [22], abusive accounts [80], and data scraping [56] have become important problems for both academia and the social networks themselves.

The traditional approach to fighting OSN abuse has been to build *binary classifiers* that distinguish "benign" content, entities, or actions from "malicious" ones. The OSN then runs these classifiers either synchronously or asynchronously and blocks or removes the detected abuse. A great deal of research has gone into designing better and better classifiers (see Section 2.1 for a survey), but even the best known classifier will make thousands to millions of mistakes when run on *every* account or action in a large OSN, leading to degraded experiences for the users who encounter these mistakes.

Most binary classifiers produce classifications by comparing a numerical score against a threshold. For instance, scores above the threshold will result in classifying a piece of content as "abusive." Tuning the threshold allows the operator to control the tradeoff between abuse and user experience in a rudimentary way. The OSN can set a "budget" on the model's precision (e.g., choose the lowest threshold such that at least $X\%$ of spam is classified correctly) or the false positive rate (e.g., choose the lowest threshold such that at most $Y\%$ of benign content is classified incorrectly). The modeling goal then becomes improving the classifier's recall within these constraints. However, producing only classification results is insufficient to prevent or mitigate abuse — eventually, the OSN must take actions with real-world consequences.

Even in a simple setting such as spam detection, where the obvious action is to simply remove the content, there are many factors to consider when taking action:

- Classifiers are never 100% precise, so users may complain that their benign content is being taken down.

- Some users may not be aware of the site's content policy; these users have the potential to be educated and thus improve the quality of their content.
- Classifiers have some latency, so malicious users may post abusive content faster than it can be taken down, leading to more spam on the OSN.

In the binary-classification approach described above, there is only one parameter we can adjust to balance these competing considerations. It follows that if we want to improve user experience without degrading our ability to remove abusive content, our only avenue is to improve the model, which is a difficult and time-consuming effort.

A first direction of improvement is to **expand the set of possible enforcement actions** beyond the binary "hard block" or "allow." For instance, we can add a third, "soft" enforcement action: an action that introduces some friction but does not completely block a user. Such actions are less disruptive to benign users while (possibly) being less effective at stopping abuse. We can now segment our classification results into three groups instead of two: the "worst" content gets blocked as before, the "best" is let through, and the "suspicious" is routed through the "soft" action. By adjusting the relative sizes of the three groups, we gain an extra degree of freedom to control the tradeoff between amount of abuse blocked and negative impact to user experience.

As a real-world example, almost every website that has a login page chooses to block some logins, let some through, and send the rest through additional checks such as a CAPTCHA or SMS code verification. CAPTCHA and SMS code verification are examples of "soft" enforcement actions that lower the cost of false positives, and thus allow the website to send "suspicious" users through some non-zero level of enforcement, which reduces false negatives. However, a multi-class classification approach would still run into the problem of defining "worst," "best," and "suspicious" in a way that is sufficiently precise to generate objective and low-noise labels.

This additional complexity compels us to consider the action-selection problem from an entirely new perspective: that of *adaptive control theory* [6] and *reinforcement learning* [68], which are frameworks that focus on decision-making with causal consequences. Reinforcement learning (RL) uses observations from previous actions to choose actions that optimize a reward (in our example, a quantity that captures the amounts of abusive content posted and benign content blocked) while also implementing a data-collection strategy that yields adaptation to non-stationary conditions. Continuous exploration is especially important in an adversarial environment, as future abusive behavior will change in response to our actions on past examples. The full technical details of our formalization appear in Section 3.

In Section 4 we present a reinforcement-learning system for fighting abuse on OSNs, which we call *Predictive Response Optimization* (PRO). We describe the system in terms that apply to *any* type of abuse, as long as it can be measured in some way. The system revolves around models that predict the cost and benefit of each action (the *contextual multi-armed bandit* setting [46]), overlaid with a *model predictive control* framework to adjust tradeoffs amongst various abuse and user-experience metrics.

In Section 5 we describe our application of PRO to the problem of reducing scraping activity in OSNs. We defined metrics, implemented the system on Instagram and Facebook, and collected data for two weeks on each. Our experiments showed that in both cases PRO stops significantly more abuse than a baseline system that determines actions by applying a set of hand-coded rules to classifier outputs. Specifically, we **reduced abuse volume by 59% on Instagram and 4.5% on fb, with no negative impact on "benign usage" metrics**.[1]

In Section 5 we also detail five case studies illustrating how PRO can quickly and automatically adapt to changes in business considerations, system behavior, and/or adversarial tactics. Specifically, our experiments show that:

- Adding a new user metric led to an 80% reduction in SMS expenditures with no increase in abuse volume.
- After we saw signs of over-enforcement on Mobile Web, adding a new user metric led to a 68% decrease in user churn.
- Introduction of a new enforcement action reduced abuse volume by 3.0% without any manual intervention.
- When a bug changed the behavior of a particular enforcement action, the system adjusted automatically to stop selecting the action.
- When adversaries began to evade a particular enforcement action, the system adjusted automatically to select that action less frequently.

In summary, our contributions are:

- We introduce the perspective that *selection of enforcement actions,* rather than binary classification, is the true goal of abuse-fighting systems in online social networks.
- We formalize action selection as a *reinforcement learning problem* that attempts to balance abuse volume against cost of blocking benign users or content.
- We design *Predictive Response Optimization* (PRO), a large-scale reinforcement-learning system for action selection and the first application of RL to abuse reduction.
- We implement the PRO system on Instagram and Facebook and observe that it significantly reduces scraping activity with no negative impact to users.
- We describe a number of case studies that demonstrate the ability of PRO to adapt to changing conditions with minimal manual intervention.

[1] The difference of an order of magnitude between the two results is due to the relative maturity of the experimental and control groups on the two OSNs. For details see Section 5.3.

2 Background and related work

Abuse on social-networking platforms can take various forms. Attackers can create fake accounts (or "sybils") [21, 27, 74, 79, 81] or compromise existing accounts, which they can then use to spread phishing links [24, 28], post fraudulent reviews and advertisements [3], disseminate fake news [23], or make fraudulent payments [15]. Such attacks motivate research on detecting and removing abusive content on social networks.

In other types of attacks, attackers collect user data from social networks to use later for malicious purposes. Previous studies demonstrate how attackers can scrape social networks to collect user information that is then used for targeted spam and phishing [7, 14, 36]. These attacks motivate research on detecting and blocking automated activity (i.e., "bots") on social networks [59].

2.1 Related work in abuse detection

Most prior works focus on how to detect abusive content and/or users by leveraging machine learning techniques that differentiate abusive from non-abusive entities. Such research often aims to improve the precision or recall of various supervised ML algorithms such as k-nearest neighbors, random forests, naive Bayes, decision trees, and neural networks [3, 11, 15, 23, 51, 67]. Given labeled data, these algorithms can use network information, behavioral patterns, or features generated from the content itself to detect adversaries [38] and even to continuously detect adversaries who try to elude the model [20]. One challenge in using supervised methods is the reliance upon labeled data, which can limit the scalability of these approaches [38]. This limitation, combined with the fact that abusive entities such as spammers also tend to act collusively, has led to the development of unsupervised methods, including graph-based and clustering methods [12, 16, 38, 47]. Whether supervised or unsupervised, these approaches focus on detecting abusive entities but ignore the negative impact of incorrect detection. Such negative impact arises only after an enforcement action is applied to the detected entity.

2.2 Enforcement methods

After identifying abusive entities, anti-abuse systems apply *enforcement actions* to induce the potentially abusive actor to stop or change their behavior. In the previously discussed abuse-detection systems, generally a single action is applied for all detected abusive entities (e.g., deleting the detected fake profile or abusive content [67]). In some cases, the system uses the detection confidence to choose between a "mild" enforcement (e.g., removing fake engagement) or a "harsh" enforcement (e.g., taking down accounts) [47].

Some OSNs, such as YouTube [82] and Facebook [53], use a "strike" system to carry out an escalating series of enforcement actions to prevent repeat offenders. On both YouTube and Facebook, the process begins with a warning. On YouTube, each strike temporarily restricts content creation, and after 3 strikes in a 90-day period the offending channel is removed [82]. On Facebook, strikes 2–6 yield temporary feature-specific restrictions, while additional strikes trigger content-creation restrictions of increasing length [53]. Currently there is no literature on abuse-minimization systems that addresses the problem of choosing between multiple enforcement methods based on a set of constraints; our work fills this gap.

2.3 CAPTCHAs, challenges, and verification

A widely used technique for combatting abuse that leverages automation (such as fake engagement or data scraping) is to present "challenges" that are difficult for bots to solve while remaining relatively accessible for humans. Von Ahn et al. [73] introduced several practical proposals for designing "CAPTCHA" schemes for this purpose: problems that most humans can solve easily but computer programs cannot. A recent survey [30] identified 10 types: text-based, image-based, audio-based, video-based, math-based, slider-based, game-based, behavior-based, sensor-based, and CAPTCHAs for liveness detection. CAPTCHAs have been used to ensure the safety of network applications [8, 50], including chat rooms, email registration, online auctions, file sharing, and polls [61].

Many systems also commonly use *multi-factor authentication* (MFA) to verify that an entity (e.g., a web session or social-media account) is being controlled by an authorized owner and not a malicious actor. Ometov et al. provided a survey of MFA methods [57]. Some common strategies include password re-verification, hardware tokens, voice biometrics, facial recognition, and phone or email verification. Both CAPTCHA and MFA actions are included as enforcement options within our system.

2.4 Optimization and control

One of our core contributions is to apply optimization and control methods to select the "best" enforcement action. Such methods have been leveraged in many application domains, but to date this list does not include OSN abuse.

Model Predictive Control (MPC) [5, 33] is a control approach that leverages a system model to predict or simulate how different inputs to the system will affect the system's output up to a certain time in the future. Based on these predictions, the operator can choose the input that leads to most desirable output. They can then repeat the control process with new observations [33, 34, 63]. MPC is robust, sample-efficient, and able to handle enforcing constraints, but it can be difficult to apply to complex systems (e.g., constructing a system model that also models uncertainty) [5].

Reinforcement learning (RL) [68] is an alternative control approach that can predict outputs of complex systems without a pre-defined model. Technically, RL is a class of algorithms that maximize specified objectives by learning from prior actions. RL algorithms include "model-free" approaches such as Q-learning, which learn the "quality" of a particular action executed while the system is in a particular state [39, 55, 75]. "Model-based" approaches, on the other hand, learn a model for the system [63, 68] and are related to "system identification" from the control field [49, 63]. RL has been used in applications such as board games (Go, chess), arcade games (PAC-MAN) [37, 65], recommender systems [19, 84], transport scheduling [10, 18, 45, 66], finance [1, 43], and autonomous driving [29, 35, 39, 42].

Prior research has combined MPC with RL to add constraints and improve safety of the RL system [5, 83]. Our work follows this approach, leveraging RL for learning the system and taking actions at an entity level and leveraging MPC to apply global constraints.

In the field of security, RL-based solutions have been proposed to defend against cyber-attacks on various IoT systems [72]. RL has also been used to improve security in cognitive radio networks [48] and to detect botnets [4]. MPC has also been applied within a security context to detect cyber-attacks in microgrids [31]. Other work focuses on detecting attacks within networked systems controlled with MPC [9]. However, to our knowledge, there is no prior work in any security context that explores the application of machine learning to optimize action selection from a list of possible enforcement actions.

3 Abuse minimization as a constrained optimization problem

In this section we reframe the goal of anti-abuse systems in online social networks: rather than existing to *classify* content or behavior, such systems exist to *optimize the tradeoff* between abuse reduction (e.g., removing spam) and impact on user experience (e.g., too much benign content removed). Specifically, our position is that **anti-abuse systems are solving the *constrained optimization problem* of minimizing abuse volume within a "budget" of operational costs.** We conjecture that this reframing will enable us to devise a system that blocks more abuse than a classification system, without adversely impacting user experience.

3.1 Abuse minimization is a tradeoff problem

Anti-abuse systems seek to achieve the dual aims of both reducing abuse prevalence and minimizing costs that arise as a result of practical considerations. Some of these quantifiable business considerations are:

- Size of the operational team that processes appeals of incorrect content deletions or account disables.
- Users choosing to leave the platform because of enforcement actions against their content.
- Negative impact to usability/engagement metrics on the OSN due to increased user friction (e.g., challenges and verifications).
- Computer and network hardware needed to run the anti-abuse system itself.

Conceptually, the abuse-cost tradeoff can be visualized as a Pareto frontier [76] if we simplify and reduce operational cost considerations to a single "cost" dimension (Figure 1). As illustrated, abuse can be fully eradicated by any method if the cost axis is not constrained — after all, if we block *all* users from using the platform, there will be no abuse left! However, for any anti-abuse method to work in a real-world setting, the system must be tuned to operate under some cost constraints ("budget" in Figure 1). The abuse/cost tradeoff space is also often multi-dimensional; i.e., multiple business-metric constraints can exist simultaneously.

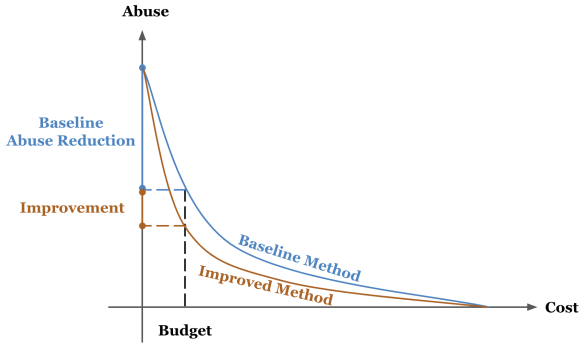


Figure 1: Two conceptual anti-abuse methods tuned under constraints.

3.2 Enforcement actions really matter

On its surface, the "what to do" problem seems fairly straightforward to solve: if an entity is abusive, remove it from the OSN! However, if we take content moderation as an example of a typical use case, we see that a policy of "delete all content classified as abusive" quickly runs into issues such as those mentioned in the Introduction.

To address these issues while remaining effective at stopping abuse, anti-abuse systems typically employ multiple enforcement actions that are designed to hinder or discourage abusive behavior to varying degrees. At the same time, these actions affect benign (i.e., non-violating) users to different extents, ranging from mild annoyance to total loss of access to their account and the value associated with it. As

an example, in the case of content moderation, a possible set of enforcement actions could include one or more of the following (ranging from least to most intrusive):

- No action,
- Incrementing a "strike counter" for the author [82],
- Showing a warning notification to the author,
- Down-ranking/reducing visibility of the content,
- Asking the author to perform an interactive challenge (e.g., CAPTCHA, MFA) before posting content [64, 73],
- Deleting the content,
- Temporarily banning the author from sharing content,
- Temporarily banning the author from the service,
- Permanently revoking the author's access to the service.

In "traditional" systems, action selection is implemented using a set of hand-written logical rules that incorporate information such as classifier confidence, past enforcement/strike counts, and other features related to the entity. The focus of prior research on the classification problem, while leaving enforcement selection to rule- or strike-based logic, results in three major pitfalls when applied in practical settings:

- Abuse prevalence could remain high due to ineffective enforcement; e.g., as a result of poor optimization against practical constraints.
- Fixed action-selection rule logic does not adapt to changing adversarial behavior; e.g., learning to bypass certain enforcement actions.
- Introducing new actions requires writing new routing rules and could cause constraint metrics to shift in the wrong direction.

Instead of relying on hand-written rules, we propose formalizing the enforcement-selection problem as a constrained optimization problem and then develop a reinforcement-learning algorithm to solve it.

3.3 Formalizing the optimization problem

Anti-abuse systems evaluate and execute actions over a potentially very large set $\mathcal{E}$ of entities (OSN accounts, pieces of content, IP addresses, and so on). If the system takes action on entity $e \in \mathcal{E}$ at time t_0, we can measure the impact of this action on entity e over the subsequent time period $\tau = [t_0, t_1]$ using a set of "abuse metrics" $Abuse_j$ $(j = 1, \ldots, N_A)$ and "cost metrics" $Cost_j$ $(j = 1, \ldots, N_C)$.

To make our discussion more concrete, we will use the running example of OSN accounts posting spam, with abuse metric $Abuse^*$ being "number of spam posts during the time period τ" and cost metric $Cost^*$ being "number of non-spam posts blocked during the time period τ."

Let $\mathcal{A}$ denote the (finite) set of all possible actions. We define the vector of all "next actions" for all entities as

$$\mathbf{a} \in \mathcal{A}^{|\mathcal{E}|} = \left\{ (a_e)_{e \in \mathcal{E}} : a_e \in \mathcal{A} \right\}. \tag{1}$$

The set $\mathcal{A}$ contains the special action A_0 denoting "no action," plus other actions such as those described in Section 3.2. In our spam example we will let $\mathcal{A}$ be a set of three actions: A_0 = "no action"; A_1 = "show a CAPTCHA"; A_2 = "disable the account."

At optimization time t_0, our goal is to determine the action vector $\mathbf{a}$ that minimizes the total abuse over the period τ

$$\min_{\mathbf{a}} \sum_{e \in \mathcal{E}} \sum_{j=1}^{N_A} Abuse_j(e \mid \mathbf{a}) \tag{2}$$

while constraining the cost over the same period

$$\sum_{e \in \mathcal{E}} Cost_j(e \mid \mathbf{a}) \leq B_j, \qquad j = 1, \ldots, N_C \tag{3}$$

where $B_j \geq 0$ is a global "budget" for the jth metric. In our concrete example, we want to minimize total spam posts while keeping the total number of blocked non-spam posts below a certain fixed number B^* determined by the business.

We assume that all abuse and cost metrics are normalized in such a way that

$$Abuse_j(e \mid \mathbf{a}^0) = Cost_j(e \mid \mathbf{a}^0) = 0, \tag{4}$$

where the baseline action vector $\mathbf{a}^0 = (A_0, \ldots, A_0)$ corresponds to applying "no action" to all entities, effectively "turning off" the anti-abuse system. Furthermore, we assume that the metrics are signed such that smaller values are "better" (i.e., we want to *minimize* both abuse and cost). This normalization ensures that $\mathbf{a}^0$ always satisfies the constraints, making the optimization problem feasible (i.e., a solution always exists).

Let us consider the effect of this normalization on our spam example. An account e_{spam} that posts only spam will (presumably) have $Abuse^*(e_{spam} \mid \mathbf{a}) \leq 0$ for all $\mathbf{a}$, since (presumably) any nontrivial action will reduce the amount of spam posted by that account. The account will also have $Cost^*(e_{spam} \mid \mathbf{a}) = 0$ for all $\mathbf{a}$ since there are no non-spam posts to block. On the other hand, an account e_{benign} that posts no spam will have $Abuse^*(e_{benign} \mid \mathbf{a}) = 0$ for all $\mathbf{a}$ and $Cost^*(e_{benign} \mid \mathbf{a}) \geq 0$ for all $\mathbf{a}$ since (presumably) any nontrivial action will only decrease the number of non-spam posts; i.e., contribute a non-negative number of blocked posts.[2]

In practice, at optimization time we don't have access to any of the abuse and cost metrics since they refer to a future time period and will only be available after a time delay. Moreover, the constrained optimization problem (2)–(3) couples together all the entities, which makes it unfeasible to solve at high frequency. Therefore, we also consider an unconstrained

[2] Note that "blocked posts" in this example includes not only non-spam posts blocked directly but also those "prevented" relative to the no-action baseline. For example, if the action is to disable the account then the $Cost^*$ metric attempts to estimate how many non-spam posts the user "would have made" had they not been blocked.

relaxation consisting of maximizing the following "reward" function with respect to $\mathbf{a}$:

$$r(\mathbf{x},\mathbf{a}) = -\sum_{e\in\mathcal{E}} r_e(\mathbf{x},\mathbf{a}), \quad r_e(\mathbf{x},\mathbf{a}) = \sum_{j=1}^{N} w_j \cdot m_j(e \mid \mathbf{x},\mathbf{a}) \quad (5)$$

where we have combined together all the metric functions

$$(m_1,\ldots,m_N) = (Abuse_1,\ldots,Abuse_{N_A},Cost_1,\ldots,Cost_{N_C}),$$

(setting $N = N_A + N_C$) and introduced multipliers $w_j \geq 0$ that determine the relative weighting of each *Abuse* and *Cost* metric. The w_j also implicitly convert all metrics into a common unit; for example an abuse metric might be in units of spam posts while a cost metric might be in units of benign users blocked. The quantity

$$\mathbf{x} = (x_e \in \mathcal{X})_{e\in\mathcal{E}} \quad (6)$$

represents the state information (features) available at optimization time for all entities, where $\mathcal{X}$ is the "feature space" used to describe the state for a particular entity. This information allows us to leverage predictive models obtained via machine learning to approximate a solution to the optimization problem. Applying the notation in (5) to our spam example gives us the per-entity reward function

$$r_e(\mathbf{x},\mathbf{a}) = Abuse^*(e \mid \mathbf{x},\mathbf{a}) + w \cdot Cost^*(e \mid \mathbf{x},\mathbf{a}). \quad (7)$$

Here we can interpret w as the "relative weight" of the two harms being traded off: blocking one non-spam post is equivalent to allowing w spam posts.

In general, maximizing (5) is not equivalent to solving (2)–(3), though in some cases there exist Lagrange multipliers w_j that make the two problems exactly equivalent. Nevertheless, the multipliers w_j can be adjusted periodically to ensure that the optimal solution of (5) tracks the optimal solution of (2)–(3) as closely as possible. We show in the next section that the unconstrained relaxation (5), combined with suitable modeling assumptions, enables high-frequency decision making by decoupling the optimization across entities.

4 Solving the optimization problem

In this section we describe a strategy to approximate solutions to the optimization problem (2)-(3). Our strategy combines elements of reinforcement learning (RL) with model predictive control (MPC) and consists of two components:

1. Optimizing actions $\mathbf{a}$ at entity level,
2. Optimizing multipliers w to enforce global constraints.

Figure 2 provides an overview of the system components and the design choices for each component.

We will show that entity-level action optimization can be described as a *contextual multi-armed bandit problem*. By introducing suitable modeling assumptions, we are able to decide actions asynchronously for each entity, at arbitrary time intervals. Similarly, our approach to finding optimal multipliers can be seen as a form of stochastic *model predictive control* (MPC) aimed at enforcing global constraints. Other works combining RL with MPC include [5, 83].

4.1 Optimizing action selection

In reinforcement learning (RL) terminology, the optimization system (a.k.a. *agent*) acts within an *environment*. The agent takes an *action* chosen from a set of possible actions depending on the agent's *state* and then receives an application-specific *reward*. The choice of action is based on the agent's *policy*, which in addition to selecting actions to maximize the expected reward ("exploit"), also strives to gain information ("explore") to improve the policy itself [68].

Maximization of (5) can be readily framed as an RL problem by defining the state $\mathbf{x}$ as in (6) and the set of actions $\mathcal{A}$ as in (1). Equation (5) defines the reward as a weighted sum of the values contributed by each entity towards each metric (in our example, the amount of spam content and number of erroneous deletions). Each of these metrics is a cumulative quantity aggregating data with timestamps between the time t_0 when an action is chosen and a future time t_1. We call the interval $\tau = [t_0, t_1]$ the *time horizon*. The environment is modeled via the state and the metric functions.

After applying an action, an entity's behavior may change, altering the received reward (i.e., metric values) over the time horizon τ. In our example, if we temporarily ban an account e from sharing content, then $Abuse_1(e)$ counting number of spam posts may decrease but $Cost_1(e)$ counting number of non-spam posts blocked may increase (relative to their values if no action were taken).

In more general RL problems, the reward is a function of transitioning from state $\mathbf{x}$ to state $\mathbf{x}'$ via action $\mathbf{a}$: $r(\mathbf{x},\mathbf{a},\mathbf{x}')$ [68]; however in Equation (5) we simply have $r(\mathbf{x},\mathbf{a},\mathbf{x}') = r(\mathbf{x},\mathbf{a})$ (i.e., the *contextual multi-armed bandit* setting [46]). In other words, instead of modeling state transitions from multiple agent actions as a Markov decision process [62], we aim to predict the incremental reward from each individual action, thus simplifying the RL problem.

The goal of the RL problem is to maximize the cumulative reward over multiple action-selection events (also known as the *return*). Without loss of generality, action selection can be viewed as sampling $\mathbf{a}$ from a probability distribution $\Pi(A \mid \mathbf{x})$ conditioned on the state $\mathbf{x}$; here A is a random variable taking values in $\mathcal{A}^{|\mathcal{E}|}$ and we call the distribution Π the *policy* [68]. (Note that this framing includes the case when actions are chosen deterministically.)

Our approach to maximizing cumulative reward is to first build a predictive model $\pi(R \mid \mathbf{x},\mathbf{a})$ describing the probability distribution for the next reward value (5) modeled as a random variable R conditioned on a given pair of states and actions

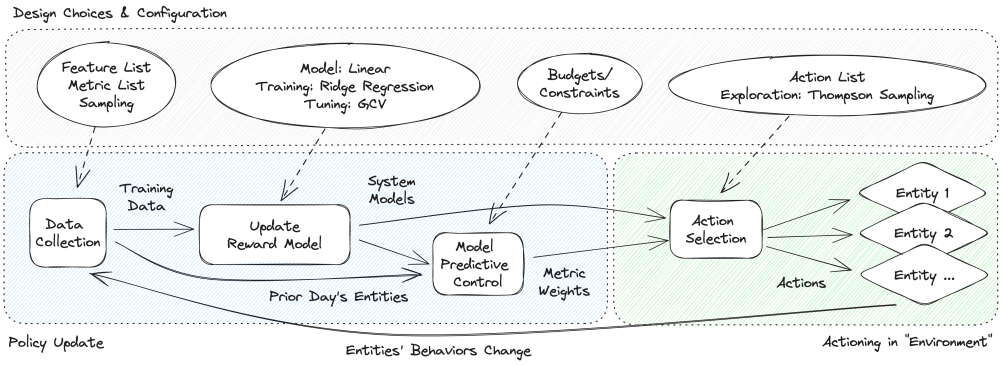


Figure 2: Predictive Response Optimization system design

$(\mathbf{x},\mathbf{a}) \in \mathcal{X} \times \mathcal{A}^{|\mathcal{E}|}$. We then implicitly define our policy Π by its sampling mechanism:

$$\mathbf{a} \leftarrow \Pi(A \mid \mathbf{x}) := \underset{\mathbf{z} \in \mathcal{A}^{|\mathcal{E}|}}{\operatorname{argmax}} \left(r \leftarrow \pi(R \mid \mathbf{x}, \mathbf{z}) \right) \tag{8}$$

In our example of spam posts, we realize the policy in (8) by building a predictive model for the reward function (7), predicting the reward for each component of the action vector $\mathbf{z} \in \mathcal{A}^{|\mathcal{E}|}$, and setting $\mathbf{a}$ to be the action vector $\mathbf{z}$ with the greatest reward.

Our sampling approach is derived from *Thompson sampling* [2,70]; in particular, the variability inherent in the model π allows us to balance exploration and exploitation (a classic challenge in reinforcement learning) by tuning model parameters [68]. Exploration is necessary to improve the model's understanding of how different actions impact each metric in various regions of the feature space. It is particularly important in adversarial environments, as malicious actors may learn to work around certain enforcement actions, rendering a previously heavily exploited enforcement action ineffective (see Section 5.6 for an example). Exploration is also useful as a mechanism to onboard newly developed enforcement actions to the system (see Section 5.5 for an example).

4.2 Reward models

In general, taking action on one entity may affect other entities' metrics. For instance, when banning a user account for spamming, other users will not be exposed to the spam anymore. However, modeling all possible interactions is prohibitively expensive. Therefore, in the model described in this section we neglect the impact of actions taken on a given entity to metrics for other entities. This decoupling allows us to determine actions in real time whenever a decision for a particular entity is needed. Formally, we set

$$\pi(R \mid \mathbf{x}, \mathbf{a}) = \prod_{e \in \mathcal{E}} \hat{\pi}(R_e \mid x_e, a_e), \tag{9}$$

where R_e is a random variable modeling the portion of the reward contributed by entity e, the contextual features x_e summarize all information about entity e that is deemed useful to predict future metric values, and $\hat{\pi}$ is a (global) model that predicts rewards at the entity level given the contextual features x_e and the action a_e applied to e. Examples of features relevant for abuse minimization at user-account level include:

- Account properties such as age, classifier scores [80], number of sessions, or frequency of requests;
- Time series of metric values for the user;
- History of prior actions taken on the account (e.g., a vector of whether each action was taken the day before, 2 days before, and so on);
- Classifier scores for the account's content (e.g., probability of spam);
- IP statistics such as number of requests in a time window or number of accounts using the IP;
- Request features such as requested URL or browser type.

Let $\mathcal{T}$ be the *training window*; i.e., a set of timestamps in the past for which we have entity-level historical data. For each $t \in \mathcal{T}$, let x_e^t be the value of the feature vector (i.e., state) x_e at time t, let a_e^t be the action taken on entity e at time t, and let m_j^t be the value of the metric m_j starting at time t (i.e., over the period $[t, t+t_1-t_0]$). For each metric m_j and action $A_k \in \mathcal{A}$, we construct datasets

$$\mathcal{D}_{jk} = \left\{ \left(x_e^t, m_j^t(e \mid x_e^t, a_e^t) \right) \quad \forall e \in \mathcal{E},\ t \in \mathcal{T} : a_e^t = A_k \right\}$$

that document historical (state, metric-value) pairs for each action and metric. In our spam example, $\mathcal{D}_{1k}$ is the dataset consisting of all (feature-vector, *Abuse**-value) pairs for the entities e that received action A_k at some time during the training window, while $\mathcal{D}_{2k}$ is the corresponding set of (feature-vector, *Cost**-value) pairs.

In our implementation we sample $t \leftarrow \mathcal{T}$ in such a way that the amount of data decreases exponentially as the timestamp $t \in \mathcal{T}$ goes back in time, biasing the dataset towards recent information while still keeping a fraction of older data. When first deploying the system (the "cold-start problem"), these

datasets can be initialized with data collected from a baseline rule-based system. Subsequently, they consist of data collected after applying actions chosen by the RL system.

After constructing the datasets, we model the metric values m_j^t as noisy Gaussian samples

$$m_j^t(e \mid x_e^t, a_e^t) \leftarrow \mathcal{N}(\nu_j(x_e^t, a_e^t), \varepsilon)$$

where ν_j ($j = 1, \ldots, N$) are realizations of independent Gaussian Processes (see [78]) with mean zero and kernel (covariance function)

$$K_j((x_1, a_1), (x_2, a_2)) = (\phi_j(x_1)^T \phi_j(x_2)) \cdot \delta(a_1, a_2). \quad (10)$$

Here the maps $\phi_j : \mathcal{X} \to \mathbb{R}^D$ represent "feature transformations": functions that process the raw input features, extract interactions, and output D-dimensional numerical vectors. For example, ϕ_j might include taking log of account age to reduce skew, applying one-hot encoding to categorical features [32], or combining action history data to create a summary feature "number of times blocked in the last 14 days." The function $\delta(x, y)$ is 1 if $x = y$ and 0 otherwise.

Given the modeling assumptions above, we can use Gaussian Process Regression [77] to obtain predictive distributions ν_j for all metrics m_j. In view of our assumption that the metrics appearing in Equation (5) are independent, we obtain reward models of the form

$$\hat{\pi}(R_e \mid x_e, a_e) = \mathcal{N}\left(\sum_{j=1}^{N} w_j \cdot \mu_j(x_e, a_e), \sum_{j=1}^{N} w_j^2 \cdot \sigma_j^2(x_e, a_e)\right). \quad (11)$$

Moreover, in view of the kernel structure, we have

$$\mu_j(x_e, a_e) = \mu_{jk}(x_e), \quad \sigma_j^2(x_e, a_e) = \sigma_{jk}^2(x_e) \quad (12)$$

where k is the index in $0, \ldots, |\mathcal{A}| - 1$ such that $a_e = A_k \in \mathcal{A}$, and the functions μ_{jk} and σ_{jk}^2 are defined as

$$\mu_{jk}(x) = \phi_j(x)^T \theta_{jk}, \quad \sigma_{jk}^2(x) = \varepsilon \cdot \left(\phi_j(x)^T \Sigma_{jk} \phi_j(x)\right). \quad (13)$$

The parameters θ_{jk} and Σ_{jk} can be learned by pooling historical data across all entities to train $N \cdot |\mathcal{A}|$ independent models (one for each metric m_j and action type k). Specifically, we can compute

$$\Sigma_{jk} = \left(X_{jk}^T X_{jk} + \lambda_{jk} \cdot I_D\right)^{-1} \quad \in \mathbb{R}^{D \times D}, \quad (14)$$

$$\theta_{jk} = \Sigma_{jk} \cdot X_{jk}^T \cdot Y_{jk} \quad \in \mathbb{R}^D, \quad (15)$$

where X_{jk} is a $\left|\mathcal{D}_{jk}\right| \times D$ matrix with rows $\phi_j(x_e^t)$ for all (e,t) in the dataset $\mathcal{D}_{jk}$, Y_{jk} is a $\left|\mathcal{D}_{jk}\right|$-dimensional column vector containing the corresponding metric values m_j^t, and I_D is the $D \times D$ identity matrix. We set Ridge regularization parameters λ_{jk} by optimizing generalized cross-validation scores [26], while the noise variance ε is a global parameter that can be used to control the rate of exploration. (In our experiments we set $\varepsilon = 0.05$.) In our spam example with two metrics and three actions, this process gives us 6 predictive models, each described by $D^2 + D$ parameters Σ_{jk}, θ_{jk}.

Computational complexity. For the training step, since the parameter computations (14)–(15) are linear, they can still be feasible for datasets $\mathcal{D}_{jk}$ with millions of entries and feature dimensionality D in the hundreds. In particular, the limiting step is the matrix multiplication and inversion of (14), which is $O\left(D^2(D + \left|\mathcal{D}_{jk}\right|)\right)$ in our implementation.

For the inference step, the simplicity of the models described in (11)–(13) (where the limiting step is $O(D^2)$) enables us to process a very large volume of requests at inference time, interpret model weights, and understand the directional impact of features on predictions and decisions. However, we find in practice that we must frequently update the model because the highly non-stationary nature of both the adversarial and business landscapes results in constant shifts in both cost budgets and system effectiveness. (See Sections 5.4 and 5.6 for examples.)

4.3 Enforcing business constraints

The multipliers w_j (Equation (5)) control the tradeoffs amongst the various abuse and cost metrics. While reinforcement-learning approaches often leverage a single model that predicts a "quality estimate" of the state resulting from a particular action applied to a given state [55], by instead decomposing the reward into a weighted combination of individual metric models we can adjust these metric tradeoff weights "on the fly" without model retraining. This property allows us to make quick adjustments if we observe the system being "too harsh" (i.e., cost constraints are violated) or "too lenient" (i.e., we are not using our entire cost budget). We adjust the multipliers automatically using a controller designed to maximize the estimated reward aggregated over all entities and over a future time period, while attempting to satisfy the budget constraints for the system (Section 3.1). Our approach can be interpreted as an instance of stochastic Model Predictive Control (MPC) [52] with multiple control variables.

In MPC, a model of the system or "plant" to be controlled (in our case the RL "environment") may be specified via state space or a transfer function, or learned via means such as system identification. Then, using this plant model, MPC will predict the plant's outputs for various controller outputs. These predictions can be at multiple time horizons, e.g., +1 second, +1 day, etc. MPC will select the controller outputs that are predicted to yield values closest to the plant's desired output. Additional constraints can also be applied, e.g., limiting the plant outputs to a certain range.

The MPC framework is well suited to our goal of minimizing abuse while enforcing the business constraints for our system. As described previously, we continually learn and update our reward models. Using these models, we can leverage the prior day's data to predict the overall action and metric

distributions for a variety of metric tradeoff weights, i.e., the "simulation" in MPC. This process is a form of multi-variable MPC where the control horizon is one step ahead. Then, if we set the metric tradeoff weights w_j to be the parameters controlled by MPC, the controller can pick these parameters such that the constraints are met (in expectation). Note that using the prior day's data for simulation assumes the distribution of features and metrics does not shift substantially from one day to the next.

Consider again the optimization problem described in Equations (2)–(3), now assuming that the action vector $\mathbf{a}$ is obtained by sampling from the policy Π defined in (8). Since Π depends on the multipliers w_j via (9) and (11), we can now consider optimizing the objective (in expectation) with respect to w:

$$\min_{w} \sum_{e\in\mathcal{E}} \sum_{j=1}^{N_A} \mathbb{E}\left[w_j \cdot m_j(e \mid \mathbf{a})\right], \quad \text{subject to}$$
$$\sum_{e\in\mathcal{E}} \mathbb{E}\left[w_j \cdot m_j(e \mid \mathbf{a})\right] \le Budget_j, \quad j = N_A+1,\ldots,N.$$

We are now left with optimizing N weights instead of $\#\mathcal{E}$ actions, which is a massive reduction in dimensionality. However, evaluating the objective and constraints by summing over all the entities is still very costly. To further reduce the computational cost, we can use a smaller set $\mathcal{S}$ of entities sampled uniformly at random from all entities processed by the optimization system in the previous period (e.g., the previous day) and use the mean reward models learned previously to estimate the expectations, leading to the optimization problem

$$\min_{w} \sum_{e\in\mathcal{S}} \sum_{j=1}^{N_A} w_j \cdot \mu_j(x_e,a_e), \quad \text{subject to} \tag{16}$$
$$\sum_{e\in\mathcal{S}} w_j \cdot \mu_j(x_e,a_e) \le b_j, \quad j = N_A+1,\ldots,N, \tag{17}$$

where $b_j = s \cdot Budget_j$ is a rescaled budged where the scaling factor s can be used to account for the relative size of the sample set $\mathcal{S}$ compared to the entire set $\mathcal{E}$, or to incorporate forecasted metric increases from one period to the next (e.g. due to a planned product change). To solve this optimization problem we use a grid search centered around the current metric tradeoff weights w to generate candidate weight sets. We then set the new weights to be the candidate weight set that minimizes the abuse metrics while remaining within the budget constraints set on the cost metrics.

5 The PRO system in practice

The description of Predictive Response Optimization in Section 4 is generic; i.e., it can be applied to any OSN abuse problem using any set of abuse and cost metrics. In this section we turn our theory into reality, showing how to adapt the system to detect and block bots scraping an OSN.

We worked with product and engineering teams to implement the PRO system on Instagram and Facebook, both of which have more than one billion monthly active users. Instagram is a "directed" social network where people follow creators and engage with their content, while Facebook is an "undirected" social network where users connect and engage with people they know in real life.

On each OSN we implemented the system, collected data, trained a PRO model, and conducted online controlled experiments [41] to compare PRO's performance with that of a "rule-based" baseline. Due to differences in the ways users interact with the platforms as well as the state of each OSN's infrastructure, the exact baseline rules are specific to each OSN. Below we describe our experimental setup, measure how much more abuse PRO can stop relative to our baseline system, and document observations about how PRO adapts to changing business, system, and adversarial conditions.

5.1 Implementation Details

Metrics. In order to implement PRO we first need to define the "Abuse" and "Cost" metrics introduced in Section 3. We selected the following metrics for our experiment:

- *weighted scraping requests*: Count of logged HTTP requests identified as scraping using a scaled labeling system, with each request weighted by the number of units of user-identifiable information returned to the user. The labeling system consists of a set of rules generated by security analysts and expanded by automation. For example, one rule to detect a particular scraping attack is `user-agent="python-requests/<version>" and endpoint="<endpoint_name>"`. We use this metric to quantify abuse.
- *days active*: Count of calendar days during the measurement period during which an account is observed to be active. This metric correlates with user engagement and is used to quantify cost: if PRO is over-enforcing then *days active* will decrease. (Due to our normalization and sign conventions (4), the actual cost metric in our model will be "loss of days active" relative to the no-action baseline; i.e., $days\ active(a^0) - days\ active(a)$.)
- *feedback events*: Count of calendar days during the measurement period during which the account files a bug report. This metric correlates with incorrect actions and is used to quantify cost: if PRO takes too many actions on benign users, some users will perceive the enforcement to be a bug and *feedback events* will increase.

Note that each of these metrics can be calculated on a per-user basis to provide training data for the RL models. They can also be aggregated across multiple users for use in the MPC controller.

Actions. At the time of our experiments, the following actions were available on both OSNs:

- Temporarily disable the account.
- Send the account through a compromise recovery flow.
- Invalidate all sessions, forcing the account to re-authenticate.
- Invalidate all sessions, plus limit the account to a single active session (i.e., each new login invalidates the existing session).
- Invalidate only the suspected automated session.
- No action.

In addition, the following actions were also available on Instagram:

- Show a warning dialog that the user has to acknowledge before they can make further requests.
- "Challenge" the account by sending a One-Time Passcode (OTP) via SMS to the account's phone number.
- Show a CAPTCHA.

Each of these actions (other than "disable" and "no action") forces the user to perform some form of authentication to regain use of their account. Our hypothesis is that different actions will have different levels of effectiveness on abusive accounts (some may go away while others may pass the challenges and continue scraping) and different impacts for benign users (some may pass the challenges and continue as before, while others may get frustrated and stop using the OSN entirely). The PRO system's goal is to optimize response selection based on the features of the account in question.

Model training. When the system starts running, the RL models have no data. However, because we have a baseline rule-based system, we can initialize the RL model training dataset using features and metric values after the rule-based system takes action on the entities. Once the RL system starts taking actions, it logs training data based on its own actions and can be retrained daily.

We started training daily models 2–4 weeks before the experiments so that the RL system was in a steady state by the time the experimental results were collected. On Instagram, our training data sets consisted of 8 million rows (accounts) and 201 columns (features). On Facebook, our training data sets consisted of 8 million rows and 15 columns. On average, model training took 3.4 hours per metric on a 26-core x86 CPU with 64 GB RAM.

To assess accuracy of the model predictions, we measure the RL models' mean squared error (MSE) against ground-truth data. To normalize the MSE (i.e., squared ℓ_2-norm of residuals) we divide it by the squared ℓ_2-norm of the ground-truth values. On Instagram the normalized errors for *weighted scraping requests*, *days active*, and *feedback events* are 0.51, 0.24, and 1.13, respectively, while on Facebook the respective normalized errors are 0.51, 0.33, and 0.99. These results show that our models for *weighted scraping requests* and *days active* are good predictors. The relatively high error on *feedback events* is due to the high sparsity of the data: we see in Tables 1 and 2 that fewer than three users out of every thousand file a bug report.

Algorithm 1 Response selection logic for Instagram

```
if max({automation classifier scores}) ≥ s_1 then
    disable account
else no action
end if
```

5.2 Experimental setup

Experiment population. Both Instagram and Facebook were running a number of automation-detection classifiers $C_1, \ldots, C_K$ prior to the deployment of PRO, as well as classifiers for producing a general account-level "abuse score" [80], which is used in the rule-based decision logic for Facebook. Each classifier outputs a real-number score $s_i \in [0,1]$, and for each classifier we computed the threshold t_i giving the classifier 90% precision according to human-labeled ground-truth data. On each OSN we then took a random sample of accounts for which *any* classifier score s_i was greater than t_i and assigned each of these accounts with probability 0.5 to either a Control group or a Test group. Accounts in the Control group received an action determined by a rule-based system (described below), while accounts in the Test group received an action determined by PRO.

For Instagram, the experiment ran from Sep 25 to Oct 8, 2023 (14 days), and the metrics from accounts assigned to each group were cumulatively aggregated over the entire experiment period and compared. 546,289 unique accounts were selected for the Control group and 545,949 unique accounts were selected for the Test group.

For Facebook, the experiment ran from Aug 7 to Oct 3, 2023 (58 days), and the metrics were cumulatively aggregated over the final 14 days (Sep 20 to Oct 3, 2023) and compared. 495,083 unique accounts were selected for the Control group, and 494,724 unique accounts were selected for the Test group.

At the time of the experiments, both OSNs used manually designed, rule-based action-selection algorithms. Rule-based algorithms are state-of-the-art, used by various OSNs (Section 2.2), and a multi-class classification baseline is not feasible due to the inability to obtain ground truth for which actions are optimal (Section 1). Algorithm 1 describes a representative example of the rules on Instagram for this abuse scenario, while Algorithm 2 does the same for Facebook. The algorithms incorporate the outputs of the automation classifiers $C_1, \ldots, C_K$ described above; in particular, we assume that these classifiers all output scores in $[0,1]$, with scores closer to 1 indicating higher likelihood of automation. Algorithm 2 also assumes we have an "account abuse" classifier such as that described in [80].

Algorithm 2 Response selection logic for Facebook

```
if max({automation classifier scores}) ≥ s1 then
    if account abuse score ≥ s2 or
      last compromise recovery was ≤ N1 days ago or
      account was registered ≤ N2 days ago then
        disable account
    else send account through compromise recovery flow
    end if
else no action
end if
```

We note that while the automation classifiers used in the experiment are retrained throughout the experiment time periods, these classifiers are shared between control and test groups and thus affect both the baseline and PRO equally, enabling us to isolate the difference in performance between rule-based and PRO action selection in the experiment.

5.3 Experimental results

Experimental results for Instagram and Facebook appear in Table 1 and Table 2, respectively. Metric values are summed across the last 14 days of each experiment and averaged per account. We determined statistical significance using a two-sample t-test. Bold p-values indicate statistically significant results ($p \leq 0.05$).

The experimental results show that our method can significantly reduce overall *Abuse* metrics (see Equation (2)). In particular, on Instagram, **PRO reduced *weighted scraping requests* by 59%** while causing no degradation in the two *Cost* metrics, while on Facebook, **PRO reduced *weighted scraping requests* by 4.5%** with no statistically significant degradation in *Cost* metrics.

The difference of an order of magnitude between the two experimental outcomes is a result of the system's development timeline. When we first began developing PRO on Instagram, the heuristic rules on that OSN were fairly rudimentary, as evidenced by Algorithm 1. Rather than improve the rules, we focused our efforts on implementing and optimizing PRO and were able to realize the observed massive reduction in scraping relative to the prior state. During this period, the rules on Facebook increased in sophistication, and when we turned our attention to the second OSN the rules were in the state exemplified by Algorithm 2. Furthermore, the PRO system implemented for Facebook is a simple port of the system developed and optimized for Instagram; we expect that we could realize further gains with commensurate optimization. Thus the large difference in impact between the two OSNs results from (a) a more sophisticated baseline algorithm on Facebook and (b) less effort invested in optimizing PRO for Facebook.[3]

[3]The fact that error metrics for model predictions are comparable on the two OSNs suggests that the difference does *not* result from the Instagram models using more features than the Facebook models.

Metric	Control	Test	Delta	p-value
weighted scraping requests	16,700	**6,830**	**−59.2%**	**0.00**
days active	2.94	**2.98**	**+1.4%**	$\mathbf{6.3 \times 10^{-36}}$
feedback events	2.50×10^{-3}	2.80×10^{-3}	+12.0%	0.089

Table 1: Experimental results for Instagram. Reported numbers are per-account averages over a 14-day period.

Metric	Control	Test	Delta	p-value
weighted scraping requests	3,540	**3,380**	**−4.51%**	**0.0395**
days active	3.26	3.25	−0.430%	0.0777
feedback events	1.20×10^{-3}	9.61×10^{-4}	−20.2%	0.163

Table 2: Experimental results for Facebook. Reported numbers are per-account averages over a 14-day period.

5.4 Incorporating new business considerations

Another key feature of PRO is the ease of re-optimizing the system to incorporate new business considerations (cf. examples in Section 3.1). Here we share two case studies of introducing such considerations to PRO on Instagram.

Case Study 1: Reducing over-enforcement. After observing signs of over-enforcement[4] on the Instagram Mobile Web product, we formulated a new constraint aimed at limiting the reduction in user activity on Mobile Web, in order to serve as a "guardrail" against over-enforcement. To implement this constraint we added to the PRO system a new cost metric quantifying *days active on Mobile Web*.

Case Study 2: Reducing SMS expenditure. Short message service (SMS) code verification is one of our available actions on Instagram. Its goal is to verify user identity and/or prevent unauthorized access and abusive traffic coming from hacked accounts. Sending these codes has an associated financial cost. In order to reduce SMS expenditures, we added a new cost metric to PRO measuring *dollars spent on SMS messaging*.

In both of the above cases, we adjusted the PRO system according to the following steps, and in both cases compared the effects of the two reward functions:

1. Log account-level attribution of the new cost metric.
   (a) For *Case Study 1*, log whether the account is active on Mobile Web each day.
   (b) For *Case Study 2*, log the total expenditure due to SMS messages sent to the account each day.

[4]Since PRO is an optimization model rather than a classification model, the concepts of "false positive" and "false negative" do not technically apply to it. However, PRO can make locally sub-optimal decisions (as determined by information obtained later). We call actions that are sub-optimal in the cost dimension "over-enforcement" (corresponding to false positives) and actions that are sub-optimal in the abuse dimension "under-enforcement" (corresponding to false negatives).

2. Join the cost-attribution logs with enforcement-action logs and account features to generate training data for new metric-prediction models (Section 4.2).
3. Train the new metric-prediction models.
4. Solve (16) to determine the metric weights in the reward function of Equation (5), with the new constraint $Budget_k$ as one of the algorithm inputs.
5. Update the system's reward function with the new metric weights and new metric prediction models.

In *Case Study 1*, we determined that the product impact of over-enforcement was significant enough to warrant an immediate model adjustment rather than an online controlled experiment; we therefore used a "before and after" approach to quantify the impact. We collected data on the previous reward function for the 7 days from Jun 20 to 26, 2023, launched the new reward function on Jun 27, and collected data from Jun 29 to Jul 5. We found that the new reward function increased *days active on Mobile Web* by **68%** ($p = 0.02$) and decreased *weighted scraping requests* by **12%** ($p = 0.006$), showing that re-weighting the reward function can both reduce cost and increase effectiveness.

In *Case Study 2*, we ran an online controlled experiment, using the new reward function in the Test group and the previous reward function in the Control group. We collected data from Sep 17 to 23, 2023 (7 days) comparing 1,277,330 accounts in Control with 1,277,823 accounts in Test.[5] The data show that we reduced *dollars spent on SMS messaging* by **80%** ($p \ll 0.001$) without any significant impact on *weighted scraping requests* ($2.0 \pm 2.6\%$ reduction; $p = 0.13$). Qualitatively, we observed that PRO switched to other available enforcement actions of similar effectiveness whenever possible, reserving SMS code verification for entities where it would be most effective at stopping abuse.

5.5 Onboarding new enforcement actions

The PRO system simplifies the process of introducing and testing new enforcement actions. In the absence of an ML-based system to select enforcement actions, action selection relies on domain expertise to create hard-coded rules that decide when to apply the new enforcement action. RL, on the other hand, addresses this "cold-start problem" via exploration.

Case Study 3: A new enforcement action. We implemented a new enforcement action on Facebook and added it to the "library" of PRO actions. The action invalidates existing web sessions created by the account, forcing the account to re-authenticate. In addition, it restricts the account from creating multiple concurrent sessions, allowing only one device to be logged in to Facebook at any given time. Our hypothesis was that the new action would be more effective against accounts that use multiple concurrent sessions to perform automation, while having smaller impact on incorrectly classified users than the account-disable action, since one session is still allowed. We ran experiments with this action using the following metrics:

- *automated requests* (abuse metric): Number of HTTP requests identified to be resulting from scraping (i.e., unweighted version of *weighted scraping requests*).
- *time spent* (cost metric): Time duration (in seconds) that the account spends active on the OSN (i.e., continuous version of *days active*).

In Figure 3, we see that initially PRO does not have any knowledge about the potential impact of the new enforcement action. In this stage we observe large fluctuations in selection rate (3a), accompanied by overall sub-optimal action selection with higher numbers of *automated requests* (3b) and lower user-engagement metrics (3c). Around day 31, the system stabilizes with smaller shifts in action-selection rate, more optimal action selection (i.e., reduction in abuse) in the test group, and higher user-engagement metrics. At this point, the system is starting to utilize the new enforcement action more effectively.

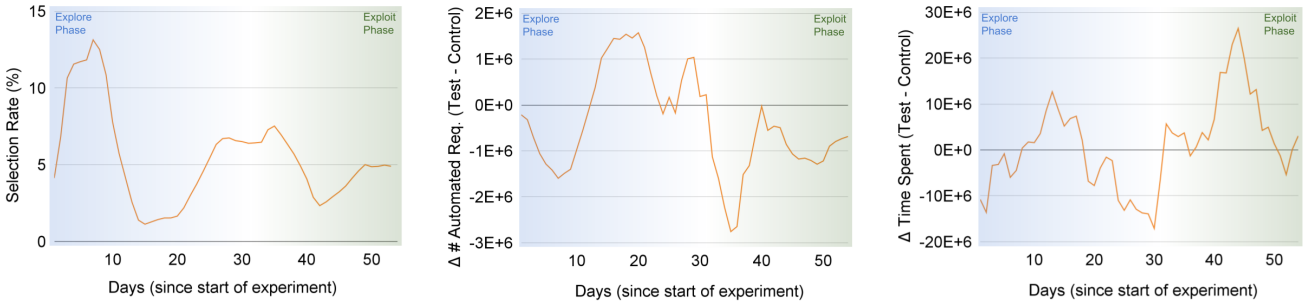
At the end of the experiment, we aggregated metrics from the final 5 days (Aug 25 to 29, 2023) comparing 1,057,156 accounts in Control with 1,055,797 accounts in Test. Results showed the new action led to a **3.0%** ($p = 0.008$) reduction in *automated requests*, and no statistically significant change in *time spent* ($0.7 \pm 1.6\%$ reduction; $p = 0.38$).

5.6 Uncontrolled systemic changes

Since our main results in Section 5.3 are based on data aggregated over a two-week period, an important question is how the system reacts to changes in feature distributions over longer periods of time ("concept drift"). We expect the PRO system to adapt automatically to such changes since we are retraining the models daily; here we share two case studies supporting this claim.

Case Study 4: Automatically adjusting to a bug. Engineers inadvertently introduced a bug into an identity-verification challenge on Instagram that asked account owners to upload photos of their face. This challenge was previously found to be effective at stopping abuse stemming from automated activity. However, the bug caused some enrolled accounts to remain in a "stuck" state with no ability to clear the challenge. After the bug manifested, new observed data points showed that the action had a significant negative impact on *Cost* metrics, which led model coefficients to change significantly after the model was retrained. As a result, PRO completely stopped selecting this action two days after the

[5] Since this experiment involved changing PRO's metric weights, which have a smaller effect than comparing PRO with a baseline selection algorithm, we increased the size of the experiment in order to ensure statistical significance.



(a) Selection Rate (b) Δ Automated Request Count (c) Δ Time Spent

Figure 3: (a) Selection rate of the new enforcement action. (b) Daily deltas (*Test* − *Control*) of the abuse metric *automated requests*. (c) Cost metric *time spent* (7-day moving average)

bug was observed, without engineers manually altering the configuration to disallow the action from being selected.

Case Study 5: Adjusting to adversarial adaptation. We incorporated a new "warning" challenge into the PRO system's action suite on Instagram. This challenge presents accounts suspected of automated activity with a warning notice and prevents any future web requests from being served until the account acknowledges the warning. We ran an online controlled experiment and aggregated metrics from Mar 24 to Apr 6, 2023 (14 days), comparing 803,813 accounts in Control with 803,753 accounts in Test. Results showed the addition of this new challenge led to a **15%** reduction ($p \ll 0.001$) in *weighted scraping requests* and no statistically significant change in *time spent* ($0.1 \pm 0.3\%$ increase; $p = 0.50$). In the Test group, PRO selected the new action **13%** of the time.

Based on these promising results, we increased the size of the experiment group. A month later (Apr 25 to May 8, 2023) we observed that the daily selection rate for the new action had a statistically significant drop ($p \ll 0.001$), falling to only **4%**. Data analysis revealed that traffic from some abusive entities resumed almost immediately after the warning challenge was presented to them, providing evidence that some adversaries had learned how to circumvent the challenge and the system needed to select other, more effective responses to stop them. Despite this adversarial adaptation, we found that overall the 14-day rolling average of *weighted scraping requests* decreased by 24% ($p = 0.006$) between Apr 6 and May 8, indicating that our changes did have beneficial impact on the overall scraping ecosystem.

6 Ethics considerations

We have assessed the value of publishing this work against potentially adverse consequences due to its methodology and/or data practices. Specifically, we considered risks related to *user harm*, *equitable selection*, *user consent*, and *user data handling* [71].

User harm. We acknowledge that fighting abuse on online social networks is a task fraught with risk: some users are harmed by the abuse itself, while others are harmed by over-enforcement when defenses become too aggressive. This entire work is devoted to systematically balancing reduction in both of these harms. Our system's "explore-exploit" strategy aims to reduce harm over the entire platform while taking into account that not all local decisions can be perfect. However, our experiments demonstrate that the PRO system offers significant benefit to users in terms of overall harm reduction.

We also considered the risk of our experiments over-enforcing on benign users due to system and/or model deficiencies. To mitigate this risk we set up our experiments to include only users that our binary classifiers predicted to be abusive with high confidence. The precision of the binary classifiers was confirmed to be greater than 90% at the time of the experiments. Given the above considerations, we believe that our experiment exposes users to risks that are "reasonable in relation to anticipated benefits" [71].

Equitable selection. Our subjects are necessarily chosen equitably since a user's participation in the experiment is determined by a random number generator whose output is independent of any user properties.

User consent. Partridge and Allman [60] observe that "direct consent is not possible in most Internet measurements," and our study is a good example. Since we cannot predict in advance who our classifiers will determine to be abusive, we would have to obtain consent either from the entire OSN population or from the specific users acted on by PRO at the moment action is taken. Building a bespoke consent flow for this study would be a large engineering task and would risk both information bias (users might act differently knowing they were in a security study) and selection bias (both benign and abusive users who choose to participate in a security study may not reflect the general population). Either type of bias could render our statistical analyses invalid and thus handicap our ability to measurably improve abuse detection.

Partridge and Allman suggest that "proxy consent" is the *de facto* standard in large-scale internet measurement studies, giving the example of "network measurements taken on a university campus typically seek consent from the university." In our case institutional consent was rendered via the OSNs' agreements to let us conduct and publish this research. In particular, the reviewers approving the research noted that all users in our study have accepted the terms of service of Instagram or Facebook (as applicable), which address use of data in the context of investigating suspicious activity and addressing policy violations. The reviewers therefore concluded that the OSN terms of service provide users a sufficient level of transparency.

We note that explicit consideration of user consent is not historically an element of large-scale internet security studies. Bilogrevic *et al.* [13] use a proxy approach similar to ours, deriving their user consent from the fact that users opted in to a setting to "Make searches and browsing better." However, recent work studying millions of users on Reddit [44], Facebook [25, 58], and Google Chrome [69] do not address user consent at all in their ethics discussions. We discuss open questions in this area in Section 7.

Data handling. Before developing and testing our system, we assessed how data would be used and protected and ensured that technical systems and/or manual processes were in place to mitigate any identified risks prior to the launching the experiment [54]. For this project, we mitigated risks by:

- Limiting data collection to a set of user features identified as being relevant for abuse detection;
- Specifically excluding any sensitive data from collection;
- Restricting access to both collected and inferred data;
- Deleting all user-identifying data within 90 days of collection;
- Using technical safeguards to ensure that the data are only used for safety, integrity, and security use cases.

7 Directions for Future Work

Optimizing long-term reward. PRO selects actions to maximize reward over a fixed time horizon. However, we may wish to select actions to reduce abuse and cost in the long term. We could view the RL problem as a "continuing" task, or as an infinite-horizon task instead of a finite-horizon one [68], and optimize using RL algorithms such as Q-learning [39, 55, 75]. For each metric, we could train on the sequence of actions taken on each entity and leverage Q-learning's ability to learn cumulative long-term discounted rewards.

Evaluating exploration strategies. Unlike supervised learning, RL involves an explore-exploit tradeoff. If PRO always issues the same action to an entity, it can never learn whether that action was actually the best choice. Having this feedback loop is even more important in an adversarial setting. However, comparing exploration strategies can be challenging. Simply A/B testing the same RL system with two exploration implementations will not work if the models share training data, as the Test model will "free-ride" on the Control model's exploration, or vice versa. Developing approaches to compare exploration methods would allow us to test other exploration strategies such as Upper Confidence Bound [17] or Quantile Regression-based sampling [40].

Addressing over- and under-enforcement. Case Study 1 describes one set of users on which the system made locally sub-optimal decisions. Other such populations include:

- "Power users," defined as non-malicious users who use the platform in such a way that their activity appears automated. The population of power users is so small that over-enforcement on this subset doesn't meaningfully impact the global cost metrics.
- "Repeat offenders," defined as users sent through the PRO system multiple times. If the system doesn't update features quickly enough then it risks repeating a response that was either too harsh for a benign user or not effective on a malicious user.
- "Low-information" users, who may use unauthorized third-party tools or otherwise inadvertently breach the OSN terms of service.

For each of these cases, we believe that some combination of new cost metrics (as in Case Study 1) and/or new enforcement actions (as in Case Study 3) can improve the model's performance.

Generalizing our solution to other anti-abuse use cases. Our experiments provide empirical evidence about our system yielding measurable improvements in reducing scraping of OSNs. Assessing how our solution can impact other abuse problems remains an open area of research.

Understanding consent in adversarial studies. In our discussion of user consent we asserted that "users might act differently knowing they were in a security study" and that "both benign and abusive users who choose to participate in a security study may not reflect the general population." These assertions have never been tested rigorously; a study that tested hypotheses on user consent in adversarial environments would provide crucial scientific input to the ethical standards for all future studies involving real-world adversaries.

Authors' Contributions

- Garrett Wilson developed model enhancements and observability infrastructure, drafted the technical portions of this paper, and coordinated the paper-revision process.
- Geoffrey Goh designed the experimentation process and developed core components including metrics, enforcement actions, serving infrastructure, and the model predictive controller.
- Yan Jiang extended PRO to Facebook and ran related experiments for this paper.
- Ajay Gupta and Jiaxuan Wang provided engineering support.
- David Freeman consulted in the initial design phase of PRO and coordinated the writing process.
- Francesco Dinuzzo created this project and led the team that built and maintained the PRO system. He designed the reinforcement learning approach to abuse mitigation, built the initial production versions of the system, and guided its development over multiple iterations.

Acknowledgments

The authors thank Katriel Cohn-Gordon, Feargus Pendlebury, Francesco Logozzo, Chris Palow, and Yiannis Papagiannis for helpful feedback on earlier drafts of this paper. We thank Sandeep Hebbani, Emile Litvak, and Gemma Silvers for encouraging publication of this paper. We thank the five anonymous reviewers for their feedback, and in particular the shepherd who helped us craft the paper into its current form.

References

[1] Naoki Abe, Prem Melville, Cezar Pendus, Chandan Reddy, David Jensen, Vince Thomas, James Bennett, Gary Anderson, Brent Cooley, Melissa Kowalczyk, et al. Optimizing debt collections using constrained reinforcement learning. In *16th Int'l Conference on Knowledge Discovery and Data Mining (KDD)*, pages 75–84, 2010.

[2] Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, volume 23, pages 39.1–39.26, 2012.

[3] Ala'M Al-Zoubi, Ja'far Alqatawna, Hossam Faris, and Mohammad A Hassonah. Spam profiles detection on social networks using computational intelligence methods: the effect of the lingual context. *Journal of Information Science*, 47(1):58–81, 2021.

[4] Mohammad Alauthman, Nauman Aslam, Mouhammd Al-Kasassbeh, Suleman Khan, Ahmad Al-Qerem, and Kim-Kwang Raymond Choo. An efficient reinforcement learning-based Botnet detection approach. *J. Network and Computer Applications*, 150:102479, 2020.

[5] Javier Arroyo, Carlo Manna, Fred Spiessens, and Lieve Helsen. Reinforced model predictive control (RL-MPC) for building energy management. *Applied Energy*, 309:118346, 2022.

[6] Karl Johan Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2008.

[7] Marco Balduzzi, Christian Platzer, Thorsten Holz, Engin Kirda, Davide Balzarotti, and Christopher Kruegel. Abusing social networks for automated user profiling. In *Recent Advances in Intrusion Detection (RAID): 13th International Symposium*, pages 422–441. Springer, 2010.

[8] M Tariq Banday and Nisar A Shah. A study of CAPTCHAs for securing web services. *arXiv preprint arXiv:1112.5605*, 2011.

[9] Angelo Barboni, Francesca Boem, and Thomas Parisini. Model-based detection of cyber-attacks in networked MPC-based control systems. *IFAC-PapersOnLine*, 51(24):963–968, 2018.

[10] Rafael Basso, Balázs Kulcsár, Ivan Sanchez-Diaz, and Xiaobo Qu. Dynamic stochastic electric vehicle routing with safe reinforcement learning. *Transportation research part E: Logistics and transportation review*, 157:102496, 2022.

[11] Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. Detecting spammers on Twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, volume 6, 2010.

[12] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *22nd International Conference on World Wide Web (WWW)*, pages 119–130, 2013.

[13] Igor Bilogrevic, Balazs Engedy, Judson L. Porter III, Nina Taft, Kamila Hasanbega, Andrew Paseltiner, Hwi Kyoung Lee, Edward Jung, Meggyn Watkins, PJ McLachlan, and Jason James. "Shhh...be quiet!" Reducing the unwanted interruptions of notification permission prompts on Chrome. In *30th USENIX Security Symposium*, pages 769–784, 2021.

[14] Garrett Brown, Travis Howe, Micheal Ihbe, Atul Prakash, and Kevin Borders. Social networks and context-aware spam. In *ACM Conference on Computer Supported Cooperative Work*, pages 403–412, 2008.

[15] Evandro Caldeira, Gabriel Brandao, and Adriano CM Pereira. Fraud analysis and prevention in e-commerce transactions. In *9th Latin American Web Congress*, pages 42–49. IEEE, 2014.

[16] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. Uncovering large groups of active malicious accounts in online social networks. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 477–488, 2014.

[17] Alexandra Carpentier, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos, and Peter Auer. Upper-confidence-bound algorithms for active learning in multi-armed bandits. In *International Conference on Algorithmic Learning Theory*, pages 189–203, 2011.

[18] Sandeep Chinchali, Pan Hu, Tianshu Chu, Manu Sharma, Manu Bansal, Rakesh Misra, Marco Pavone, and Sachin Katti. Cellular network traffic scheduling with deep reinforcement learning. In *32nd AAAI Conference on Artificial Intelligence*, 2018.

[19] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

[20] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *10th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 99–108, 2004.

[21] George Danezis and Prateek Mittal. SybilInfer: detecting sybil nodes using social networks. In *NDSS*, pages 1–15, 2009.

[22] Louis F DeKoven, Trevor Pottinger, Stefan Savage, Geoffrey M Voelker, and Nektarios Leontiadis. Following their footsteps: Characterizing account automation abuse and defenses. In *Internet Measurement Conference (IMC)*, pages 43–55, 2018.

[23] Ahmed ELazab and MA Mahmoud. Fraud news detection for online social networks. *Artif Intell Syst Mach Learn*, 10(8):177–182, 2018.

[24] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y Zhao. Detecting and characterizing social spam campaigns. In *10th ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 35–47, 2010.

[25] Maximilian Golla, Grant Ho, Marika Lohmus, Monica Pulluri, and Elissa M. Redmiles. Driving 2FA adoption at scale: Optimizing two-factor authentication notification design patterns. In *30th USENIX Security Symposium*, pages 109–126, 2021.

[26] Gene H Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good Ridge parameter. *Technometrics*, 21(2):215–223, 1979.

[27] Neil Zhenqiang Gong, Mario Frank, and Prateek Mittal. Sybilbelief: A semi-supervised learning approach for structure-based sybil detection. *IEEE Transactions on Information Forensics and Security*, 9(6):976–987, 2014.

[28] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @ spam: the underground on 140 characters or less. In *17th ACM conference on Computer and Communications Security (CCS)*, pages 27–37, 2010.

[29] Shangding Gu, Guang Chen, Lijun Zhang, Jing Hou, Yingbai Hu, and Alois Knoll. Constrained reinforcement learning for vehicle motion planning with topological reachability analysis. *Robotics*, 11(4):81, 2022.

[30] Meriem Guerar, Luca Verderame, Mauro Migliardi, Francesco Palmieri, and Alessio Merlo. Gotta CAPTCHA'em all: A survey of 20 years of the human-or-computer dilemma. *ACM Computing Surveys (CSUR)*, 54(9):1–33, 2021.

[31] Mohammad Reza Habibi, Hamid Reza Baghaee, Frede Blaabjerg, and Tomislav Dragičević. Secure MPC/ANN-based false data injection cyber-attack detection and mitigation in DC microgrids. *IEEE Systems Journal*, 16(1):1487–1498, 2021.

[32] J.T. Hancock and T.M. Khoshgoftaar. Survey on categorical data for neural networks. *J. Big Data*, 7, 2020.

[33] Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[34] Kailas S Holkar and Laxman M Waghmare. An overview of model predictive control. *International Journal of Control and Automation*, 3(4):47–63, 2010.

[35] David Isele, Alireza Nakhaei, and Kikuo Fujimura. Safe reinforcement learning on autonomous vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–6. IEEE, 2018.

[36] Tom N Jagatic, Nathaniel A Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.

[37] Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. *Safe reinforcement learning using probabilistic shields*. Dagstuhl: Schloss Dagstuhl, 2020.

[38] Meng Jiang, Peng Cui, and Christos Faloutsos. Suspicious behavior detection: Current trends and future directions. *IEEE Intelligent Systems*, 31(1):31–39, 2016.

[39] Gabriel Kalweit, Maria Huegle, Moritz Werling, and Joschka Boedecker. Deep constrained Q-learning. *arXiv preprint arXiv:2003.09398*, 2020.

[40] Roger Koenker and Kevin F. Hallock. Quantile regression. *Journal of Economic Perspectives*, 15(4):143–156, December 2001.

[41] Ron Kohavi, Alex Deng, Brian Frasca, Toby Walker, Ya Xu, and Nils Pohlmann. Online controlled experiments at large scale. In *19th International Conference on Knowledge Discovery and Data Mining (KDD)*, page 1168–1176. ACM, 2013.

[42] Hanna Krasowski, Xiao Wang, and Matthias Althoff. Safe reinforcement learning for autonomous lane changing using set-based prediction. In *23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7. IEEE, 2020.

[43] Pavlo Krokhmal, Jonas Palmquist, and Stanislav Uryasev. Portfolio optimization with conditional value-at-risk objective and constraints. *Journal of Risk*, 4:43–68, 2002.

[44] Deepak Kumar, Jeff Hancock, Kurt Thomas, and Zakir Durumeric. Understanding the behaviors of toxic accounts on reddit. In *Proceedings of The Web Conf*, 2023.

[45] Hepeng Li, Zhiqiang Wan, and Haibo He. Constrained EV charging scheduling based on safe deep reinforcement learning. *IEEE Transactions on Smart Grid*, 11(3):2427–2439, 2019.

[46] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *19th International Conference on World Wide Web (WWW)*. ACM, 2010.

[47] Yixuan Li, Oscar Martinez, Xing Chen, Yi Li, and John E Hopcroft. In a world that counts: Clustering and detecting fake social engagement at scale. In *25th International Conference on World Wide Web (WWW)*, pages 111–120, 2016.

[48] Mee Hong Ling, Kok-Lim Alvin Yau, Junaid Qadir, Geong Sen Poh, and Qiang Ni. Application of reinforcement learning for security enhancement in cognitive radio networks. *Applied Soft Computing*, 37:809–829, 2015.

[49] Lennart Ljung. *System Identification: Theory for the User (2nd Edition)*. Prentice Hall, 2 edition, 1999.

[50] OB Longe, ABC Robert, and U Onwudebelu. Checking internet masquerading using multiple CAPTCHA challenge-response systems. In *2009 2nd International Conference on Adaptive Science & Technology (ICAST)*, pages 244–249. IEEE, 2009.

[51] Michael Mccord and M Chuah. Spam detection on Twitter using traditional classifiers. In *Autonomic and Trusted Computing: 8th International Conference, (ATC)*, pages 175–186. Springer, 2011.

[52] Ali Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[53] Meta Platforms, Inc. Meta Transparency Center. https://transparency.fb.com/enforcement/taking-action/restricting-accounts/, 2024.

[54] Meta Platforms, Inc. Privacy progress update. https://about.meta.com/privacy-progress, 2024.

[55] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[56] MUSA. Mitigating Unauthorized Scraping Alliance. https://antiscrapingalliance.org/, 2024.

[57] Aleksandr Ometov, Sergey Bezzateev, Niko Mäkitalo, Sergey Andreev, Tommi Mikkonen, and Yevgeni Koucheryavy. Multi-factor authentication: A survey. *Cryptography*, 2:1, 2018.

[58] Jeremiah Onaolapo, Nektarios Leontiadis, Despoina Magka, and Gianluca Stringhini. SocialHEISTing: Understanding stolen Facebook accounts. In *30th USENIX Security Symposium*, pages 4115–4132, 2021.

[59] Mariam Orabi, Djedjiga Mouheb, Zaher Al Aghbari, and Ibrahim Kamel. Detection of bots in social media: A systematic review. *Information Processing & Management*, 57(4):102250, 2020.

[60] C. Partridge and M. Allman. Ethical considerations in network measurement papers. *Communications of the ACM*, 59:58–64, 2016.

[61] Clark Pope and Khushpreet Kaur. Is it human or computer? defending e-commerce with CAPTCHAs. *IT Professional*, 7(2):43–49, 2005.

[62] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[63] Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Ann. Rev. Control, Robotics, & Autonomous Systems*, 2:253–279, 2019.

[64] Andrew Searles, Yoshimichi Nakatsuka, Ercan Ozturk, Andrew Paverd, Gene Tsudik, and Ai Enkoji. An empirical study & evaluation of modern CAPTCHAs. In *32nd USENIX Security Symposium*, pages 3081–3097, 2023.

[65] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[66] Arambam James Singh, Akshat Kumar, and Hoong Chuin Lau. Hierarchical multiagent reinforcement learning for maritime traffic management. *IFAAMAS*, 2020.

[67] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Detecting spammers on social networks. In *26th Annual Computer Security Applications Conference*, pages 1–9, 2010.

[68] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[69] Kurt Thomas, Jennifer Pullman, Kevin Yeo, Ananth Raghunathan, Patrick Gage Kelley, Luca Invernizzi, Borbala Benko, Tadek Pietraszek, Sarvar Patel, Dan Boneh, and Elie Bursztein. Protecting accounts from credential stuffing with password breach alerting. In *28th USENIX Security Symposium*, 2019.

[70] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[71] United States National Archives and Records Administration. Criteria for IRB approval of research. 45 CFR 46.111 (July 19, 2018), https://www.ecfr.gov/on/2018-07-19/title-45/section-46.111.

[72] Aashma Uprety and Danda B Rawat. Reinforcement learning for IoT security: A comprehensive survey. *IEEE Internet of Things Journal*, 8(11):8693–8706, 2020.

[73] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using hard AI problems for security. In *Advances in Cryptology – Eurocrypt 2003*, pages 294–311. Springer, 2003.

[74] Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y. Zhao. You are how you click: Clickstream analysis for sybil detection. In *22nd Usenix Security Symposium*, 2013.

[75] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[76] Leland Wilkinson. Revising the Pareto chart. *The American Statistician*, 60(4):332–334, Nov 2006.

[77] Christopher KI Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In *Learning in Graphical Models*, pages 599–621. Springer, 1998.

[78] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[79] Cao Xiao, David Mandell Freeman, and Theodore Hwa. Detecting clusters of fake accounts in online social networks. In *8th ACM Workshop on Artificial Intelligence and Security (AISec)*, pages 91–101, 2015.

[80] Teng Xu, Gerard Goossen, Huseyin Kerem Cevahir, Sara Khodeir, Yingyezhe Jin, Frank Li, Shawn Shan, Sagar Patel, David Freeman, and Paul Pearce. Deep entity classification: Abusive account detection for online social networks. In *30th USENIX Security Symposium*, 2021.

[81] Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y. Zhao, and Yafei Dai. Uncovering social network sybils in the wild. In *Internet Measurement Conference*, 2011.

[82] YouTube. YouTube Community Guidelines And Policies. https://www.youtube.com/howyoutubeworks/policies/community-guidelines/#taking-action-on-violations, 2024.

[83] Mario Zanon and Sébastien Gros. Safe reinforcement learning using robust MPC. *IEEE Transactions on Automatic Control*, 66(8):3638–3652, 2020.

[84] Xiangyu Zhao, Changsheng Gu, Haoshenglun Zhang, Xiwang Yang, Xiaobing Liu, Jiliang Tang, and Hui Liu. DEAR: Deep reinforcement learning for online advertising impression in recommender systems. In *35th AAAI Conference on Artificial Intelligence*, pages 750–758, 2021.