You are How You Click: Clickstream Analysis for Sybil Detection

Gang Wang, Tristan Konolige, Christo Wilson[†], Xiao Wang[‡],
Haitao Zheng and Ben Y. Zhao
UC Santa Barbara [†]*Northeastern University* [‡]*Renren Inc.*
{*gangw, tkonolige, htzheng, ravenben*}*@cs.ucsb.edu, cbw@ccs.neu.edu, xiao.wang@renren-inc.com*

Abstract

Fake identities and Sybil accounts are pervasive in today's online communities. They are responsible for a growing number of threats, including fake product reviews, malware and spam on social networks, and astroturf political campaigns. Unfortunately, studies show that existing tools such as CAPTCHAs and graph-based Sybil detectors have not proven to be effective defenses.

In this paper, we describe our work on building a practical system for detecting fake identities using server-side clickstream models. We develop a detection approach that groups "similar" user clickstreams into behavioral clusters, by partitioning a similarity graph that captures distances between clickstream sequences. We validate our clickstream models using ground-truth traces of 16,000 real and Sybil users from Renren, a large Chinese social network with 220M users. We propose a practical detection system based on these models, and show that it provides very high detection accuracy on our clickstream traces. Finally, we worked with collaborators at Renren and LinkedIn to test our prototype on their server-side data. Following positive results, both companies have expressed strong interest in further experimentation and possible internal deployment.

1 Introduction

It is easier than ever to create fake identities and user accounts in today's online communities. Despite increasing efforts from providers, existing services cannot prevent malicious entities from creating large numbers of fake accounts or Sybils [9]. Current defense mechanisms are largely ineffective. Online Turing tests such as CAPTCHAs are routinely solved by dedicated workers for pennies per request [22], and even complex human-based tasks can be overcome by a growing community of malicious crowdsourcing services [23, 39]. The result of this trend is a dramatic rise in forged and malicious online content such as fake reviews on Yelp [35], malware and spam on social networks [10, 11, 32], and large, Sybil-based political lobbying efforts [27].

Recent work has explored a number of potential solutions to this problem. Most proposals focus on detecting Sybils in social networks by leveraging the assumption that Sybils will find it difficult to befriend real users. This forces Sybils to connect to each other and form strongly connected subgraphs [36] that can be detected using graph theoretic approaches [8, 34, 45, 46]. However, the efficacy of these approaches in practice is unclear. While some Sybil communities have been located in the Spanish Tuenti network [7], another study on the Chinese Renren network shows the large majority of Sybils actively and successfully integrating themselves into real user communities [43].

In this paper, we describe a new approach to Sybil detection rooted in the fundamental behavioral patterns that separate real and Sybil users. Specifically, we propose the use of *clickstream models* as a tool to detect fake identities in online services such as social networks. Clickstreams are traces of click-through events generated by online users during each web browsing "session," and have been used in the past to model web traffic and user browsing patterns [12, 20, 24, 28]. Intuitively, Sybils and real users have very different goals in their usage of online services: where real users likely partake of numerous features in the system, Sybils focus on specific actions (*i.e.* acquiring friends and disseminating spam) while trying to maximize utility per time spent. We hypothesize that these differences will manifest as significantly different (and distinctive) patterns in clickstreams, making them effective tools for "profiling" user behavior. In our context, we use these profiles to distinguish between real and Sybil users.

Our work focuses on building a practical model for accurate detection of Sybils in social networks. We develop several models that encode distinct event sequences and inter-event gaps in clickstreams. We build weighted

graphs of these sequences that capture pairwise "similarity distance" between clickstreams, and apply clustering to identify groups of user behavior patterns. We validate our models using ground-truth clickstream traces from 16,000 real and Sybil users from Renren, a large Chinese social network with 220M users. Using our methodology, we build a detection system that requires little or no knowledge of ground-truth. Finally, we validate the usability of our system by running initial prototypes on internal datasets at Renren and LinkedIn.

The key contributions of this paper are as follows:

- To the best of our knowledge, we are the first to analyze click patterns of Sybils and real users on social networks. By analyzing detailed clickstream logs from a large social network provider, we gain new insights on activity patterns of Sybils and normal users.
- We propose and evaluate several clickstream models to characterize user clicks patterns. Specially, we map clickstreams to a similarity graph, where clickstreams (vertices) are connected using weighted edges that capture pairwise similarity. We apply graph partitioning to identify clusters that represent specific click patterns. Experiments show that our model can efficiently distinguish between clickstreams of Sybil and normal users.
- We develop a practical Sybil detection system based on our clickstream model, requiring minimal input from the service provider. Experiments using ground-truth data show that our system generates $<1\%$ false positives and $<4\%$ false negatives.
- Working closely with industrial collaborators, we have deployed prototypes of our system at Renren and LinkedIn. Security teams at both companies have run our system on real user data and received very positive results. While corporate privacy policies limit the feedback visible to us, both companies have expressed strong interest in further experimentation and possible deployment of our system.

To the best of our knowledge, we are the first to study clickstream models as a way to detect fake accounts in online social networks. Moving forward, we believe clickstream models are a valuable tool that can complement existing techniques, by not only detecting well-disguised Sybil accounts, but also reducing the activity level of any remaining Sybils to that of normal users.

Roadmap. We begin in Section 2 by describing the problem context and our ground-truth dataset, followed by preliminary analysis results in Section 3. Next, in Section 4 we propose our clickstream models to effectively distinguish Sybil with normal users. Then in Section 5, we develop an incremental Sybil detector that can scale with today's large social networks. We then extend this detector in Section 6 by proposing an unsupervised Sybil detector, where only a minimal (and fixed) amount of ground-truth is needed. Finally, in Section 7, we describe experimental experience of testing our prototype code in real-world social networks (Renren and LinkedIn). We then discuss related work in Section 8 and conclude in Section 9.

Dataset	Users	Clicks	Date (2011)	Sessions
Sybil	9,994	1,008,031	Feb.28-Apr.30	113,595
Normal	5,998	5,856,941	Mar.31-Apr.30	467,179

Table 1: Clickstream dataset.

2 Background

In this section, we provide background for our study. First, we briefly introduce the Renren social network and the malicious Sybils that attack it. Second, we describe the key concepts of user clickstreams, as well as the ground-truth dataset we use in our study.

Renren. Renren is the oldest and largest Online Social Network (OSN) in China, with more than 220 million users [17]. Renren offers similar features and functionalities as Facebook: users maintain personal profiles and establish social connections with their friends. Renren users can update their status, write blogs, upload photos and video, and share URLs to content on and off Renren. When a user logs-in to Renren, the first page they see is a "news-feed" of their friends' recent activities.

Sybils. Like other popular OSNs, Renren is targeted by malicious parties looking to distribute spam and steal personal information. As in prior work, we refer to the fake accounts involved in these attacks as Sybils [43]. Our goal is to detect and deter these malicious Sybils; our goal is not to identify benign fakes, *e.g.* pseudonymous accounts used by people to preserve their privacy.

Prior studies show that attackers try to friend normal users using Sybil accounts [43]. On Renren, Sybils usually have complete, realistic profiles and use attractive profile pictures to entice normal users. It is challenging to identify these Sybils using existing techniques because their profiles are well maintained, and they integrate seamlessly into the social graph structure.

Clickstream Data. In this paper, we investigate the feasibility of using *clickstreams* to detect Sybils. A clickstream is the sequence of HTTP requests made by a user to a website. Most requests correspond to a user explicitly fetching a page by clicking a link, although some requests may be programmatically generated (*e.g.* XmlHttpRequest). In our work, we assume that a clickstream can be unambiguously attributed to a specific user account, *e.g.* by examining the HTTP request cookies.

Our study is based on detailed clickstreams for 9994

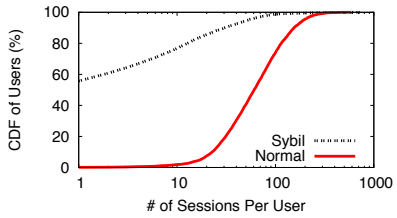


Figure 1: # of sessions per user.

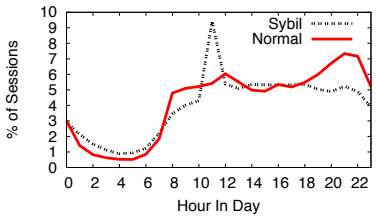


Figure 2: Sessions through the day.

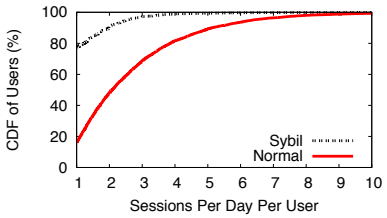


Figure 3: Sessions per day per user.

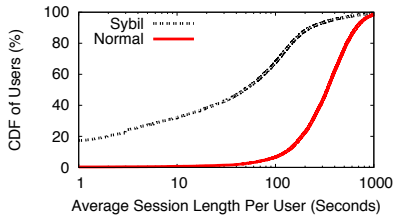


Figure 4: Average session length per user.

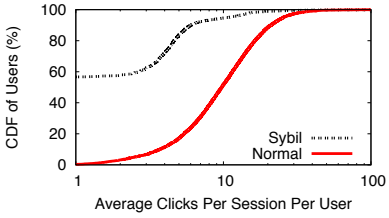


Figure 5: Average # of clicks per session per user.

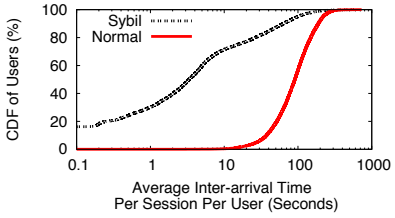


Figure 6: Average time interval between clicks per session per user.

Sybils and 5998 normal users on Renren. Sybil clickstreams were selected at random from the population of malicious accounts that were banned by Renren in March and April 2011. Accounts could be banned for abusive behaviors such as spamming, harvesting user data or sending massive numbers of friend requests. Normal user clickstreams were selected uniformly at random from Renren user population in April 2011, and were manually verified by Renren's security team.

The dataset summary is shown in Table 1. In total, our dataset includes 1,008,031 and 5,856,941 clicks for Sybils and normal users, respectively. Each click is characterized by a timestamp, an anonymized userID, and an *activity*. The activity is derived from the request URL, and describes the action the user is undertaking. For example, the "friend request" activity corresponds to a user sending a friend request to another user. We discuss the different *categories* of activities in detail in Section 3.2.

Each user's clickstream can be divided into *sessions*, where a session represents the sequence of a user's clicks during a single visit to Renren. Unfortunately, users do not always explicitly end their session by logging out of Renren. As in prior work, we assume that a user's session is over if they do not make any requests for 20 minutes [6]. Session duration is calculated as the time interval between the first and last click within a session. Overall, our traces contain 113,595 sessions for Sybils and 467,179 sessions for normal users.

3 Preliminary Clickstream Analysis

We begin the analysis of our data by looking at the high-level characteristics of Sybil and normal users on Renren. Our goals are to provide an overview of the dataset, and to motivate the use of clickstreams as a rich data source for uncovering malicious behavior. Towards these ends, we analyze our data in four ways: *first*, we examine session-level characteristics. *Second*, we analyze the activities users engage in during each session. *Third*, we construct a state-based Markov Chain model to characterize the transitions between clicks during sessions. Finally, we use a Support Vector Machine (SVM) approach to learn the important features that distinguish Sybil and normal user clickstreams.

3.1 Session-level Characteristics

In this section, we seek to determine the session-level differences between normal and Sybil accounts in our dataset. First, we examine the total number of sessions from each user. As shown in Figure 1, >50% of Sybils have only a single session; far fewer than normal users. It is likely that these Sybils sent spam during this single session and were banned shortly thereafter. A small portion of Sybils are very active and have >100 sessions.

Next, we examine when Sybils and normal users are active each day. Figure 2 shows that all users exhibit a clear diurnal pattern, with most sessions beginning during daytime. This indicates that at least a significant portion of Sybils in our dataset could be controlled by real people exhibiting normal behavioral patterns.

Next, we investigate the number of sessions per user per day. Figure 3 shows that 80% of Sybils only login to Renren once per day or less, versus 20% of normal users. The duration of Sybil sessions is also much shorter, as shown in Figure 4: 70% of Sybil session are <100 seconds long, versus 10% of normal sessions. The vast ma-

jority of normal sessions last several minutes.

Figure 5 shows the number of clicks per session per user. Almost 60% of Sybil sessions only contain one click, whereas 60% of normal user sessions have $\geq$10 clicks. Not only do Sybil sessions tend to be shorter, but Sybils also click much faster than normal users. As shown in Figure 6, the average inter-arrival time between Sybil clicks is an order of magnitude shorter than for normal clicks. This indicates that Sybils do not linger on pages, and some of their activities may be automated.

The observed session-level Sybil characteristics are driven by attacker's attempts to circumvent Renren's security features. Renren limits the number of actions each account can take, *e.g.* 50 friend requests per day, and 100 profiles browsed per hour. Thus, in order to maximize efficiency, attackers create many Sybils, quickly login to each one and perform malicious activities (*e.g.* sending unsolicited friend requests and spam), then logout and move to the next Sybil. As shown in Table 2, Sybils spend a great deal of clicks sending friend requests and browsing profiles, despite Renren's security restrictions.

3.2 Clicks and Activities

Having characterized the session-level characteristics of our data, we now analyze the type and frequency clicks within each session. As shown in Table 2, we organize clicks into *categories* that correspond to high-level OSN features. Within each category there are *activities* that map to particular Renren features. In total, we observe 55 activities that can be grouped into 8 primary categories. These categories are:

- **Friending:** Includes sending friend requests, accepting or denying those requests, and un-friending.
- **Photo:** Includes uploading photos, organizing albums, tagging friends, browsing friend's photos, and writing comments on photos.
- **Profile:** This category encompasses browsing user profiles. Like Facebook, profiles on Renren can be browsed by anyone, but the information that is displayed is restricted by the owner's privacy settings.
- **Share:** Refers to users posting hyperlinks on their wall. Common examples include links to videos and news stories on external websites, or links to blog posts and photo albums on Renren.
- **Message:** Includes status updates, wall posts, and real-time instant-messages (IM).
- **Blog:** Encompasses writing blogs, browsing blog articles, and leaving comments on blogs.
- **Notification:** Refers to clicks on Renren's notification mechanism that alerts users to comments or likes on their content.

Category	Description	Sybil Clks		Nrml Clks	
		# (K)	%	# (K)	%
Friending	Send request	417	**41**	16	0
	Accept invitation	20	2	13	0
	Invite from guide	16	2	0	0
Photo	Visit photo	242	**24**	4,432	**76**
	Visit album	25	2	330	**6**
Profile	Visit profiles	160	**16**	214	4
Share	Share content	27	3	258	**4**
Message	Send IM	20	2	99	2
Blog	Visit/reply blog	12	1	103	2
Notification	Check notification	8	1	136	2

Table 2: Clicks from normal users and Sybils on different Renren activities. # of clicks are presented in thousands. Activities with <1% of clicks are omitted for brevity.

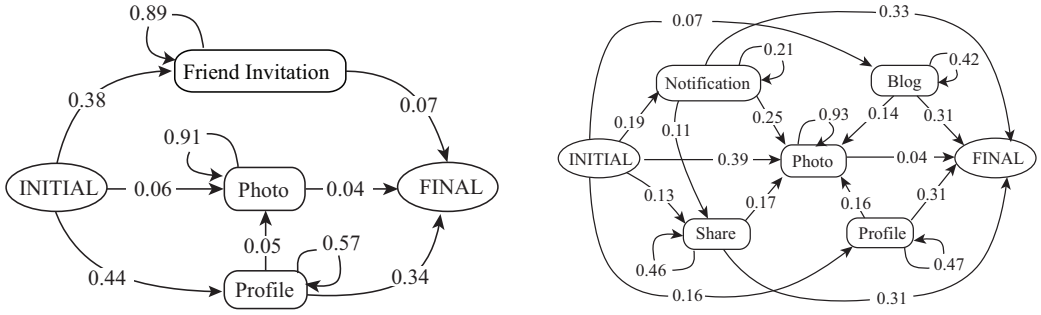
- **Like:** Corresponds to the user liking (or unliking) content on Renren.

Table 2 displays the most popular activities on Renren. The number of clicks on each activity is shown (in thousands), as well as the percent of clicks. Percentages are calculated for Sybils and normal users separately, *i.e.* each "%" column sums to 100%. For the sake of brevity, only activities with $\geq$1% of clicks for either Sybils or normal users are shown. The "Like" category has no activity with $\geq$1% of clicks, and is omitted from the table.

Table 2 reveals contrasting behavior between Sybils and normal users. Unsurprisingly, normal users' clicks are heavily skewed toward viewing photos (76%), albums (6%), and sharing (4%). In contrast, Sybils expend most of their clicks sending friend requests (41%), viewing photos (24%), and browsing profiles (16%). The photo browsing and profile viewing behavior are related: these Sybils crawl Renren and download users' personal information, including profile photos.

Sybils' clicks are heavily skewed toward friending (41% for Sybils, 0.3% for normal users). This behavior supports one particular attack strategy on Renren: friending normal users and then spamming them. However, given that other attacks are possible (*e.g.* manipulating trending topics [16], passively collecting friends [32]), we cannot rely on this feature alone to identify Sybils.

Normal users and Sybils share content (4% and 3%, respectively) as well as send messages (2% and 2%) at similar rates. This is an important observation, because sharing and messaging are the primary channels for spam dissemination on Renren. The similar rates of legitimate and illegitimate sharing/messaging indicate that spam detection systems cannot simply leverage numeric thresholds to detect spam content.



(a) State transitions for a Sybil account.

(b) State transitions for a real user.

Figure 7: Categories and transition probabilities in the clickstream models of Sybils and normal users.

3.3 Click Transitions

Sections 3.1 and 3.2 highlight some of the differences between Sybils and normal users. Next, we examine differences in click ordering, *i.e.* how likely is it for a user to transition from activity A to activity B during a single session?

We use a Markov Chain model to analyze click transitions. In this model, each state is a click category, and edges represent transitions between categories. We add two abstract states, initial and final, that mark the beginning and end of each click session. Figure 7 shows the category transition probabilities for both Sybils and normal users. The sum of all outgoing transitions from each category is 1.0. To reduce the complexity of the Figure, edges with probability <5% have been pruned (except for transitions to the final state). Categories with no incoming edges after this pruning process are also omitted.

Figure 7(a) demonstrates that Sybils follow a very regimented set of behaviors. After logging-in Sybils immediately begin one of three malicious activities: friend invitation spamming, spamming photos, or profile browsing. The profile browsing path represents crawling behavior: the Sybil repeatedly views user profiles until their daily allotment of views is exhausted.

Compared to Sybils, normal users (Figure 7(b)) engage in a wider range of activities, and the transitions between states are more diverse. The highest centrality category is photos, and it is also the most probable state after login. Intuitively, users start from their newsfeed, where they are likely to see and click on friends' recent photos. The second most probable state after login is checking recent notifications. Sharing and messaging are both low probability states. This makes sense, given that studies of interactions on OSNs have shown that users generate new content less than once per day [41, 17].

It is clear from Figure 7 that currently, Sybils on Renren are not trying to precisely mimic the behavior of normal users. However, we do not feel that this type of modeling represents a viable Sybil detection approach. Simply put, it would be trivial for Sybils to modify their behavior in order to appear more like normal users. If Sybils obfuscated their behavior by decreasing their transition probability to friending and profile browsing while increasing their transition probability to photos and blogs, then distinguishing between the two models would be extremely difficult.

3.4 SVM Classification

The above analysis shows that Sybil sessions have very different characteristics compared to normal user sessions. Based on these results, we explore the possibility of distinguishing normal and Sybil sessions using a Support Vector Machine (SVM) [26]. For our SVM experiments, we extract 4 features from session-level information and 8 features from click activities:

- **Session Features:** We leverage 4 features extracted from user sessions: average clicks per session, average session length, average inter-arrival time between clicks, and average sessions per day.
- **Click Features:** As mentioned in Section 3.2, there are 8 categories of clicks activities on Renren. For each user, we use the percentage of clicks in each category as a feature.

We computed values for all 12 features for all users in our dataset, input the data to an SVM, and ran 10 fold cross-validation. The resulting classification accuracy was 98.9%, with 0.8% false positives (*i.e.* classify normal users as Sybils) and 0.13% false negatives (*i.e.* classify Sybils as normal users). Table 3 shows the weights assigned to the top 5 features. Features with positive weight values are more indicative of Sybils, while features with negative weights indicate they are more likely in normal users. Overall, higher absolute value of the weights corresponds to features that more strongly indicate either Sybils or normal users. These features agree with our ad-hoc observations in previous sections.

Feature	Weight
% of clicks under *Friending*	+5.65
% of clicks under *Notification*	-3.68
Time interval of clicks (TBC)	-3.73
Session length (SL)	+1.34
% of clicks under *Photo*	+0.93

Table 3: Weight of features generated by SVM.

While our SVM results are quite good, an SVM-based approach is still a supervised learning tool. In practice, we would like to avoid using any ground truth datasets to train detection models, since they can introduce unknown biases. Later, we will describe our unsupervised detection techniques in detail.

3.5 Discussion

In summary, we analyze the Renren clickstream data to characterize user behavior from three angles: sessions, click activities, and click transitions. SVM analysis of these basic features demonstrates that clickstreams are useful for identifying Sybils on social networks.

However, these basic tools (session distributions, Markov Chain models, SVM) are of limited use in practice: they require training on large samples of ground-truth data. For a practical Sybil detection system, we must develop clickstream analysis techniques that leverage unsupervised learning on real-time data samples, *i.e.* require zero or little ground-truth. In the next section, we will focus on developing clickstreams models for real-time, unsupervised Sybil detection.

4 Clickstream Modeling and Clustering

In Section 3, we showed that clickstream data for Sybils and normal users captured the differences in their behavior. In this section, we build models of user activity patterns that can effectively distinguish Sybils from normal users. Our goal is to cluster similar clickstreams together to form general user "profiles" that capture specific activity patterns. We then leverage these clusters (or profiles) to build a Sybil detection system.

We begin by defining three models to represent a user's clickstream. For each model, we describe similarity metrics that allow us to cluster similar clickstreams together. Finally, we use our ground-truth data to evaluate the efficacy of each model in distinguishing Sybils from normal users. We build upon these results later to develop practical Sybil detection systems based on clickstream analysis.

4.1 Clickstream Models

We define three models to capture a user's clickstream.

Click Sequence Model. We start with the most straightforward model, which only considers click events. As shown in Section 3, Sybils and normal users exhibit different click transition patterns and focus their energy on different activities. The Click Sequence (CS) Model treats each user's clickstream as a sequence of click events, sorted by order of arrival.

Time-based Model. As shown in Figure 6, Sybils and normal users generate click events at different speeds. The Time-based Model focuses on the distribution of gaps between events: each user's clickstream is represented by a list of inter-arrival times $[t_1, t_2, t_3, ..., t_n]$ where n is the number of clicks in a user's clickstream.

Hybrid Model. The Hybrid Model combines click types and click inter-arrival times. Each user's clickstream is represented as an in-order sequence of clicks along with inter-event gaps between clicks. For example: $a(t_1)c(t_2)a(t_3)d(t_4)b$ where a, b, c, d are click types, and t_i is the time interval between the i^{th} and $(i+1)^{th}$ event.

Click Types. Both the Click Sequence Model and the Hybrid Model represent each event in the sequence by its click event type. We note that we can control how granular the event types are in our sequence representation. One approach is to encode clicks based on their specific *activity*. Renren's logs define 55 unique activities. Another option is to encode click events using their broader *category*. In our dataset, our 55 activities fall under 8 click categories (see Section 3.2). Our experimental analysis evaluates both representations to understand the impact of granularity on model accuracy.

4.2 Computing Sequence Similarity

Having defined three models of clickstream sequences, we now move on to investigating methods to quantify the similarity between clickstreams. In other words, we want to compute the *distance* between pairs of clickstreams. First, we discuss general approaches to computing the distance between sequences. Then we discuss how to apply each approach to our three clickstream models.

4.2.1 Defining Distance Functions

Common Subsequences. The first distance metric involves locating the common subsequences of varying lengths between two clickstreams. We formalize a clickstream as a sequence $S = (s_1 s_2 ... s_i ... s_n)$, where s_i is the i^{th} element in the sequence. We then define T_k as the set of all possible k-grams (k consecu-

tive elements) in sequence S: $T_k(S) = \{k\text{-}gram | k\text{-}gram = (s_i s_{i+1} ... s_{i+k-1}), i \in [1, n+1-k]\}$. Simply put, each k-gram in $T_k(S)$ is a subsequence of S. Finally, the distance between two sequences can then be computed based on the number of common subsequences shared by the two sequences. Inspired by the *Jaccard Coefficient* [19], we define the distance between sequences S_1 and S_2 as:

$$D_k(S_1, S_2) = 1 - \frac{|T_k(S_1) \cap T_k(S_2)|}{|T_k(S_1) \cup T_k(S_2)|} \quad (1)$$

We will discuss the choice of k in Section 4.2.2.

Common Subsequences With Counts. The common subsequence metric defined above only measures distinct common subsequences, *i.e.* it does not consider the frequency of common subsequences. We propose a second distance metric that rectifies this by taking the *count* of common subsequences into consideration. For sequences S_1, S_2 and a chosen k, we first compute the set of all possible subsequences from both sequences as $T = T_k(S_1) \cup T_k(S_2)$. Next, we count the frequency of each subsequence within each sequence i ($i = 1, 2$) as array $[c_{i1}, c_{i2}, ..., c_{in}]$ where $n = |T|$. Finally, the distance between S_1 and S_2 can be computed as the normalized *Euclidean Distance* between the two arrays:

$$D(S_1, S_2) = \frac{1}{\sqrt{2}} \sqrt{\sum_{j=1}^{n} (c_{1j} - c_{2j})^2} \quad (2)$$

Distribution-based Method. Unfortunately, the prior metrics cannot be applied to sequences of continuous values (*i.e.* the Time-based Model). Instead, for continuous value sequences S_1 and S_2, we compute the distance by comparing their value distribution using a two-sample KolmogorovSmirnov test (*K-S* test). A two-sample *K-S* test is a general nonparametric method for comparing two empirical samples. It is sensitive to differences in location and shape of the empirical Cumulative Distribution Functions (CDF) of the two samples. We define the distance function using K-S statistics:

$$D(S_1, S_2) = sup_t |F_{n,1}(t) - F_{n',2}(t)| \quad (3)$$

where $F_{n,i}(t)$ is the CDF of values in sequence S_i.

4.2.2 Applying Distances Functions to Clickstreams

Having defined three distance functions for computing sequence similarity, we now apply these metrics to our three clickstream models. Table 4 summarizes the distance metrics we apply to each of our models. The Time-based Model is the simplest case, because it only has one corresponding distance metric (K-S Test). For the Click Sequence and Hybrid Models, we use several different parameterizations of our distance metrics.

Model	Distance Metrics
Click Sequence Model	*unigram*, *unigram+count*, *10gram*, *10gram+count*
Time-based Model	*K-S* test
Hybrid Model	*5gram*, *5gram+count*

Table 4: Summary of distance functions.

Click Sequence Model. We use the common subsequence and common subsequence with counts metrics to compute distances in the CS model. However, these two metrics require that we choose k, the length of k-gram subsequences to consider. We choose two values for k: 1 and 10, which we refer to as *unigram* and *10gram*. *Unigram* represents the trivial case of comparing common click events in two clickstreams, while ignoring the ordering of clicks. *10gram* includes all unigrams, as well as bigrams, trigrams, *etc*. As shown in Table 4, we also instantiate *unigram+count* and *10gram+count*, which include the frequency counts of each unique subsequence.

Although values of $k > 10$ are possible, we limit our experiments to $k = 10$ for two reasons. First, when $k = n$ (where n is the length of a clickstream), the computational complexity becomes $O(n^2)$. This overhead is significant when you consider that $O(n^2)$ subsequences will be computed for every user in a clickstream dataset. Second, long subsequences have diminishing utility, because they are likely to be unique for a particular user. In our tests, we found $k = 10$ to be a good limit on computational overhead and subsequence over-specificity.

Hybrid Model. Like the Click Sequence Model, distances between sequences in the Hybrid Model can also be computed using the common subsequence and common subsequence plus count metrics. The only change between the Click Sequence and Hybrid Models is that we must discretize the inter-arrival times between clicks so they can be encoded into the sequence. We do this by placing inter-arrival times into log-scale buckets (in seconds): $[0,1], [1,10], [10,100], [100,1000], [1000,\infty]$. Based on Figure 6, the inter-arrival time distribution is highly skewed, so log-scale buckets are better suited than linear buckets to evenly encode the times.

After we discretize the inter-arrival times and insert them into the clickstream, we use $k = 5$ as the parameter for configuring the two distance metrics. Further increasing k offers little improvement in the model but introduces extra computation overhead. As shown in Table 4, we refer to these metrics as *5gram* and *5gram+count*. Thus, each 5gram contains three consecutive click events along with two tokens representing inter-arrival time gaps between them.

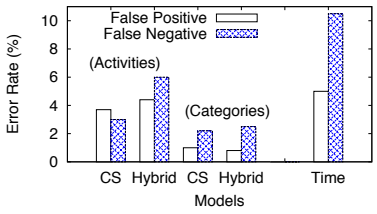


Figure 8: Error rate of three models.

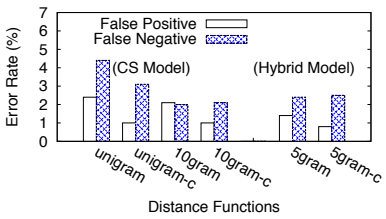


Figure 9: Error rate using different distance functions.

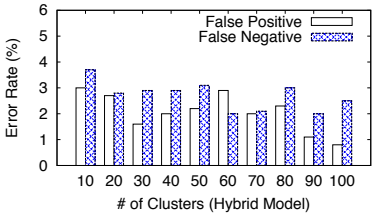


Figure 10: Impact of number of clusters (K).

4.3 Sequence Clustering

At this point we have defined models of clickstreams as well as metrics for computing the distance between them. Our next step is to cluster users with similar clickstreams together. As shown in Section 3, Sybil and normal users exhibit very different behaviors, and should naturally form distinctive clusters.

To achieve our goal, we build and then partition a *sequence similarity graph*. Each user's clickstream is represented by a single node. The sequence similarity graph is complete, *i.e.* every pair of nodes is connected by a weighted edge, where the weight is the similarity distance between the sequences. Partitioning this graph means producing the desired clusters while minimizing the total weight of cut edges: users with similar activities (high weights between them) will be placed in the same cluster, while users with dissimilar activities will be placed in different clusters. Thus the clustering process separates Sybil and normal users. Note that not all Sybils and normal users exhibit homogeneous behavior; thus, we expect there to be multiple, distinct clusters of Sybils and normal users.

Graph Clustering. To cluster sequence similarity graphs, we use METIS [18], a widely used multilevel k-way partitioning algorithm. The objective of METIS is to minimize the weight of edges that cross partitions. In the sequence similarity graph, longer distances (*i.e.* dissimilar sequences) have lower weights. Thus, METIS is likely to place dissimilar sequences in different partitions. METIS requires a parameter K that specifies the number of partitions desired. We will assess the impact of K on our system performance in Section 4.4.

Cluster Quality. A key question when evaluating our methodology is assessing the quality of clusters produced by METIS. In Section 4.4, we leverage our ground-truth data to evaluate false positives and negatives after clustering the sequence similarity graph. We label each cluster as "Sybil" or "normal" based on whether the majority of nodes in the cluster are Sybils or normal users. Normal users that get placed into Sybil clusters are false positives, while Sybils placed in normal clusters are false negatives. We use these criteria to evaluate different clickstream models and distance functions.

4.4 Model Evaluation

We now evaluate our clickstream models and distance functions to determine which can best distinguish Sybil activity patterns from those of normal users. We examine four different variables: 1) choice of clickstream model, 2) choice of distance function for each model, 3) what representation of clicks to use (specific activities or categories), and 4) K, the number of desired partitions for METIS.

Experiment Setup. The experimental dataset consists of 4000 normal users and 4000 Sybils randomly selected from our dataset. In each scenario, we build click sequences for each user (based on a given clickstream model and click representation), compute all distances between each pair of sequences, and then cluster the resulting sequence similarity graph for a given value of K. Finally, each experimental run is evaluated based on the false positive and negative error rates.

Model Analysis. First, we examine the error rates of different clickstream models and click representations in Figure 8. For the CS and Hybrid models, we encode clicks based on activities as well as categories. In the Time model, all clicks are encoded as inter-arrival times. In this experiment, we use *10gram+count*, *5gram+count*, and *K-S* as the distance function for CS, Hybrid, and Time, respectively. We fix $K = 100$. We investigate the impact of distance functions and K in subsequent experiments.

Two conclusions can be drawn from Figure 8. First, the CS and Hybrid models significantly outperform the Time-based model, especially in false negatives. This demonstrates that click inter-arrival times alone are insufficient to disambiguate Sybils from normal users. Manual inspection of false negative Sybils from the Time experiment reveals that these Sybils click at the same rate as normal users. Thus these Sybils are either operated by real people, or the software that controls them has been intentionally rate limited.

The second conclusion from Figure 8 is that encoding clicks based on category outperforms encoding by activity. This result confirms findings from the existing literatures on web usage mining [3]: representing clicks using high-level categories (or *concepts*) instead of raw click types better exposes the browsing patterns of users. A possible explanation is that high-level categories have better tolerance for noise in the clickstream log. In the rest of our paper, we use categories to encode clicks.

Next, we examine the error rate of different distance functions for the CS and Hybrid models. As shown in Figure 9, we evaluate the CS model using the *unigram* and *10gram* functions, as well as counting versions of those functions. We evaluate the Hybrid model using the *5gram* and *5gram+count* functions.

Several conclusions can be drawn from Figure 9. First, the *unigram* functions have the highest false negative rates. This indicates that looking at clicks in isolation (*i.e.* without click transitions) is insufficient to discover many Sybils. Second, the counting versions of all three distance functions produce low false positive rates. This demonstrates that the repeat frequency of sequences is important for identifying normal users. Finally, we observe that CS *10gram+count* and Hybrid have similar accuracy. This shows that click inter-arrival times are not necessary to achieve low error rates.

Finally, we examine the impact of the number of clusters K on detection accuracy. Figure 10 shows the error rate of Hybrid *5gram+count* as we vary K. The overall trend is that larger K produces lower error rates. This is because larger K grants METIS more opportunities to partition weakly connected sequences. This observation is somewhat trivial: if $K = N$, where N is the number of sequences in the graph, then the error rate would be zero given our evaluation methodology. In Section 6, we discuss practical reasons why K must be kept $\approx$100.

Summary. Our evaluation shows that the Click Sequence and Hybrid models perform best at disambiguating Sybils and normal users. *10gram+count* and *5gram+count* are the best distance functions for each model, respectively. We find that accuracy is highest when clicks are encoded based on categories, and when the number of partitions K is large. In the following sections, we will use these parameters when building our Sybil detection system.

5 Incremental Sybil Detection

Our results in Section 4 showed that our models can effectively distinguish between Sybil clickstreams and normal user clickstreams. In this section, we leverage this methodology to build a real-time, incremental Sybil detector. This system works in two phases: first, we create clusters of Sybil and normal users based on ground-truth data, as we did in Section 4. Second, we compute the position of unclassified clickstreams in our sequence similarity graph. If an unclassified clickstream falls into a cluster representing clickstreams from ground-truth Sybils, we conclude the new clickstream is a Sybil. Otherwise, it is benign.

5.1 Incremental Detection

To classify a new clickstream given an existing clustered sequence similarity graph, we must determine how to "re-cluster" new sequences into the existing graph. We investigate three algorithms.

The first is *K Nearest Neighbor* (KNN). For a given unclassified sequence, we find the top-K nearest sequences in the ground-truth data. If the majority of these sequences are located in Sybil clusters, then the new sequence is classified as a Sybil sequence.

The second algorithm is *Nearest Cluster* (NC). We compute the average distance from an unclassified sequence to all sequences in each cluster. The unclassified sequence is then added to the cluster with the closest average distance. The new sequence is classified as Sybil or normal based on the cluster it is placed in.

The third algorithm is a less computationally-intensive version of Nearest Cluster that we refer to as *Nearest Cluster-Center* (NCC). NC and KNN require computing the distance from an unclassified sequence to all sequences in the ground-truth clusters. We can streamline NC's classification process by precomputing *centers* for each cluster. In NCC, we only need to compute the distance from an unclassified sequence to the center of each existing cluster.

For each existing cluster, the center is chosen by *closeness centrality*. Intuitively, the center sequence is the one that has the shortest distance to all the other sequences in the same cluster. To be more robust, we precompute three centers for each cluster, that is, the three sequences with highest closeness centrality.

5.2 System Evaluation

In this section, we evaluate our incremental Sybil detection system using our ground-truth clickstream dataset. We start by evaluating the basic accuracy of the system at classifying unknown sequences. Next, we evaluate how quickly the system can identify Sybils, in terms of number of clicks in their clickstream. Finally, we explore how long the system can remain effective before it needs to be retrained using updated ground-truth data.

Detection Accuracy. We start with a basic evaluation of system accuracy using our ground-truth dataset. We split the dataset into training data and testing data. Both datasets include 3000 Sybils and 3000 normal users. We build sequence similarity graphs from the training data

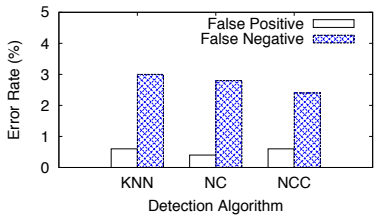


Figure 11: Error rate of three reclustering algorithms.

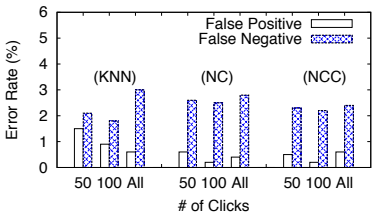


Figure 12: Error rate vs. maximum # of clicks in each sequence.

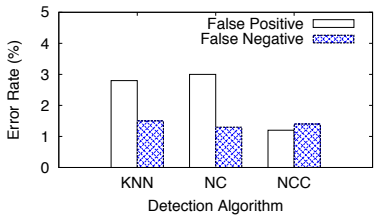


Figure 13: Detection accuracy when training data is two weeks old.

using Hybrid Model with *5gram+count* as distance function. The number of clusters $K = 100$. In each sequence similarity graph, we label the Sybil and normal clusters.

Next, we examine the error rates of the incremental detector when unclassified users (3000 Sybils and 3000 normal users) are added to the sequence similarity graph. We perform this experiment three times, once for each of the proposed reclustering algorithms (KNN, NC and NCC). As shown in Figure 11, the error rates for all three reclustering algorithms are very similar, and all three have $<1\%$ false positives. NC has slightly fewer false positives, while NCC has the fewest false negatives.

Detection Speed. The next question we want to address is: *what is the minimum number of clicks necessary to accurately classify clickstreams?* Another way to frame this question is in terms of detection speed: *how quickly (in terms of clicks) can our system accurately classify clickstreams?* To identify and respond to Sybils quickly, we must detect Sybils using the minimal number of click events.

Figure 12 shows the results of our evaluation when the maximum number of clicks in all sequences are capped. The "All" results refer to a cap of infinity, *i.e.* all clicks in our dataset are considered. Note that not all sequences in our dataset have 50 or 100 clicks: some Sybils were banned before they produced this may clicks. Hence, the caps are upper bounds on sequence length.

Surprisingly, the "All" results are not the most accurate overall. As shown in Figure 12, using all clicks results in more false negatives. This occurs due to overfitting: given a large number of very long clickstreams from normal users, it is likely that they will occasionally exhibit unusual, Sybil-like behavior. However, this problem is mitigated if the sequence length is capped, since this naturally excludes these infrequent, aberrant clickstreams.

In contrast to the "All" results, the results from the ≤ 50 click experiments produce the most false positives. This demonstrates that there is a minimum sequence length necessary to perform accurate classification of clickstreams. We repeated these experiments using CS/*10gram+count* and received similar result, which we omit for brevity.

There are two additional, practical take-aways from Figure 12. First, the NCC algorithm performs equally well versus NC and KNN. This is a positive result, since the computational complexity of NCC is dramatically lower than NC and KNN. Second, we observe that our system can produce accurate results (false positives $<1\%$, false negatives $<3\%$) when only considering short sequences. This means that the system can make classifications quickly, without needing to store very long clickstreams in memory.

Accuracy Over Time. In order for our incremental detection system to be practically useful, its accuracy should remain high for long periods of time. Put another way, sequence similarity graphs trained with old data should be able to detect fresh Sybil clickstreams. To evaluate the accuracy of our system over time, we split our dataset based on date. We train our detector using the early data, and then apply the detector to the later data. We restrict our analysis to data from April 2011; although we have Sybil data from March 2011, we do not have corresponding data on normal users for this month.

Figure 13 shows the accuracy of the detector when it is trained on data from March 31-April 15, then applied to data from April 16-30. As the results show, the detector remains highly accurate for at least two weeks after it has been trained using the NCC reclustering algorithm. Unfortunately, the limited duration of our dataset prevents us from examining accuracy at longer time intervals.

We repeated this experiment using only one week of training data, but the false negative rate of the detector increased to $\approx 10\%$. This shows that the detector needs to be trained on sufficient data to provide accurate results.

6 Unsupervised Sybil Detection

Our incremental Sybil detection system from Section 5 has a serious shortcoming: it must be trained using large samples of ground-truth data. In this section, we develop an unsupervised Sybil detection system that requires only a small, constant amount of ground-truth. The key idea is to build a clustered sequence similarity graph as before. But instead of using full ground-truth

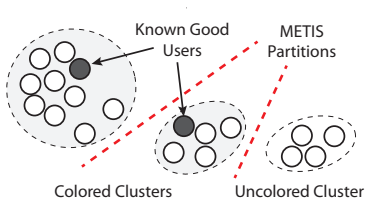


Figure 14: Unsupervised clustering with coloring.

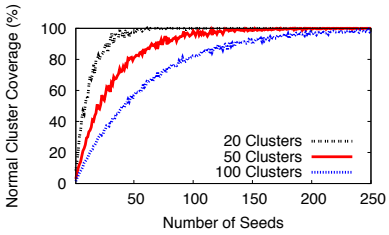


Figure 15: # of seeds vs. % of correctly colored normal user clusters.

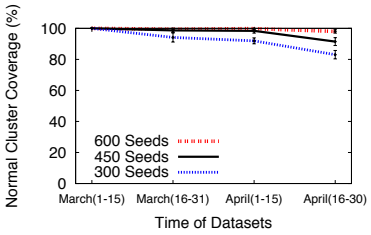


Figure 16: Consistency over time of normal seeds for coloring.

of all clickstreams to mark a cluster as Sybil or normal, we only need a small number of clickstreams of known real users as "seeds" that color the clusters they reside in. These seeds can be manually verified as needed. We *color* all clusters that include a *seed* sequence as "normal," while uncolored clusters are assumed to be "Sybil." Since normal users are likely to fall under a small number of behavioral profiles (clusters in the graph), we expect a small fixed number of seeds will be sufficient to color all clusters of normal user clickstreams.

Figure 14 depicts our unsupervised approach, showing how METIS partitions nodes into clusters which are then colored if they contain seed users. Once the system is trained in this manner, it can be used incrementally to detect more Sybils over time, as described in Section 5.

In this section, we discuss the design of our unsupervised system and evaluate its performance. We begin by analyzing the number and composition of seeds that are necessary to ensure high accuracy of the system. Next, we evaluate the performance of the system by comparing its accuracy to our ground-truth data. Finally, we examine how the ratio of Sybils to normal users in the unclassified data impacts system accuracy.

6.1 Seed Selection and Composition

Number of Seeds. The most important parameter in our unsupervised Sybil detection system is the number of seeds. On one hand, the number of seeds needs to be large and diverse enough to color all "normal" clusters. Normal clusters that remain uncolored are potential false positives. On the other hand, the seed set needs to be small enough to be practical. If the size of the seed set is large, it is equivalent to having ground-truth about the dataset, which is the situation we are trying to avoid.

We now conduct experiments to determine how many seeds are necessary to color the clusters. We choose 3000 Sybils and 3000 normal users at random from our dataset to be the unclassified dataset. We also randomly choose some number of additional normal users to be the seeds. As in Section 5, we use the Hybrid Model with the *5gram+count* distance function. We also conducted experiments with CS/*10gram+count*, but the results are very similar and we omit them for brevity.

Figure 15 depicts the percentage of normal of clusters that are correctly colored for different values of K (number of METIS partitions) as the number of seeds is varied. As expected, fewer seeds are necessary when K is small because there are fewer clusters (and thus each cluster includes more sequences). When $K = 100$, 250 seeds (or 4% of all normal users in the experiment) are able to color 99% of normal clusters.

Seed Consistency Over Time. Next, we examine whether a set of seeds chosen at an early date are equally effective at coloring clusters based on later data. In other words, we want to know if the seeds are consistent over time. If this is not the case, it would represent additional overhead on the deployment of our system.

To test seed consistency over time, we divide our two months of Sybil clickstream data into four, two-week long datasets. We add an equal number of randomly selected normal users to each of the four datasets. Finally, we select an additional x random normal users to act as seeds. We verify (for each value of x) that these seeds color 100% of the normal clusters in the first temporal dataset. We then evaluate what percentage of normal clusters are colored in the subsequent three temporal datasets. In all experiments, we set $K = 100$, *i.e.* the worst case scenario for our graph coloring approach.

The results of the temporal consistency experiments are shown in Figure 16. In general, even though the Sybil and normal clickstreams change over time, the vast majority of normal clusters are successfully colored. Given 600 seeds, 99% of normal clusters are colored after 4 weeks, although the percentage drops to 83% with 300 seeds. These results demonstrate that the seed set does not need to be drastically altered over time.

6.2 Coloring Evaluation

We now evaluate the overall effectiveness of our Sybil detection system when it leverages unsupervised training. In these experiments, we use our entire clickstream dataset. We choose x random normal users as seeds,

build and cluster the sequence similarity graph using Hybrid/*5gram+count*, and then color the clusters that contain the seeds. Finally, we calculate the false positive and negative rates using the same methodology as in Section 5, *i.e.* by comparing the composition of the colored clusters to ground-truth.

The results are shown in Figure 17. As the number of seeds increases, the false positive rate decreases. This is because more seeds mean more normal clusters are correctly colored. With just 400 seeds, the false positive rate drops to $<1\%$. Unfortunately, relying on unsupervised training does increase the false negative rate of our system by 2% versus training with ground-truth data. However, in cases where ground-truth data is unavailable, we believe that this is a reasonable trade-off. Note that we also repeated these experiment with CMS/*10gram+count*, and it produced slightly higher false positive rates, although they were still $<1\%$.

Unbalanced Training Dataset. Next, we evaluate the impact of having an unbalanced training dataset (*e.g.* more normal users than Sybils) on the accuracy of our system. Thus far, all of our experiments have assumed a roughly equal percentage of Sybils and normal users in the data. However, in practice it is likely that normal users will outnumber Sybils when unsupervised learning is used. For example, Facebook suspects that 8.7% of its user base is illegitimate, out of >1 billion total users [1].

We now evaluate how detection accuracy changes when we decrease the percentage of Sybils in the training data. In these experiments, we construct training sets of 6000 total users with different normal-to-Sybil ratios. We then run unsupervised training with 500 seeds. Finally, we incrementally add an additional 3000 Sybils and 3000 normal users to the colored similarity graph using the NCC algorithm (see Section 5.1). We ran additional tests using the NC and KNN algorithms, but the results were very similar and we omit them for brevity.

Figure18 shows the final error rate of the system (*i.e.* after 6000 users have been incrementally added) for varying normal-to-Sybil ratios. The false positive rate remains $\leq 1.2\%$ regardless of the normal-to-Sybil ratio. This is a very good result: even with highly skewed training data, the system is unlikely to penalize normal users. Unfortunately, the false negative rate does rise as the number of Sybils in the training data falls. This result is to be expected: the system cannot adequately classify Sybil clickstreams if it is trained on insufficient data.

Handling False Positives. The above analysis demonstrates that our system achieves high accuracy with a false positive rate of 1% or less. Through manual inspection, we find that "false positives" generated by our detector exhibit behaviors generally attributed to Sybils, including aggressively sending friend requests or browsing profiles. In real-world OSNs, suspicious users identified by our system could be further verified via existing complementary systems that examines other aspects of users. For example, this might include systems that classify user profiles [32, 43], systems that verify user real-world identity [2], or even Sybil detection systems using crowdsourced human inspection [38]. These efforts could further protect benign users from misclassification.

7 Practical Sybil Detection

In this section, we examine the practical performance of our proposed Sybil detection system. First, we shipped our code to the security teams at Renren and LinkedIn, where it was evaluated on fresh data in a production environment. Both test results are very positive, and we report them here. Second, we discuss the fundamental limits of our approach, by looking at our impact on Sybil accounts that can perfectly mimic the clickstream patterns of normal users.

7.1 Real-world Sybil Detection

With the help of supportive collaborators at both Renren and LinkedIn, we were able to ship prototype code to the security teams at both companies for internal testing on fresh data. We configured our system to use unsupervised learning to color clusters. Sequence similarity graphs are constructed using the Hybrid Model and the *5gram+count* distance function, and the number of METIS partitions K is 100.

Renren. Renren's security team trained our system using clickstreams from 10K users, of which 8K were randomly selected, and 2K were previously identified as suspicious by the security team. These clickstreams were collected between January 17–27, 2013. 500 honest users that have been manually verified by Renren's security team were used as seeds. Once trained, our system was fed clickstreams from 1 million random users (collected in early February, 2013) for classification as normal or suspicious. In total, our system identified 22K potential Sybil accounts. These accounts are now being investigated by the security team.

While corporate privacy policies prevented Renren from sharing detailed results with us, their feedback was very positive. They also indicated that our system identified a new attack performed by a large cluster of users whose clickstream behavior focused heavily on photo sharing. Manual inspection revealed that these photos used embedded text to spread spam for brands of clothes and shoes. Traditional text analysis-based spam detectors and URL blacklists were unable to catch this new attack, but our system identified it immediately.

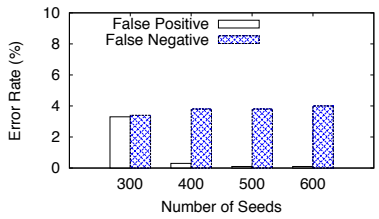


Figure 17: Detection accuracy versus number of seeds.

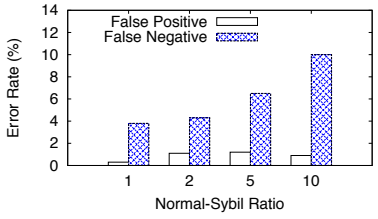


Figure 18: Detection accuracy versus Normal-Sybil ratio.

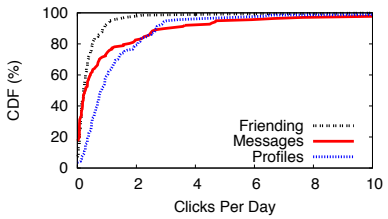


Figure 19: Clicks per day by outlier normal users.

LinkedIn. LinkedIn's security team used our software to analyze the clickstreams of 40K users, of which 36K were randomly sampled, and 4K were previously identified as suspicious by the security team. These clickstreams were gathered in February, 2013. Again, our feedback was very positive, but did not include precise statistics. However, we were told that our system confirmed that ≈1700 of the 4000 suspicious accounts are likely to be Sybils. Our system also detected an additional 200 previously unknown Sybils.

A closer look at the data shows that many of the accounts not detected by our system were borderline accounts with specific flags popping up in their profiles. For example, some accounts had unusual names or occupational specialties, while others had suspicious URLs in their profiles. These results remind us that a behavior model is clearly only a part of the equation, and should be used in conjunction with existing profile analysis tools and spam detectors [5, 10, 37, 38, 44].

Ongoing Collaboration. In summary, the security teams at both Renren and LinkedIn were very pleased with the initial results of our system. We plan to continue collaborating with both groups to improve our system and implement it in production.

7.2 Limits of Sybil Detection

Finally, we wish to discuss the worst case scenario for our system, *i.e.* a scenario where attackers have full knowledge of the clickstream patterns for real users, and are able to instrument the behavior of their Sybils to mimic them precisely. In this attack model, the attacker's goal is to have Sybils carry out malicious actions (*e.g.* sending spam) without being detected. However, to evade detection, these Sybils must limit themselves to behavior consistent with that of normal users.

We can thus bound the capabilities of Sybils that avoid detection in this attack model. First, the Sybil's clickstream must remain inside the "normal" clusters produced by our detector. Second, the most aberrant behavior within a given "normal" cluster is exhibited by real users within the cluster who are farthest from the center. The activities performed by these *outliers* serve as effective bounds on Sybil behavior. Sybil clickstreams cannot deviate from the center of the cluster more than these outliers, otherwise they will be excluded from the cluster and risk detection. Thus, we can estimate the maximum amount of malicious activity a Sybil could perform (without getting caught) by studying these outliers.

We now examine the behavior of outliers. We calibrate our system to produce clusters with false positive rate $<1\%$ using Hybrid/*5gram+count*, and $K = 100$. In this configuration, the detector outputs 40 Sybil and 60 normal clusters when run on our full dataset. Next, we identify the two farthest outliers in each normal cluster. Finally, we plot the clicks per day in three activities from the 120 outliers in Figure 19. We focus on clicks for sending friend requests, posting status updates/wall messages, and viewing user profiles. These activities correspond to the three most common attacks we observe in our ground-truth data, *i.e.* sending friend request spam, status/wall spam, and profile crawling.

As shown in Figure 19, 99% of outliers generate ≤ 10 clicks per day in the target activities. In the vast majority of cases, even the outliers generate <2 clicks per day. These results show that the effective bound on Sybil behavior is very tight, *i.e.* to avoid detection, Sybils can barely generate any clicks each day. These bounds significantly increase the cost for attackers, since they will need many more Sybils to maintain the same level of spam generation capacity.

8 Related Work

Sybil Detection on OSNs. Studies have shown that Sybils are responsible for large amounts of spam on Facebook [10], Twitter [11, 32], and Renren [43]. Various systems have been proposed by the research community to detect and mitigate these Sybils. One body of work leverages social graphs to detect Sybils. These systems detect tight-knit Sybil communities that have a small quotient-cut from the honest region of the graph [46, 45, 34, 36, 8, 7]. However, recent studies have demonstrated the limitations of this approach. Yang *et al.*

show that Sybils on Renren blend into the social graph rather than forming tight communities [43]. Mohaisen *et al*. show that many social graphs are not fast-mixing, which is a necessary precondition for community-based Sybil detectors to be effective [21].

A second body of work has used machine learning to detect Sybil behavior on Twitter [44, 5, 37] and Facebook [31]. However, relying on specific features makes these systems vulnerable to Sybils with different attack strategies. Finally, one study proposes using crowdsourcing to identify Sybils [38].

Web Usage Mining. Researchers have studied the usage patterns of web services for the last decade [30]. Several studies focus on session-level analysis to learn user's browsing habits [14, 13, 24]. Others develop session clustering techniques [4, 42, 40, 33, 25], Markov Chain models [20, 28], and tree-based models [12] to characterize user browsing patterns. We also leverage a Markov Chain model and clustering in our work. Two studies have focused specifically on characterizing clickstreams from OSNs [6, 29].

The vast majority of the web usage mining literature focuses on characterizing the behavior of normal users. To the best of our knowledge, there are only two studies that leverage clickstreams for anomaly detection [15, 28]. Both of these studies use session-level features to identify crawlers, one focusing on e-commerce and the other on search engines. Their techniques (*e.g*. session distributions, Markov Chain models) require training on large samples of ground-truth data, and cannot scale to today's large social networks.

9 Conclusion

To the best of our knowledge, this is the first work to leverage clickstream models for detecting malicious users in OSNs. Our results show that we can build an accurate Sybil detector by identifying and coloring clusters of "similar" clickstreams. Our system has been validated on ground-truth data, and a prototype has already detected new types of image-spam attacks on Renren.

We believe clickstream models can be a powerful technique for user profiling in contexts outside of OSNs. In our ongoing work, we are studying ways to extend clickstream models to detect malicious crowdsourcing workers and forged online product and travel reviews.

IRB Protocol

This work was carried out under an approved IRB protocol. All data was anonymized by Renren prior to our use. The clickstreams are old enough that the events they describe are no longer accessible via the current website. All experiments run on recent user data were conducted on-site at Renren and LinkedIn respectively, and all results remain on-site.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback, and Yanjie Liang (Renren) and David Freeman (LinkedIn) for their assistant in experiments. This work is supported in part by NSF grants CNS-1224100 and IIS-0916307 and DARPA GRAPHS (BAA-12-01). 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] Facebook has more than 83 million illegitimate accounts. BBC News, August 2012.

[2] Verify facebook account. `https://www.facebook.com/help/398085743567023/`, 2013.

[3] BANERJEE, A., AND GHOSH, J. Concept-based clustering of clickstream data. In *Proc. of ICIT* (2000).

[4] BANERJEE, A., AND GHOSH, J. Clickstream clustering using weighted longest common subsequences. In *Proc. of the Web Mining Workshop, SIAM Conference on Data Mining* (2001).

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

[6] BENEVENUTO, F., RODRIGUES, T., CHA, M., AND ALMEIDA, V. Characterizing user behavior in online social networks. In *Proc. of IMC* (2009).

[7] CAO, Q., SIRIVIANOS, M., YANG, X., AND PREGUEIRO, T. Aiding the detection of fake accounts in large scale social online services. In *Proc. of NSDI* (2012).

[8] DANEZIS, G., AND MITTAL, P. Sybilinfer: Detecting sybil nodes using social networks. In *Proc of NDSS* (2009).

[9] DOUCEUR, J. R. The Sybil attack. In *Proc. of IPTPS* (2002).

[10] 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).

[11] GRIER, C., THOMAS, K., PAXSON, V., AND ZHANG, M. @spam: the underground on 140 characters or less. In *Proc. of CCS* (2010).

[12] GÜNDÜZ, C., AND ÖZSU, M. T. A web page prediction model based on click-stream tree representation of user behavior. In *Proc. of SIGKDD* (2003).

[13] HEER, J., AND CHI, E. H. Mining the structure of user activity using cluster stability. In *Proc. of the Workshop on Web Analytics, SIAM Conference on Data Mining* (2002).

[14] HEER, J., AND CHI, E. H. Separating the swarm: categorization methods for user sessions on the web. In *Proc. of CHI* (2002).

[15] HOFGESANG, P. I., AND KOWALCZYK, W. Analysing clickstream data: From anomaly detection to visitor profiling. In *Proc. of ECML/PKDD Discovery Challenge* (2005).

[16] IRANI, D., BALDUZZI, M., BALZAROTTI, D., KIRDA, E., AND PU, C. Reverse social engineering attacks in online social networks. In *Proc of DIMVA* (2011).

[17] JIANG, J., WILSON, C., WANG, X., HUANG, P., SHA, W., DAI, Y., AND ZHAO, B. Y. Understanding latent interactions in online social networks. In *Proc. of IMC* (2010).

[18] KARYPIS, G., KUMAR, V., AND KUMAR, V. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing 48* (1998), 96–129.

[19] LEVANDOWSKY, M., AND WINTER, D. Distance between sets. *Nature 234* (1971), 34–35.

[20] LU, L., DUNHAM, M., AND MENG, Y. Mining significant usage patterns from clickstream data. In *Proc. of WebKDD* (2005).

[21] MOHAISEN, A., YUN, A., AND KIM, Y. Measuring the Mixing Time of Social Graphs. In *Proc. of IMC* (2010).

[22] MOTOYAMA, M., LEVCHENKO, K., KANICH, C., MCCOY, D., VOELKER, G. M., AND SAVAGE, S. Re: Captchas – understanding captcha-solving from an economic context. In *Proc. of USENIX Security* (2010).

[23] 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).

[24] OBENDORF, H., WEINREICH, H., HERDER, E., AND MAYER, M. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In *Proc. of CHI* (2007).

[25] PETRIDOU, S. G., KOUTSONIKOLA, V. A., VAKALI, A. I., AND PAPADIMITRIOU, G. I. Time-aware web users' clustering. *IEEE Trans. on Knowl. and Data Eng.* (2008), 653–667.

[26] PLATT, J. C. Advances in kernel methods. MIT Press, 1999, ch. Fast training of support vector machines using sequential minimal optimization, pp. 185–208.

[27] Russian twitter political protests 'swamped by spam'. BBC News, December 2011.

[28] SADAGOPAN, N., AND LI, J. Characterizing typical and atypical user sessions in clickstreams. In *Proc. of WWW* (2008).

[29] SCHNEIDER, F., FELDMANN, A., KRISHNAMURTHY, B., AND WILLINGER, W. Understanding online social network usage from a network perspective. In *Proc. of IMC* (2009).

[30] SRIVASTAVA, J., COOLEY, R., DESHPANDE, M., AND TAN, P. N. Web usage mining: discovery and applications of usage patterns from Web data. *SIGKDD Explor. Newsl. 1*, 2 (2000), 12–23.

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

[32] THOMAS, K., ET AL. Suspended accounts in retrospect: An analysis of twitter spam. In *Proc. of IMC* (2011).

[33] TING, I.-H., KIMBLE, C., AND KUDENKO, D. Ubb mining: Finding unexpected browsing behaviour in clickstream data to improve a web site's design. In *Proc. of International Conference on Web Intelligence* (2005).

[34] TRAN, N., MIN, B., LI, J., AND SUBRAMANIAN, L. Sybil-resilient online content voting. In *Proc. of NSDI* (2009).

[35] VEGA, C. Yelp outs companies that pay for positive reviews. ABC News, November 2012. `http://abcnews.go.com/blogs/business/2012/11/yelp-outs-companies-that-pay-for-positive-reviews`.

[36] VISWANATH, B., POST, A., GUMMADI, K. P., AND MISLOVE, A. An analysis of social network-based sybil defenses. In *Proc. of SIGCOMM* (2010).

[37] WANG, A. H. Don't follow me: Spam detection on twitter. In *Proc. of SECRYPT* (2010).

[38] 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).

[39] 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).

[40] WANG, W., AND ZAÏANE, O. R. Clustering web sessions by sequence alignment. In *Proc. of DEXA* (2002).

[41] WILSON, C., BOE, B., SALA, A., PUTTASWAMY, K. P. N., AND ZHAO, B. Y. User interactions in social networks and their implications. In *Proc. of EuroSys* (2009).

[42] XIAO, J., AND ZHANG, Y. Clustering of web users using session-based similarity measures. In *Proc. of ICCNMC* (2001).

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

[44] YARDI, S., ROMERO, D., SCHOENEBECK, G., AND BOYD, D. Detecting spam in a twitter network. *First Monday 15*, 1 (2010).

[45] YU, H., GIBBONS, P. B., KAMINSKY, M., AND XIAO, F. Sybillimit: A near-optimal social network defense against sybil attacks. In *Proc. of IEEE S&P* (2008).

[46] YU, H., KAMINSKY, M., GIBBONS, P. B., AND FLAXMAN, A. Sybilguard: defending against sybil attacks via social networks. In *Proc. of SIGCOMM* (2006).