Man vs. Machine: Practical Adversarial Detection of Malicious Crowdsourcing Workers

Gang Wang[†], Tianyi Wang[†‡], Haitao Zheng[†] and Ben Y. Zhao[†]
[†]*Computer Science, UC Santa Barbara* [‡]*Electronic Engineering, Tsinghua University*
{*gangw, tianyi, htzheng, ravenben*}*@cs.ucsb.edu*

Abstract

Recent work in security and systems has embraced the use of machine learning (ML) techniques for identifying misbehavior, *e.g.* email spam and fake (Sybil) users in social networks. However, ML models are typically derived from *fixed* datasets, and must be periodically retrained. In adversarial environments, attackers can adapt by modifying their behavior or even sabotaging ML models by polluting training data.

In this paper[1], we perform an empirical study of adversarial attacks against machine learning models in the context of detecting malicious crowdsourcing systems, where sites connect paying users with workers willing to carry out malicious campaigns. By using human workers, these systems can easily circumvent deployed security mechanisms, *e.g.* CAPTCHAs. We collect a dataset of malicious workers actively performing tasks on Weibo, China's Twitter, and use it to develop ML-based detectors. We show that traditional ML techniques are accurate (95%–99%) in detection but can be highly vulnerable to adversarial attacks, including simple *evasion attacks* (workers modify their behavior) and powerful *poisoning attacks* (where administrators tamper with the training set). We quantify the robustness of ML classifiers by evaluating them in a range of practical adversarial models using ground truth data. Our analysis provides a detailed look at practical adversarial attacks on ML models, and helps defenders make informed decisions in the design and configuration of ML detectors.

1 Introduction

Today's computing networks and services are extremely complex systems with unpredictable interactions between numerous moving parts. In the absence of accurate deterministic models, applying Machine Learning (ML) techniques such as decision trees and support vector machines (SVMs) produces practical solutions to a variety of problems. In the security context, ML techniques can extract statistical models from large noisy datasets, which have proven accurate in detecting misbehavior and attacks, *e.g.* email spam [35, 36], network intrusion attacks [22, 54], and Internet worms [29]. More recently, researchers have used them to model and detect malicious users in online services, *e.g.* Sybils in social networks [42, 52], scammers in e-commerce sites [53] and fraudulent reviewers on online review sites [31].

Despite a wide range of successful applications, machine learning systems have a weakness: they are vulnerable to adversarial countermeasures by attackers aware of their use. First, through either reading publications or self-experimentation, attackers may become aware of details of the ML detector, *e.g.* choice of classifier and parameters used, and modify their behavior to *evade* detection. Second, more powerful attackers can actively tamper with the ML models by polluting the training set, reducing or eliminating its efficacy. Adversarial machine learning has been studied by prior work from a theoretical perspective [6, 12, 27], using simplistic all-or-nothing assumptions about adversaries' knowledge about the ML system in use. In reality, however, attackers are likely to gain incomplete information or have partial control over the system. An accurate assessment of the robustness of ML techniques requires evaluation under *realistic* threat models.

In this work, we study the robustness of machine learning models against practical adversarial attacks, in the context of detecting malicious crowdsourcing activity. Malicious crowdsourcing, also called crowdturfing, occurs when an attacker pays a group of Internet users to carry out malicious campaigns. Recent crowdturfing attacks ranged from "artificial grassroots" political campaigns [32, 38], product promotions that spread false rumors [10], to spam dissemination [13, 39]. Today, these campaigns are growing in popularity in dedicated

[1]Our work received approval from our local IRB review board.

crowdturfing sites, *e.g.* ZhuBaJie (ZBJ)[2] and SanDaHa (SDH)[3], and generic crowdsourcing sites [26, 48].

The detection of crowdturfing activity is an ideal context to study the impact of adversarial attacks on machine learning tools. First, crowdturfing is a growing threat to today's online services. Because tasks are performed by intelligent individuals, these attacks are undetectable by normal measures such as CAPTCHAs or rate limits. The results of these tasks, fake blogs, slanderous reviews, fake social network accounts, are often indistinguishable from the real thing. Second, centralized crowdturfing sites like ZBJ and SDH profit directly from malicious crowdsourcing campaigns, and therefore have strong monetary incentive and the capability to launch adversarial attacks. These sites have the capability to modify aggregate behavior of their users through interface changes or explicit policies, thereby either helping attackers evade detection or polluting data used as training input to ML models.

Datasets. For our analysis, we focus on Sina Weibo, China's microblogging network with more than 500 million users, and a frequent target of crowdturfing campaigns. Most campaigns involve paying users to retweet spam messages or to follow a specific Weibo account. We extract records of 20,416 crowdturfing campaigns (1,012,923 tasks) published on confirmed crowdturfing sites over the last 3 years. We then extract a 28,947 Weibo accounts belonging to crowdturfing workers. We analyze distinguishing features of these accounts, and build detectors using multiple ML models, including SVMs, Bayesian, Decision Trees and Random Forests.

We seek answers to several key questions. First, can machine learning models detect crowdturfing activity? Second, once detectors are active, what are possible countermeasures available to attackers? Third, can adversaries successfully manipulate ML models by tampering with training data, and if so, can such efforts succeed in practice, and which models are most vulnerable?

Adversarial Attack Models. We consider two types of practical adversarial models against ML systems: those launched by individual *crowd-workers*, and those involving coordinated behavior driven by administrators of centralized *crowdturfing sites*. First, individual workers can perform *evasion* attacks, by adapting behavior based on their knowledge of the target classifier (*e.g.* ML algorithms, feature space, trained models). We identify a range of threat models that vary the amount of knowledge by the adversary. The results should provide a comprehensive view of how vulnerable ML systems to evasion, ranging from the worst case (total knowledge by attacker) to more practically scenarios. Second, more powerful attacks are possible with the help of crowdturfing site administrators, who can manipulate ML detectors by *poisoning* or polluting training data. We study the impact on different ML algorithms from two pollution attacks: injecting false data samples, and altering existing data samples.

Our study makes four key contributions:

- We demonstrate the efficacy of ML models for detecting crowdturfing activity. We find that Random Forests perform best out of multiple classifiers, with 95% detection accuracy overall and 99% for "professional" workers.
- We develop adversarial models for *evasion attacks* ranging from optimal evasion to more practical/limited strategies. We find while such attacks can be very powerful in the optimal scenario (attacker has total knowledge), practical attacks are significantly less effective.
- We evaluate a powerful class of *poison* attacks on ML training data and find that *injecting* carefully crafted data into training data can significantly reduce detection efficacy.
- We observe a consistent tradeoff between fitting accuracy and robustness to adversarial attacks. More accurate fits (especially to smaller, homogeneous populations) make models more vulnerable to deviations introduced by adversaries. The exception is Random Forests, which naturally supports fitting to multiple populations, thus allowing it to maintain both accuracy and robustness in our tests.

To the best of our knowledge, this is the first study to examine automated detection of large-scale crowdturfing activity, and the first to evaluate adversarial attacks against machine learning models in this context. Our results show that accurate models are often vulnerable to adversarial attacks, and that robustness against attacks should be a primary concern when selecting ML models.

2 Datasets and Methodology

In this section, we provide background on crowdturfing, and introduce our datasets and methodology.

2.1 Background: Crowdturfing Systems

Malicious crowdsourcing (crowdturfing) sites are web services where attackers pay groups of human workers to perform questionable (and often malicious) tasks. While these services are growing rapidly world-wide, two of the largest are Chinese sites ZhuBaJie (ZBJ) and SanDaHa (SDH) [48]. Both sites leave records of campaigns publicly visible to recruit new workers, making it possible for us to crawl their data for analysis.

[2] `http://www.zhubajie.com/c-tuiguang/`
[3] `http://www.sandaha.com/`

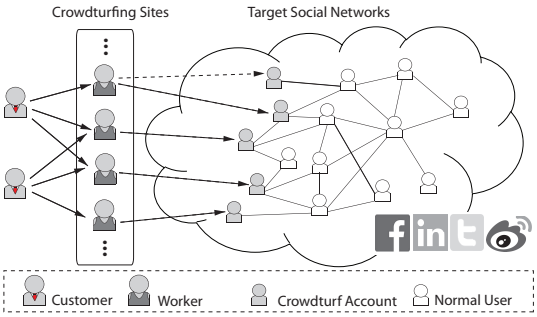


Figure 1: Crowdturfing process.

Figure 1 illustrates how crowdturfing campaigns work. Initially, a *customer* posts a campaign onto the crowdturfing site, and pays the site to carry it out. Each campaign is a collection of small *tasks*, *e.g.* tasks to send or retweet messages advertising a malware site. *Workers* accept the task, and use their fake accounts in the target social network(s) (*e.g.* Twitter) to carry out the tasks. Today, crowdturfing campaigns often spam web services such as social networks, online review sites, and instant-messaging networks [48]. While workers can be any Internet user willing to spam for profit, customers often require workers to use "high quality" accounts (*i.e.* established accounts with real friends) to perform tasks [48]. In the rest of the paper, we refer workers' social network accounts as *crowdturf accounts*.

Crowdturfing on Weibo. Sina Weibo is China's most popular microblogging social network with over 500 million users [30]. Like Twitter, Weibo users post 140-character *tweets*, which can be *retweeted* by other users. Users can also *follow* each other to form asymmetric social relationships. Unlike Twitter, Weibo allows users to have conversations via *comments* on a tweet.

Given its large user population, Weibo is a popular target for crowdturfing systems. There are two major types of crowdturfing campaigns. One type asks workers to follow a customer's Weibo account to boost their perceived popularity and visibility in Weibo's ranked social search. A second type pays crowd-workers to retweet spam messages or URLs to reach a large audience. Both types of campaigns directly violate Weibo's ToS [2]. A recent statement (April 2014) from a Weibo administrator shows that Weibo has already begun to take action against crowdturfing spam [1].

2.2 Ground Truth and Baseline Datasets

Our study utilizes a large *ground-truth* dataset of crowdturfing worker accounts. We extract these accounts by filtering through records of all campaigns and tasks targeting Weibo from ZBJ and SDH, and extracting all Weibo accounts that accepted these tasks. This is possible because ZBJ and SDH keep complete records of campaigns and transaction details (*i.e.* workers who completed tasks, and their Weibo identities) visible.

Category	# Weibo IDs	# (Re) Tweets	# Comments
Turfing	28,947	18,473,903	15,970,215
Authent.	71,890	7,600,715	13,985,118
Active	371,588	34,164,885	75,335,276

Table 1: Dataset summary.

As of March 2013, we collected a total of 20,416 Weibo campaigns (over 3 years for ZBJ and SDH), with a total of 1,012,923 individual tasks. We extracted 34,505 unique Weibo account IDs from these records. 5,558 of which have already been blocked by Weibo. We collected user profiles for the remaining 28,947 active accounts, including social relationships and the latest 2000 tweets from each account. These accounts have performed at least one crowdturfing task. We refer to this as the *Turfing* dataset.

Baseline Datasets for Comparison. We need a baseline dataset of "normal" users for comparison. We start by snowball sampling a large collection of Weibo accounts[4]. We ran breadth-first search (BFS) in November 2012 using 100 Seeds randomly chosen from Weibo's public tweet stream, giving us 723K accounts. Because these crawled accounts can include malicious accounts, we need to do further filtering to obtain a real set of "normal" users.

We extract two different baseline datasets. First, we construct a conservative *Authenticated* dataset, by including only Weibo users who have undergone an optional identity verification by phone number or Chinese national ID (equivalent to US drivers license). A user who has bound her Weibo account to her real-world identity can be held legally liable for her actions, making these authenticated accounts highly unlikely to be used as crowdturfing activity. Our *Authenticated* dataset includes 71,890 accounts from our snowball sample. Second, we construct a larger, more inclusive baseline set of *Active* users. We define this set as users with at least 50 followers and 10 tweets (filtering out dormant accounts[5] and Sybil accounts with no followers). We also cross reference these users against all known crowdturfing sites to remove any worker accounts. The resulting dataset includes 371,588 accounts. While it is not guaranteed to be 100% legitimate users, it provides a broader user sample that is more representative of average user behavior.

[4]Snowball crawls start from an initial set of seed nodes, and runs breadth-first search to find all reachable nodes in the social graph [3].

[5]Dormant accounts are unlikely to be workers. To qualify for jobs, ZBJ/SDH workers must meet minimum number of followers/tweets.

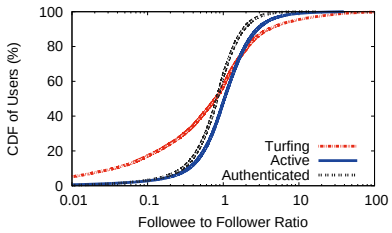


Figure 2: Followee-to-Follower ratio.

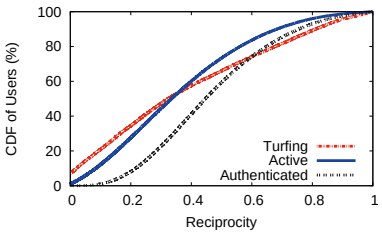


Figure 3: Reciprocity.

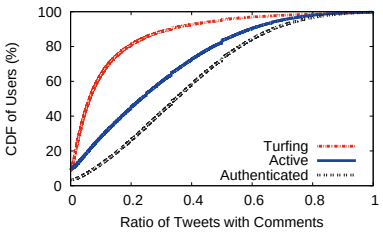


Figure 4: Ratio of commented tweets.

This is likely to provide a lower bound for detector accuracy, since more carefully curated baselines would produce higher detection accuracy. Our datasets are listed in Table 1.

2.3 Our Methodology

We have two goals: evaluating the efficacy of ML classifiers to detect crowdturfing workers, and evaluating the practical impact of adversarial attacks on ML classifiers.

- We analyze ground-truth data to identify key behavioral features that distinguish crowdturfing worker accounts from normal users (§3).
- We use these features to build a number of popular ML models, including Bayesian probabilistic models via Bayes' theorem (*i.e.* conditional probability), Support Vector Machines (SVMs), and algorithms based on single or multiple decision trees (*e.g.* Decision Trees, Random Forests) (§4).
- We evaluate ML models against adversarial attacks ranging from weak to strong based on level of knowledge by attackers (typically evasion attacks), and coordinated attacks potentially guided by centralized administrators (possible poison or pollution of training data).

3 Profiling Crowdturf Workers

We begin our study by searching for behavioral features that distinguish worker accounts from normal users. These features will be used to build ML detectors in §4.

User Profile Fields. We start with user profile features commonly used as indicators of abnormal behavior. These features include followee-to-follower ratio (FFRatio), reciprocity (*i.e.* portion of user's followees who follow back), user tweets per day, account age, and ratio of tweets with URLs and mentions.

Unfortunately, our data shows that most of these features alone cannot effectively distinguish worker accounts from normal users. First, FFRatio and reciprocity are commonly used to identify malicious spammers [4, 43, 50]. Intuitively, spammers follow a large number of random users and hope for them to follow back, thus they have high FFRatio and low reciprocity. However, our analysis shows worker accounts have balanced FFRatios, the majority of them even have more followers than followees (Figure 2), and their reciprocity is very close to those of normal users (Figure 3). Other profile features are also ineffective, including account age, tweets per day, ratio of tweets with URLs and mentions. For example, existing detectors usually assume attackers create many "fresh" accounts to spam [4, 43], thus account age has potential. But we find that more than 75% of worker accounts in our dataset have been active for at least one year.

These results show that crowd-worker accounts in many respects resemble normal users, and are not easily detected by profile features alone [47].

User Interactions. Next, we move on to features related to user interactions. The intuition is that crowdturf workers are task-driven, and log on to work on tasks, but spend minimal time interacting with others. User interactions in Weibo are dominated by comments and retweets. We perform analysis on both of them and find consistent results which show they are good metrics to distinguish workers from non-workers. For brevity, we limit our discussion to results on comment interactions.

Figure 4 shows crowdturf accounts are less likely to receive comments on their tweets. For 80% of crowdturf accounts, less than 20% of their tweets are commented; while for 70% of normal users, their ratio of commented tweets exceeds 20%. This makes sense, as the fake content posted by crowdturf workers may not be interesting enough for others to comment on. We also examine the number of people that each user has bidirectional comments with (bi-commentors). Crowdturf workers rarely interact with other users, with 66% of accounts having at most one bi-commentor.

Tweeting Clients. Next we look at the use of tweeting clients (devices). We can use the "device" field associated with each tweet to infer how tweets are sent. Tweet clients fall into four categories: web-based browsers, apps on mobile devices, third-party account management

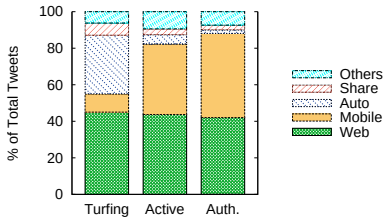


Figure 5: Tweet client usage.

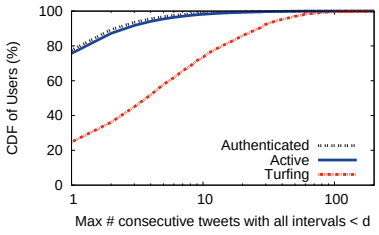


Figure 6: Max size of tweeting burst (threshold $d = 1$ minute).

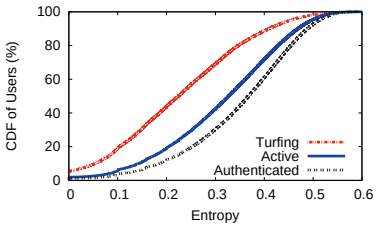


Figure 7: Normalized entropy of tweeting inter-arrival time.

Categ.	Top Tweet Clients
Web	Weibo Web, Weibo PC, 360Browser, Weibo Pro.
Mobile	iPhone, Android, iPad, XiaoMi
Auto	PiPi, Good Nanny, AiTuiBao, Treasure Box
Share	Taobao, Youku, Sina Blog, Baidu

Table 2: High-level categories for tweeting clients.

tools, and third-party websites via "share" buttons (Table 2). Figure 5 shows key differences in how different users use tweet clients. First, crowdturf workers use mobile (10%) much less than normal users (36% – 46%). One reason is that crowdturf workers rely on web browsers to interact with crowdturfing sites to get (submit) tasks and process payment, actions not supported by most mobile platforms.

We also observe that crowdturf workers are more likely to use automated tools. A close inspection shows that workers use these tools to automatically post non-spam tweets retrieved from a central content repository (*e.g.* a collection of hot topics). Essentially, crowdturf accounts use these generic tweets as cover traffic for their crowdturfing content. Third, crowdturf accounts "share" from third-party websites more often, since that is a common request in crowdturfing tasks [48].

Temporal Behavior. Finally, we look at temporal characteristics of tweeting behavior: tweet burstiness and periodicity. First, we expect task-driven workers to send many tweets in a short time period. We look for potential bursts, where each burst is defined as m consecutive tweets with inter-arrival times $< d$. We examine each user's maximum burst size (m) with different time thresholds d, *e.g.* Figure 6 depicts the result for d is set to 1 minute. We find that crowdturf accounts are more likely to post consecutive tweets within one-minute, something rarely seen from normal users. In addition, crowdturf workers are more likely to produce big bursts (*e.g.* 10 consecutive tweets with less than one-minute interval).

Second, workers accept tasks periodically, which can leave regular patterns in the timing of their tweets. We use *entropy* to characterize this regularity [16], where low entropy indicates a regular process while high entropy indicates randomness of tweeting. We treat each user's tweeting inter-arrival time as a random variable, and compute the first-order entropy [16]. Figure 7 plots user's entropy, normalized by the largest entropy in our dataset. Compared to normal users, crowdturf accounts in general have lower entropy, indicating their tweeting behaviors have stronger periodic patterns.

4 Detecting Crowdturfing Workers

We now use the features we identified to build a number of crowdturfing detectors using machine learning models. Here, we summarize the set of features we use for detection, and then build and evaluate a number of machine-learning detectors using our ground-truth data.

4.1 Key Features

We chose for our ML detectors a set of 35 features across five categories shown below.

- *Profile Fields (9).* We use 9 user profile fields[6] as features: follower count, followee count, followee-to-follower ratio, reciprocity, total tweet count, tweets per day, mentions per tweet, percent of tweets with mentions, and percent of tweets with embedded URLs.
- *User Interactions (8).* We use 8 features based on user interactions, *i.e.* comments and retweets. 4 features are based on user comments: percent of tweets with comments, percent of all comments that are outgoing, number of bi-commentors, and comment h-index (a user with h-index of h has at least h tweets each with at least h comments). We include 4 analogous retweet features.
- *Tweet Clients (5).* We compute and use the % of tweets sent from each tweet client type (web, mobile, automated tools, third-party shares and others) as a feature.

[6] Although profile fields *alone* cannot effectively detect crowdturf accounts (§3), they are still useful when combined with other features.

Alg.	Settings
NB	Default
BN	Default, K2 function
SVMr	Kernel γ =1, Cost parameter C =100
SVMp	Kernel degree d =3, Cost parameter C =50
J48	Confidence factor C =0.25, Instance/leaf M =2
RF	20 trees, 30 features/tree

Table 3: Classifier configurations.

- *Tweet Burstiness (12).* These 12 features capture the size and number of tweet bursts. A burst is m consecutive tweets where gaps between consecutive tweets are at most d minutes. For each user, we first compute the maximum burst size (m) while varying threshold d from 0.5 to 1, 30, 60, 120, 1440. Then we set d to 1 minute, and compute the number of bursts while varying size m over 2, 5, 10, 50, 100, and 500.
- *Tweeting Regularity (1).* This is the entropy value computed from each user's tweeting time-intervals.

4.2 Classification Algorithms

With these features, we now build classifiers to detect crowdturf accounts. We utilize a number of popular algorithms widely used in security contexts, including two Bayesian methods: Naive Bayesian (NB) [20] and BayesNet (BN) [18]; two Support Vector Machine methods [33]: SVM with radial basis function kernel (SVMr) and SVM with polynomial kernel (SVMp); and two Tree-based methods: C4.5 Decision Tree (J48[7]) [34] and Random Forests (RF) [7]. We leverage existing implementations of these algorithms in WEKA [17] toolkits.

Classifier and Experimental Setup. We start by constructing two experimental datasets, each containing all 28K turfing accounts, plus 28K randomly sampled baseline users from the "authenticated" and "active" sets. We refer to them as *Authenticated+Turfing* and *Active+Turfing*.

We use a small sample of ground-truth data to tune the parameters of different classifiers. At a high-level, we use grid search to locate the optimized parameters based on cross-validation accuracy. For brevity, we omit the details of the parameter tuning process and give the final configurations in Table 3. Note that features are normalized for SVM algorithms (we tested unnormalized approach which produced higher errors). We use this configuration for the rest of our experiments.

Basic Classification Performance. We run each classification algorithm on both experimental datasets with 10-fold cross-validation.[8] Figure 8 presents their classification error rates, including false positives (classifying normal users as crowdturf workers) and false negatives (classifying crowdturf accounts as normal users).

[7]J48 is WEKA's C4.5 implementation.

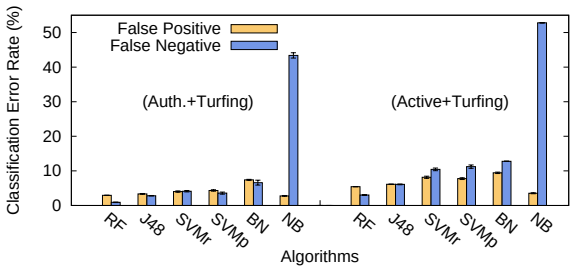


Figure 8: Classification error rates. Tree-based algorithms and SVMs outperform Bayesian methods.

We make four key observations. First, the two simple Bayesian methods generally perform worse than other algorithms. Second, Decision Tree (J48) and Random Forests (RF) are more accurate than SVMs. This is consistent with prior results that show SVMs excel in addressing high-dimension problems, while Tree algorithms usually perform better when feature dimensionality is low (35 in our case) [8]. Third, Random Forests outperform Decision Tree. Intuitively, Random Forests construct multiple decision trees from training data, which can more accurately model the behaviors of multiple types of crowdturf workers [7]. In contrast, decision tree would have trouble fitting distinct types of worker behaviors into a single tree. Finally, we observe that the two experiment datasets show consistent results in terms of relative accuracy across classifiers.

Comparing the two datasets, it is harder to differentiate crowdturf workers from *active* users than from *authenticated* users. This is unsurprising, since *authenticated* accounts often represent accounts of public figures, while *active* users are more likely to be representative of the normal user population. In the rest of the experiments, wherever the two datasets show consistent results, we only present the results on *Active+Turfing* for brevity, which captures the worse case accuracy for detectors.

4.3 Detecting Professional Workers

Our machine learning detectors are generally effective in identifying worker accounts. However, the contribution of tasks per worker is quite skewed, *i.e.* 90% of all tasks are completed by the top 10% most active "professional" workers (Figure 9). Intuitively, these "professional workers" are easier to detect than one-time workers. By focus-

[8]Cross-validation is used to compare the performance of different algorithms. We will split the data for training and testing the detectors later.

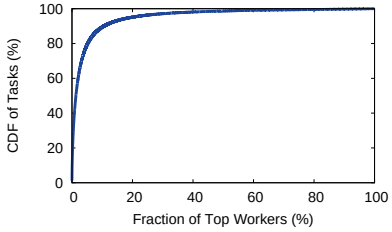


Figure 9: % of Tasks finished by top % of workers. The majority of spams were produced by top active workers.

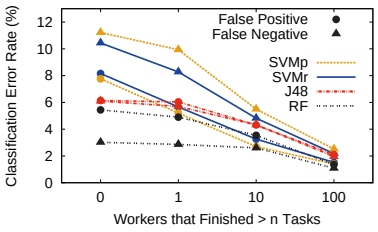


Figure 10: Classifying different levels of workers. Workers are filtered by # of crowdturfing tasks finished.

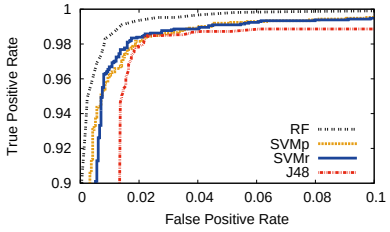


Figure 11: ROC curves of classifying professional workers (workers who finished more than 100 tasks).

ing on them, we can potentially improve detection accuracy while still effectively eliminate the largest majority of crowdturf output.

We evaluate classifier accuracy in detecting professional workers, by setting up a series of datasets each consisting of workers who performed more than n tasks (with n set to 1, 10, and 100). Each dataset also contains an equal number of randomly sampled normal users. We focus on the most accurate algorithms: Random Forests (RF), Decision Tree (J48) and SVM (SVMr and SVMp), and run 10-fold cross-validation on each of the datasets.

Figure 10 shows the classification results on *Active+Turfing*. As expected, our classifiers are more accurate in identifying "professional" workers. Different algorithms converge in accuracy as we raise the minimum productivity of professional workers. Accuracy is high for crowdturf workers who performed $>$100 tasks: Random Forests only produce 1.2% false positive rate and 1.1% false negative rate (99% accuracy). Note that while these top workers are only 8.9% of the worker population, they are responsible for completing 90% of all tasks. In the rest of the paper, we use "professional workers" to refer to workers who have completed $>$100 tasks.

False Positives vs. False Negatives. In practice, different application scenarios will seek different tradeoffs between false positives (FP) and false negatives (FN). For example, a system used as a pre-filter before more sophisticated tools (*e.g.* manual examination) will want to minimize FN, while an independent system without additional checks will want to minimize false positives to avoid hurting good users.

Figure 11 shows the ROC[9] curves of the four algorithms on the dataset of professional workers. Again, Random Forests perform best: they achieve extremely low false positive rate of $<$0.1% with only 8% false negative rate, or $<$0.1% false negative rate with only 7% false positive rate. We note that SVMs provide better false positive and false negative tradeoffs than J48, even though they have similar accuracy rates.

Imbalanced Data. We check our results on imbalanced data, since in practice there are more normal users than crowdturf workers. More specifically, we run our classifier (RF, professional) on imbalanced *testing* data with turfing-to-normal ratio ranging from 0.1 to 1. Note that we can still train our classifiers on balanced *training* data since we use supervised learning (we make sure training and testing data have no overlap). We find all the classifiers have accuracy above 98% (maximum FP 1.5%, FN 1.3%) against imbalanced testing data. We omit the plot for brevity.

Summary. Our results show that current ML systems can be used to effectively detect crowdturf workers. While this is a positive result, it assumes no adversarial response from the crowdturfing system. The following sections will examine detection efficacy under different levels of adversarial attacks.

5 Adversarial Attack: Evasion

We show that ML detectors can effectively identify "passive" crowdturf accounts in Weibo. In practice, however, crowdturfing adversaries can be highly adaptive: they will change their behaviors over time or can even intentionally attack the ML detectors to escape detection. We now re-evaluate the robustness of ML detectors under different adversarial environments, focusing on two types of adversaries:

1. *Evasion Attack:* individual crowd-workers adjust their behavior patterns to evade detection by trained ML detectors.
2. *Poisoning Attack:* administrators of crowdturfing sites participate, manipulating the ML detector training process by poisoning the training data.

We focus on evasion attacks in this section, and delay the study of poisoning attacks to §6. First, we define the evasion attack model. We then implement evasion attacks of different strengths, and study the performance

[9]ROC (receiver operating characteristic) is a plot that illustrates classifier's false positives and true positives versus detection threshold.

of ML detectors accordingly. Specifically, we consider "optimal evasion" attacks, where adversaries have full knowledge about the ML detectors and the Weibo system, and more "practical" evasion attacks, where adversaries have limited knowledge about the detectors and the Weibo system.

5.1 Basic Evasion Attack Model

Evasion attacks refer to individual crowdturfing workers seeking to escape detection by altering their own behavior to mimic normal users. For example, given knowledge of a deployed machine learning classifier, a worker may attempt to evade detection by selecting a subset of user features, and replacing their values with the *median* of the observed normal user values. Since mimicking normal users reduces crowdturfing efficiency, workers are motivated to minimize the number of features they modify. This means they need to identify a minimal core set of features enabling their detection.[10]

This attack makes two assumptions. *First*, it assumes that adversaries, *i.e.* workers, know the list of features used by the classifiers. Technical publications, *e.g.* on spam detection [4, 43, 50], make it possible for adversaries to make reasonable guesses on the feature space. *Second*, it assumes that adversaries understand the characteristics of normal users in terms of these features. In practice, this knowledge can be obtained by crawling a significant portion of Weibo accounts.

Depending on their knowledge of the ML features and of normal user behavior, adversaries can launch evasion attacks of different strengths. We implement and evaluate ML models on a range of threat models with varying levels of adversary knowledge and computational capabilities. We start from the *optimal evasion* scenario, where adversaries have *complete* knowledge of the feature set. The corresponding ML detector results represent worst-case performance (or lower bound) under evasion attacks. We also study a set of *practical evasion* models where adversaries have limited (and often noisy) knowledge, and constrained resources.

5.2 Optimal Evasion Attack

In this *ideal* case, adversaries have perfect knowledge about the set of features they need to modify. To understand the impact of the feature choices, we look at multiple variants of the optimal evasion models. These include the *per-worker optimal evasion model*, where each worker finds her own optimal set of features to alter, the *global optimal evasion* where all workers follow the same optimal set of features to alter, and *feature-aware evasion* where workers alter the most important features. We implement these evasion models on our ground-truth dataset, and evaluate ML detector accuracy. Note that these attacks we identify are not necessarily practical, but are designed to explore worse-case scenarios for ML models.

Per-worker Optimal Evasion. Intuitively, each worker should have her own optimal strategy to alter features, *e.g.* some workers need to add followers first, while others need to reduce tweeting burstiness. Doing so is hard in practice: each worker has to apply exhaustive search to identify its optimal strategy that minimizes the set of features to modify.

We implement this scenario on our *Active+Turfing* dataset. We first split the data into equal-sized training and testing datasets, and use the top-4 most accurate algorithms to build classifiers with authentic training data. We then run detection on worker accounts in the testing dataset. Here for each worker, we exhaustively test all combinatorial combinations of possible features to modify until the classifier classifies this worker as normal. In this way, we find the minimal set of features each user must modify to avoid detection.

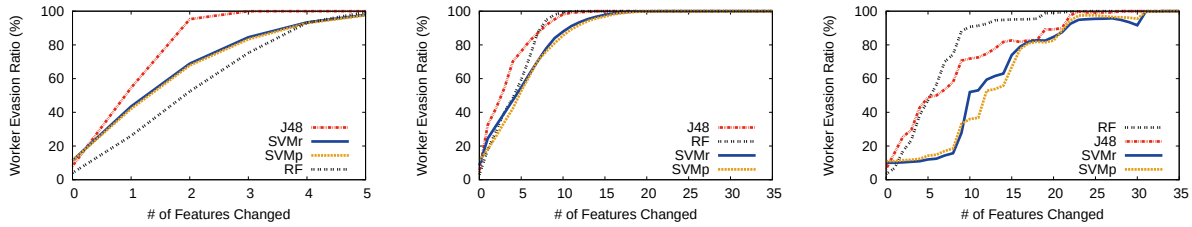
Figure 12(a) plots the evasion rate for the four ML algorithms. Clearly, this optimal evasion model is highly effective. By simply altering one feature, 20-50% of workers can evade detection (different workers can choose to alter different features). And by altering five features, 99% of workers can evade all four classifiers. We also observe that the Random Forests (RF) algorithm achieves the best robustness, since it requires the most number of features to be altered.

Global Optimal Evasion. The per-worker model makes a strong assumption that each worker can identify her own optimal feature set. Next, we loosen this assumption and only assume that all workers exercise a uniform strategy. This is possible if a third-party (*e.g.* site admin) guides workers in altering their features.

To identify the global optimal strategy, we search exhaustively through all possible feature combinations, and locate the feature set (for a given size) that allows the majority of workers to achieve evasion. The corresponding evasion rate result is in Figure 12(b). 99% of workers can successfully evade all four detectors by altering 15 features, which is much larger than the per-worker case (5 features). This is because any one-size-fits-all strategy is unlikely to be ideal for individual workers, thus the feature set must be large enough to cover all workers.

Feature-aware Evasion. Achieving optimal evasion is difficult, since it requires adversaries to have knowledge of the trained classifiers. Instead, this model assumes that adversaries can accurately identify the relatively "importance" of the features. Thus workers alter the most important features to try to avoid detection.

[10] For simplicity, we consider features to be independent.



(a) Per-worker Optimal Evasion (b) Global Optimal Evasion (c) Feature Importance Aware Evasion

Figure 12: Evasion rate of optimal evasion strategies (all workers).

We implement this attack by building the classifiers and then computing the feature importance. For this we use the χ^2 (Chi Squared) statistic [51], a classic metric to measure feature's discriminative power in separating data instances of different classes[11]. During detection, workers alter features based on their rank.

Figure 12(c) plots evasion results for the four classifiers. We make two key observations. First, this feature-aware strategy is still far away from the per-worker optimal case (Figure 12(a)), mostly because it is trying to approximate global optimal evasion. Second, performance depends heavily on the underlying classifier. For RF and J48, performance is already very close to that of the global optimal case, while the two SVM algorithms are more resilient. A possible explanation is that the χ^2 statistic failed to catch the true feature importance for SVM, since SVM normalizes feature values before training the classifier. These results suggest that without knowing the specific ML algorithm used by the defenders, it is hard to avoid detection even knowing the importance of features.

5.3 Evasion under Practical Constraints

Our results show workers can evade detection given complete knowledge of the feature set and ML classifiers. However, obtaining complete knowledge is very difficult in practice. Thus we examine *practical* evasion threat models to understand their efficacy compared to optimal evasion models. We identify practical constraints facing adversaries, present several practical threat models and evaluate their impact on our detectors.

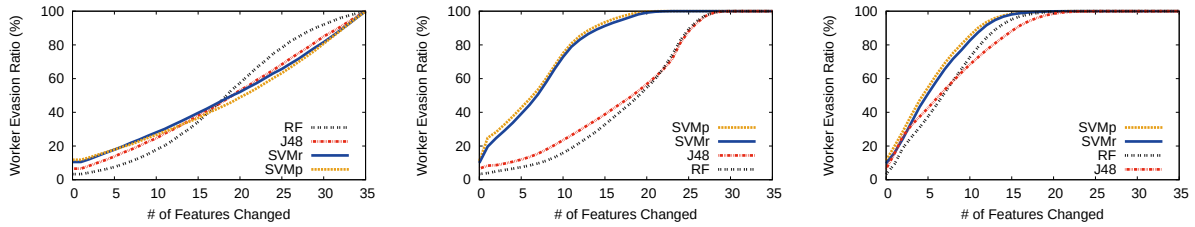
Practical Constraints. In practice, adversaries face two key resource constraints. First, they cannot reverse-engineer the trained classifier (*i.e.* the ML algorithm used or its model parameters) by querying the classifier and analyzing the output – it is too costly to establish millions of profiles with controlled features and wait for some of them to get banned. Thus workers cannot perform exhaustive search to launch optimal evasion attacks, but have to reply on their partial knowledge for evasion. Second, it is difficult for adversaries to obtain *complete* statistics of normal users. They can estimate normal user statistics via a (small) sampling of user profiles, but estimation errors are likely to reduce their ability to precisely mimic normal users.

Next, we will examine each constraint separately, and evaluate the likely effectiveness of attacks under the more realistic conditions.

Distance-aware Evasion. We consider the first constraint which forces workers to rely on *partial* knowledge to guide their evasion efforts. In this case, individual workers are only aware of their own accounts and normal user statistics. When choosing features to alter, they can prioritize features with the largest differential between them and normal users. They must quantify the "distance" between each crowdturf account and normal users on a given feature. Here, we pick two very intuitive distance metrics and examine the effectiveness of the corresponding evasion attacks. For now, we ignore the second constraint by assuming workers have perfect knowledge of average user behaviors.

- *Value Distance (VD):* Given a feature k, this captures the distance between a crowd-worker i and normal user statistics by $VD(i,k) = \frac{|F_k(i) - Median(N_k)|}{Max(N_k) - Min(N_k)}$ where $F_k(i)$ is the value of feature k in worker i, and N_k is normal user statistical distribution on feature k. When altering feature k, worker i replaces $F_k(i)$ with $Median(N_k)$.
- *Distribution Distance (DD):* Here the distance depends on where $F_k(i)$ is positioned within N_k. For example, if $F_k(i)$ is around 50%-tile of N_k, then worker i is similar to a normal user. Therefore, we define the distance by $DD(i,k) = |Percentile(N_k, F_k(i)) - 50|/100$ where $Percentile(N_k, F_k(i))$ is the percentile of $F_k(i)$ in the normal user CDF N_k. Note that when $F_k(i)$ exceeds the range of N_k, this distance metric becomes invalid. However, our data suggests that this rarely happens (<1%).

[11] We also tested information gain to rank features, which produced similar ranking results (*i.e.* the same top-10 as using χ^2).



(a) Random Evasion Strategy (Random) (b) Value Distance Aware Strategy (VD) (c) Distribution Distance Aware Strategy (DD)

Figure 13: Evasion rate of practical evasion strategies (all workers).

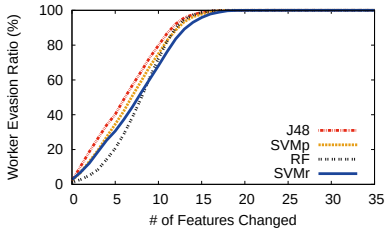


Figure 14: Evasion rate using distribution distance aware strategy (DD) for professional workers.

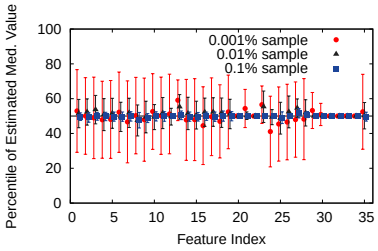


Figure 15: The percentile of *estimated* median value in the true normal user CDF.

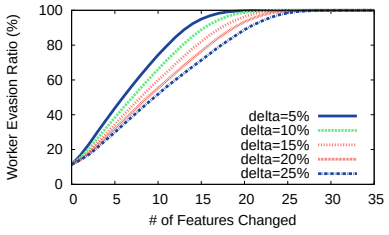


Figure 16: Impact of median value estimation error on evasion rate, using DD evasion on SVMp.

To evaluate the impact of practical evasion attacks, we split the *Active+Turfing* data into equal-sized training and testing sets. After classifier training, we simulate the distance-aware evasion attacks on the testing data. Figure 13(b) and 13(c) show evasion rates based on VD and DD respectively. As a baseline, we also show Figure 13(a) where adversaries *randomly* select features to alter. Compared to random evasion, distance-based approaches require much less feature altering. For example, when altering 15 features, random approach only saves $<$40% of workers, while distance strategies provide as high as 91% (VD-SVMp) and 98% (DD-SVMp).

The four classifiers perform very differently. RF and J48 classifiers are much more vulnerable to DD based evasion than to VD based evasion. While SVMs perform similarly in both strategies. In general, Tree-based algorithms are more robust than SVM classifiers against distance-aware evasions. This is very different to what we observed in the optimal evasion cases (Figure 12(a)–12(b)), where SVMs are generally more robust. This suggests that theoretical bounds on ML algorithms may not truly reflect their performance in practice.

Consistently, the impact of practical evasion attacks is much weaker than that of optimal evasion (*i.e.* per-worker optimal). Adversaries are severely constrained by lack of knowledge of detection boundaries of the classifiers, and have to guess based on "distance" information. The implication is that the less adversaries know about classifiers, the harder it is for them to evade detection.

We also evaluate the attack impact on classifiers to detect professional workers. We find the general trends are similar and only show the results of DD-based attack in Figure 14. We note that it is easier to evade classifiers dedicated to detect professionals (compared with Figure 13(c)). This is because when trained to a smaller, more homogeneous worker population, classifiers expect strong malicious behaviors from crowd-workers. Thus even a small deviation away from the model towards normal users will help achieve evasion.

Impact of Normal User Estimation Errors. We extend the above model by accounting for possible errors in estimating normal user behaviors (the second constraint). These errors exist because adversaries can only sample a limited number of users, leading to noisy estimations. Here, we investigate the impact of sampling strategies on the attack efficacy.

For all 35 features, we vary the sampling rate, *i.e.* the ratio of normal users sampled by adversaries, by taking random samples of 0.001%, 0.01% to 0.1% of the *Active* dataset. We repeat each instance 100 times, and compute the mean and standard deviation of the estimated median feature values (Figure 15). We show each feature's *percentile* in the true CDF of the *Active* dataset. In this case, the optimal value is 50%. Clearly sampling rate does impact feature estimation. With the 0.001% sampling rate, the estimated value varies significantly across instances. Raising the sample rate to 0.1% means attackers can accurately estimate the median value using only

a few instances. Furthermore, we see that burstiness features (*e.g.* features 30-34) are easy to sample, since normal user values are highly skewed to zero.

Finally, we evaluate the impact of estimation errors on practical evasion attacks. This time we run distance-aware evasions based on the *estimated* median feature values. For each worker's feature k, we estimate the median value $M'(k)$ with a given bound of error Δ. That is, $M'(k)$ is randomly picked from the percentiles within $[50\% - \Delta, 50\% + \Delta]$ on the true CDF of normal user behaviors. By iterating through different Δ (from 5% to 25%), our results show that Δ only has a minor impact. The most noticeable impact is on SVMp using DD distance (Figure 16). Overall, we conclude that as long as adversaries can get a decent guess on normal user behaviors, the residual noise in the estimation Δ should not affect the efficacy of evasion attacks.

Summary. Our work produces two key observations.

- Given complete knowledge, evasion attacks are very effective. However, adversaries under more realistic constraints are significantly less effective.
- While no classifier is robust against all attack scenarios, there is a consistent inverse relationship between single model fitting accuracy and robustness to adversarial evasion. Highly accurate fit to a smaller, more homogeneous population (*e.g.* professionals) makes models more vulnerable to evasion attacks.

6 Adversarial Attack: Poisoning

After examining evasion attacks, we now look at how centralized crowdturfing sites can launch more powerful attacks to manipulate machine learning models. Specifically, we consider the poisoning attack where administrators of crowdturfing sites intentionally pollute the training dataset used to build ML classifiers, forcing defenders to produce inaccurate classifiers. Since the training data (*i.e.* crowdturfing accounts) actually comes from these crowdturfing sites, administrators are indeed capable of launching these attacks.

In the following, we examine the impact of poisoning attacks on ML detection accuracy. We consider two mechanisms for polluting training data. The first method directly adds misleading/synthetic samples to the training set. Adversaries in the second method simply alter data records, or modify operational policies to alter the composition of the training data used by ML models.

6.1 Injecting Misleading Samples

Perhaps the simplest way to pollute any training data is to add misleading or false samples. In our case, since the training data has two classes (groups) of accounts, this can be done by mixing normal user samples into the "turfing" class, *i.e.* poisoning the turfing class, or mixing crowdturf samples into the "normal" user class, *i.e.* poisoning the normal class. Both introduce incorrectly labeled training data to mislead the classifier.

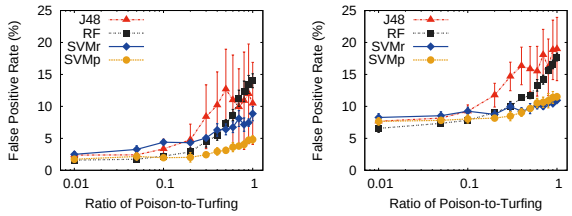
Poisoning Turfing Class. To poison the turfing class, adversaries (*e.g.* ZBJ and SDH administrators) add normal Weibo accounts to the public submission records in their own systems. Since ML classifiers take ground-truth crowdturf accounts from those public records, these benign accounts will then be mixed into the training data and labeled as "turfing." The result is a model that marks some characteristics of normal users as crowdturfing behavior, likely increasing false positive rate in detection.

We simulate the attack with our ground-truth dataset. At a high level, we train the classifiers on "polluted" training data, and then examine changes in classifiers' detection accuracy. Here we experiment with two strategies to pollute the turfing class. First, as a baseline strategy, adversaries *randomly* select normal users as poison samples to inject into the turfing class. Second, adversaries can inject *specific* types of normal users, causing the classifiers to produce *targeted* mistakes.

Random Poisoning: We simulate this poisoning attack with *Active+Turfing* dataset, where adversaries inject random normal accounts to the turfing class. Specifically, for training, the turfing class (14K accounts) is a mixture of crowdturf accounts and poison samples randomly selected from *Active*, with a mixing ratio of p. The normal class is another 14K normal accounts from *Active*. Then we use 28K of the rest accounts (14K turfing and 14K normal users) for testing. For any given p, we repeat the experiment 10 times with different random poison samples and training-testing partitions to compute average detection rates.

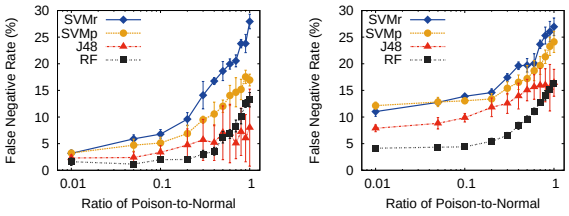
Results are shown in Figure 17(b). As a baseline comparison, we also present the results of the classifiers for professional workers in Figure 17(a). We have three observations. First, as poison-to-turfing ratio p increases, false positive rates go up for all four algorithms. False negative rates are not much affected by this attack, thus are omitted from the plot.[12] Second, we find that the SVM classifiers are more resilient: SVMp's false positive rate increases $<5\%$ as p approaching 1.0, while the analogous increases exceed 10% for Random Forests and J48. Particularly, J48 experiences more drastic fluctuations around average, indicating it is very sensitive to the choice of poison samples. This is consistent with our prior observation that more accurate single model fitting (*i.e.* J48 is more accurate than SVM) is more vulnerable to adversarial attacks. Similarly, highly accurate detec-

[12] False negative rates increase $< 2\%$ when p approaches 1.0.



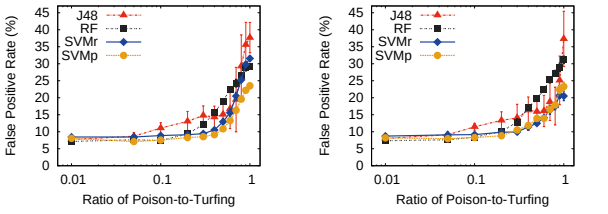
(a) Professional Workers (b) All Workers

Figure 17: Poisoning training dataset by injecting random normal user samples to the turfing class.



(a) Professional Workers (b) All Workers

Figure 19: Poisoning training dataset by adding turfing samples to normal class.



(a) Injecting Accounts with $>50\%$ tweets commented (b) Injecting Accounts with <150 followers

Figure 18: Targeted poisoning. Adversaries inject specific type of normal users to the turfing class (all workers).

tion of the more homogeneous population of professional workers (§4) means the models experience larger relative impacts from attacks compared to classifiers over all workers.

Note that we limited the poison-to-turfing ratio <1, since in practice adversaries cannot inject unlimited poison samples to defender's training data. First, injecting noise causes inconvenience to their own customers in locating qualified workers. Second, defenders may collect ground-truth records from multiple crowdturfing sites.

Targeted Poisoning: Next, we explore *targeted* poisoning to the turfing class. Here the adversaries want to carefully inject selected poison samples so classifiers make targeted mistakes. For example, our classifier uses "ratio of commented tweets" as a feature with the intuition that worker's tweets rarely receive comments (§3). Once adversaries gain this knowledge, they can intentionally select accounts whose tweets often receive comments as the poison samples. As a result, the trained classifier will mistakenly learn that users with high comment ratio can be malicious, thus are likely to misclassify this kind of normal users as crowd-workers.

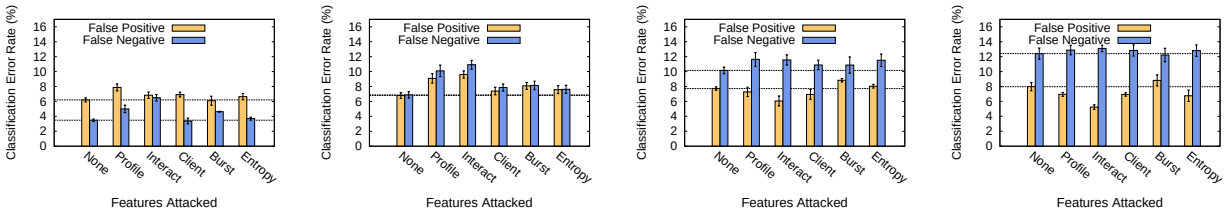
To evaluate the impact of targeted poisoning, we perform similar experiments, except that we select poison samples based on specific feature. Figure 18 shows the attacking results on two example features: ratio of tweets with comments and follower count. Compared with Figure 17, targeted poisoning can trigger higher false positives than randomly selecting poison samples. Also, the previous observations still hold with SVM being more robust and J48 experiencing unstable performance (large deviation from average).

Poisoning Normal User Class. Next, we analyze the other direction where adversaries inject turfing samples into the "normal" class to boost the *false negative rate* of classifiers. This may be challenging in practice since the normal user pool – Weibo's whole user population – is extremely large. Hence it requires injecting a significant amount of misleading samples in order to make an impact. Here from defender's perspective, we aim to understand how well different classifiers cope with "noisy" normal user data.

We repeat the previous "Random Poisoning" attack on the normal class. Figure 19(a) and Figure 19(b) show the attack results on classifiers for professional workers and all workers respectively. As we increase the ratio of poison samples, the false negatives of all four classifiers increase. This is expected as the classifiers will mistakenly learn crowdturf characteristics when modeling normal users, thus are likely to misclassify turfing accounts as benign later. In addition, we find the robustness of different classifiers varies, with Random Forests algorithm producing the lowest overall false negatives. Finally, we again observe that the more accurate classifier for professional workers suffers larger relative impacts from adversaries than classifiers for all-workers.

6.2 Altering Training Data

The above poisoning attacks focus on misleading classifiers to catch the wrong target. However, it does not fundamentally prevent crowd-workers from detection, since workers' behavior patterns are still very differently from normal users. To this end, we explore a second poisoning attack, where adversaries directly *alter* the training data by tuning crowd-workers' behavior to mimic normal users. The goal is to make it difficult (or even impossible) to train an accurate classifier that isolates crowdturf accounts with normal accounts.



(a) Random Forests (b) J48 (c) SVMr (d) SVMp

Figure 20: Performance of different classifiers when adversaries alter crowd-workers' features to mimic normal users. The horizontal lines represent the baseline false positive (false negative) rates when no feature is altered.

To carry out this attack, adversaries (*e.g.* administrators of ZBJ and SDH) need to modify the behaviors of numerous crowdturf workers. This can be done by centrally enforcing operational policies to their own system. For example, enforcing minimal time interval between taking tasks to reduce the tweeting burstiness or enforcing screening mechanisms to reject worker accounts with "malicious" profile features. In the following, we evaluate the attack impact using simulations, followed by the discussion of practical costs.

Feature Altering Attack. To simulate this attack, we let adversaries select a set of features F of crowdturf accounts and alter F to mimic the corresponding features of normal users. Unlike evasion attacks that can simply mimic normal users' median values, here we need to mimic the whole distribution in order to make the two classes indistinguishable on these features. Since the feature altering is for all workers in the crowdturfing system, thus it actually applies to crowdturf accounts in both training and testing datasets. Finally, note that features are not completely independent, *i.e.* changing one feature may cause changes in others. To mitigate this, we tune features under the same category simultaneously.

Figure 20 shows the attack results on *Turfing+Active* dataset. We attack each feature category and repeat the experiment for 10 times. Here we simulate attacking one category at a time, and will discuss attacking category combinations later. In general, the attack makes all classifiers produce higher error rates compared with baseline where no feature is altered (the horizontal lines). However the impact is mild compared to injection-based poisoning attacks. For example, the most effective attack is on J48 when altering interaction features, which causes error rate increased by 4%, while injection-based attack can boost error rate by more than 20% (Figure 18). One possible reason is that unlike injection-based poisoning, altering-based poisoning does not cause inconsistencies in training and testing data, but only make the two classes harder to separate.

Costs of Altering. In practice, feature altering comes with costs, and some features may be impossible to manipulate even by crowdturfing administrators. For instance, *Tweeting Regularity* (Entropy) and *Burstiness* features are easier to alter. Recall that crowdturfing systems can enforce minimal (random) time delay between workers taking on new tasks, or use delays to increase entropy. Changing the *Tweet Client* feature is also possible, since crowdturfing systems can develop mobile client software for their workers, or simply release tools for workers to fake their tweeting clients.

Profile and *Interaction* features are more difficult to alter. Some features are mandatory for common tasks. For example, workers need to maintain a certain number of followers in order to spread spam to reach large enough audiences. In addition, some features are rooted in the fact that crowd-workers don't use their accounts organically, which, making it hard to generate normal user interactions. Although, crowdturfing systems could potentially use screening mechanisms to reject obviously-malicious crowdturf accounts from their system. However, this high bar will greatly shrink the potential worker population, and likely harm the system's spam capacity.

Considering practical costs, we consider whether it is more impactful to alter the combinations of features from different categories. Here we focus on altering the low cost features in *Tweet Client* (C), *Burstiness* (B) and *Entropy* (E). As shown in Table 4, attacking feature combinations produces slightly higher error rates than attacking a single feature category, but the overall effect is still small (less than 4% error rate increase).

Features Attacked	Error Rate (FP %, FN %)			
	RF	**J48**	**SVMr**	**SVMp**
None	(6.2, 3.4)	(6.7, 6.8)	(7.7, 10.1)	(7.9, 12.1)
C+B	(5.7, 4.4)	(7.9, 8.7)	(8.7, 12.2)	(8.0, 14.0)
B+E	(6.5, 3.9)	(7.1, 7.8)	(8.7, 12.5)	(7.3, 13.1)
C+E	(6.4, 4.5)	(7.9, 8.2)	(7.5, 11.8)	(6.3, 13.8)
C+B+E	(5.8, 4.2)	(8.3, 8.5)	(8.6, 13.2)	(7.7, 15.2)

Table 4: Error rates when features are altered in combinations. We focus on attacking low-cost features: Tweet Client (C), Burstiness (B) and Entropy (E).

Summary and Discussion. Through our analysis, we find that injecting misleading samples into training data causes more significant errors than uniformly altering worker behavior. In addition, we again observe the inverse relationship between single model fitting accuracy and robustness.

To protect their workers, crowdturfing sites may also try to apply stronger access control to their public records in order to make training data unavailable for ML detectors[13]. However, this creates obvious inconvenience for crowdturfing sites, since they rely on these records to attract new workers. Moreover, even if records were private, defenders can still obtain training data by joining as "customers," offering tasks, and identifying accounts of participating workers.

7 Related Work

Crowdturfing. Prior works used measurements on crowdturfing sites to understand their operation and economic structure [23, 24, 26, 48]. Some systems have been developed to detect paid human spammers in online review sites [31] and Q&A systems [9, 45]. To the best of our knowledge, our work is the first to explore detection of crowdturfing behaviors in adversarial settings.

OSN Spammer Detection. Researchers have developed mechanisms to detect fake accounts (Sybil) and spam campaigns in online social networks, including Facebook [15, 49], Twitter [43], Renren [52] and LinkedIn [46]. Most prior works develop ML models using features of spammer profiles (*e.g.* FFRatio, blacklisted URLs) or bot-like behaviors [4, 11, 42, 47, 50]. However, a recent study shows dedicated spam bots can still infiltrate social networks without being detected [14]. In our case, crowdturf accounts are carefully maintained by human users, and their questionable activities are camouflaged with synthetic cover traffic. This makes their detection challenging, until we add additional behavioral features (*e.g.* user-interaction, task-driven behavior).

Adversarial Machine Learning. In an early study [19], researchers classify ML adversarial attacks into two high-level categories: *causative* attacks where adversaries alter the training process to damage the classifier performance, and *exploratory* attacks where adversaries try to circumvent an already-trained classifier. Much of existing work focuses on *exploratory* attacks [5, 12, 25, 28] with less focusing on *causative* attacks [6, 37], since it's usually more difficult for adversaries to access training data in practice. In this paper, we studied both angles as both attacks are practically feasible from crowdturfing adversaries.

Several studies have examined attacks on specific ML-based applications, from email spam detection [12] to network intuition detection [37, 40, 44] to malicious (PDF) file classification [5, 25, 41] and malware detection [21]. Our work focuses on crowdturfing and explores a wider range of adversarial attacks, including active evasion and more powerful poison attacks against the model training process.

8 Conclusion and Discussion

We use a large-scale ground truth dataset to develop machine learning models to detect malicious crowdsourcing workers. We show that while crowdturfing workers resemble normal users in their profiles, ML models can effectively detect regular workers (95% accuracy) or "professionals" (99% accuracy) using distinguishing features such as user interactions and tweet dynamics.

More importantly, we use crowdturfing defense as context to explore the robustness of ML algorithms against adversarial attacks. We evaluate multiple adversarial attack models targeting both training and testing phases of ML detectors. We find that these attacks are effective against all machine learning algorithms, and coordinated attacks (such as those possible in crowdturfing sites) are particularly effective. We also note a consistent tradeoff where more accurate fits (especially to a smaller, more homogeneous population) result in higher vulnerability to adversarial attacks. The exception appears to be Random Forests, which often achieves both high accuracy and robustness to adversaries, possibly due to its natural support for multiple populations.

Limitations and Future Work. We note that our study has several limitations. First, our analysis focuses on Weibo, and our adversary scenarios may not generalize fully to other platforms (*e.g.* review sites, instant message networks). However, more work is necessary to validate our findings on other platforms. Second, our adversarial models use simplifying assumptions, *i.e.* features are independent and costs for feature modification are uniform. In addition, attackers may behave differently to disrupt the operation of ML detectors.

Moving forward, one goal is to validate our adversarial models in practice, perhaps by carrying out a user-study on crowdturfing sites where we ask workers to actively evade and disrupt ML detectors. In addition, our results show we must explore approaches to improve the robustness of ML-based systems. Our analysis showed that ML algorithms react differently to different adversarial attacks. Thus one possible direction is to develop hybrid systems that integrate input from multiple classi-

[13] As of late 2013, some crowdturfing sites (*e.g.* ZBJ) have already started to follow this direction, by limiting access to public transaction records to verified active participants.

fiers, ideally without affecting overall accuracy. We also observe that limiting adversaries' knowledge of the target system can greatly reduce their attack abilities. How to effectively prevent adversaries from gaining knowledge or reverse-engineering models is also a topic for future work.

9 Acknowledgments

We would like to thank the anonymous reviewers for their helpful feedback, and Xifeng Yan for insightful discussions. This work is supported in part by NSF grants IIS-1321083, CNS-1224100, IIS-0916307, by the DARPA GRAPHS program (BAA-12-01), and by the Department of State. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] Sina Weibo Admin Statement on Spam. `http://www.weibo.com/p/1001603697836242954625`, April 2014.

[2] Sina Weibo Terms of Service. `http://service.account.weibo.com/roles/guiding`, 2014. (The link is accessible after login).

[3] AHN, Y.-Y., HAN, S., KWAK, H., MOON, S., AND JEONG, H. Analysis of topological characteristics of huge online social networking services. In *Proc. of WWW* (2007).

[4] BENEVENUTO, F., MAGNO, G., RODRIGUES, T., AND ALMEIDA, V. Detecting spammers on twitter. In *Proc. of CEAS* (2010).

[5] BIGGIO, B., CORONA, I., MAIORCA, D., NELSON, B., SRNDIC, N., LASKOV, P., GIACINTO, G., AND ROLI, F. Evasion attacks against machine learning at test time. In *Proc. of ECML PKDD* (2013).

[6] BIGGIO, B., NELSON, B., AND LASKOV, P. Poisoning attacks against support vector machines. In *Proc. of ICML* (2012).

[7] BREIMAN, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[8] CARUANA, R., KARAMPATZIAKIS, N., AND YESSENALINA, A. An empirical evaluation of supervised learning in high dimensions. In *Proc. of ICML* (2008).

[9] CHEN, C., WU, K., SRINIVASAN, V., AND R, K. B. The best answers? think twice: Online detection of commercial campaigns in the cqa forums. *CoRR* (2012).

[10] CHEN, X. Dairy giant mengniu in smear scandal. China Daily, October 2010.

[11] CHU, Z., GIANVECCHIO, S., WANG, H., AND JAJODIA, S. Who is tweeting on twitter: human, bot, or cyborg? In *Proc. of ACSAC* (2010).

[12] DALVI, N., DOMINGOS, P., MAUSAM, SANGHAI, S., AND VERMA, D. Adversarial classification. In *Proc. of KDD* (2004).

[13] EATON, K. Mechanical turk's unsavory side effect: Massive spam generation. Fast Company, December 2010.

[14] FREITAS, C. A., BENEVENUTO, F., GHOSH, S., AND VELOSO, A. Reverse engineering socialbot infiltration strategies in twitter. *CoRR abs/1405.4927* (2014).

[15] GAO, H., HU, J., WILSON, C., LI, Z., CHEN, Y., AND ZHAO, B. Y. Detecting and characterizing social spam campaigns. In *Proc. of IMC* (2010).

[16] GIANVECCHIO, S., AND WANG, H. Detecting covert timing channels: an entropy-based approach. In *Proc. of CCS* (2007).

[17] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. The weka data mining software: an update. *SIGKDD Explor. Newsl. 11*, 1 (2009).

[18] HECKERMAN, D., GEIGER, D., AND CHICKERING, D. M. Learning bayesian networks: The combination of knowledge and statistical data. *Mach. Learn. 20*, 3 (1995), 197–243.

[19] HUANG, L., JOSEPH, A. D., NELSON, B., RUBINSTEIN, B. I., AND TYGAR, J. D. Adversarial machine learning. In *Proc. of AISec* (2011).

[20] JOHN, G. H., AND LANGLEY, P. Estimating continuous distributions in bayesian classifiers. In *Proc. of UAI* (1995).

[21] KANTCHELIAN, A., AFROZ, S., HUANG, L., ISLAM, A. C., MILLER, B., TSCHANTZ, M. C., GREENSTADT, R., JOSEPH, A. D., AND TYGAR, J. D. Approaches to adversarial drift. In *Proc. of AISec* (2013).

[22] LAKHINA, A., CROVELLA, M., AND DIOT, C. Diagnosing network-wide traffic anomalies. In *Proc. of SIGCOMM* (2004).

[23] LEE, K., TAMILARASAN, P., AND CAVERLEE, J. Crowdturfers, campaigns, and social media: Tracking and revealing crowdsourced manipulation of social media. In *Proc. of ICWSM* (2013).

[24] LEE, K., WEBB, S., AND GE, H. The dark side of micro-task marketplaces: Characterizing fiverr and automatically detecting crowdturfing. In *Proc. of ICWSM* (2014).

[25] MAIORCA, D., CORONA, I., AND GIACINTO, G. Looking at the bag is not enough to find the bomb: an evasion of structural methods for malicious pdf files detection. In *Proc. of ASIACCS* (2013).

[26] MOTOYAMA, M., MCCOY, D., LEVCHENKO, K., SAVAGE, S., AND VOELKER, G. M. Dirty jobs: The role of freelance labor in web service abuse. In *Proc. of Usenix Security* (2011).

[27] NELSON, B., RUBINSTEIN, B. I. P., HUANG, L., JOSEPH, A. D., HON LAU, S., LEE, S. J., RAO, S., TRAN, A., AND TYGAR, J. D. Near-optimal evasion of convex-inducing classifiers. In *Proc. of AISTATS* (2010).

[28] NELSON, B., RUBINSTEIN, B. I. P., HUANG, L., JOSEPH, A. D., LEE, S. J., RAO, S., AND TYGAR, J. D. Query strategies for evading convex-inducing classifiers. *J. Mach. Learn. Res. 13*, 1 (2012).

[29] NEWSOME, J., KARP, B., AND SONG, D. Polygraph: Automatically generating signatures for polymorphic worms. In *Proc. of IEEE S&P* (2005).

[30] ONG, J. China's sina weibo grew 73% in 2012, passing 500 million registered accounts. The Next Web, Feb. 2013.

[31] OTT, M., CHOI, Y., CARDIE, C., AND HANCOCK, J. T. Finding deceptive opinion spam by any stretch of the imagination. In *Proc. of ACL* (2011).

[32] PHAM, N. Vietnam admits deploying bloggers to support government. BBC News, January 2013.

[33] PLATT, J. C. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning* (1998).

[34] QUINLAN, J. R. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[35] RAMACHANDRAN, A., FEAMSTER, N., AND VEMPALA, S. Filtering spam with behavioral blacklisting. In *Proc. of CCS* (2007).

[36] ROBINSON, G. A statistical approach to the spam problem. *Linux J. 2003*, 107 (2003).

[37] RUBINSTEIN, B. I., NELSON, B., HUANG, L., JOSEPH, A. D., LAU, S.-H., RAO, S., TAFT, N., AND TYGAR, J. D. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proc. of IMC* (2009).

[38] SHEAR, M. D. Republicans use crowdsourcing to attack obama campaign. The New York Times, May 2012.

[39] SIMONITE, T. Hidden industry dupes social media users. MIT Review, December 2011.

[40] SOMMER, R., AND PAXSON, V. Outside the closed world: On using machine learning for network intrusion detection. In *Proc. of IEEE S&P* (2010).

[41] SRNDIC, N., AND LASKOV, P. Practical evasion of a learning-based classifier: A case study. In *Proc. of IEEE S&P* (2014).

[42] STRINGHINI, G., KRUEGEL, C., AND VIGNA, G. Detecting spammers on social networks. In *Proc. of ACSAC* (2010).

[43] THOMASY, K., GRIERY, C., PAXSONY, V., AND SONGY, D. Suspended accounts in retrospect: An analysis of twitter spam. In *Proc. of IMC* (2011).

[44] VENKATARAMAN, S., BLUM, A., AND SONG, D. Limits of learning-based signature generation with adversaries. In *Proc. of NDSS* (2008).

[45] WANG, G., GILL, K., MOHANLAL, M., ZHENG, H., AND ZHAO, B. Y. Wisdom in the social crowd: an analysis of quora. In *Proc. of WWW* (2012).

[46] WANG, G., KONOLIGE, T., WILSON, C., WANG, X., ZHENG, H., AND ZHAO, B. Y. You are how you click: Clickstream analysis for sybil detection. In *Proc. of USENIX Security* (2013).

[47] WANG, G., MOHANLAL, M., WILSON, C., WANG, X., METZGER, M., ZHENG, H., AND ZHAO, B. Y. Social turing tests: Crowdsourcing sybil detection. In *Proc. of NDSS* (2013).

[48] WANG, G., WILSON, C., ZHAO, X., ZHU, Y., MOHANLAL, M., ZHENG, H., AND ZHAO, B. Y. Serf and turf: crowdturfing for fun and profit. In *Proc. of WWW* (2012).

[49] WILSON, C., SALA, A., PUTTASWAMY, K. P. N., AND ZHAO, B. Y. Beyond social graphs: User interactions in online social networks and their implications. *ACM Transactions on the Web 6*, 4 (November 2012).

[50] YANG, C., HARKREADER, R. C., AND GU, G. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers. In *Proc. of RAID* (2011).

[51] YANG, Y., AND PEDERSEN, J. O. A comparative study on feature selection in text categorization. In *Proc. of ICML* (1997).

[52] YANG, Z., WILSON, C., WANG, X., GAO, T., ZHAO, B. Y., AND DAI, Y. Uncovering social network sybils in the wild. In *IMC* (2011).

[53] ZHANG, L., YANG, J., AND TSENG, B. Online modeling of proactive moderation system for auction fraud detection. In *Proc. of WWW* (2012).

[54] ZHANG, Y., GE, Z., GREENBERG, A., AND ROUGHAN, M. Network anomography. In *Proc. of IMC* (2005).